Deep Learning (https://www.cc.gatech.edu/~hays/compvision/proj6/)

Setup

In [1]:
```
%matplotlib notebook
%load_ext autoreload
%autoreload 2
import cv2
import numpy as np
import random
import torch.nn as nn
import torch.optim as optim
import os.path as osp
import matplotlib.pyplot as plt
from utils import *
import student_code as sc
import torchvision.models as models
data_path = osp.join('../data', '15SceneData')
num_classes = 15

# If you have a good Nvidia GPU with an appropriate environment,
# try setting the use_GPU flag to True (the environment provided does
# not support GPUs and we will not provide any support for GPU
# computation in this project). Please note that
# we will evaluate your implementations only using CPU mode so even if
# you use a GPU, make sure your code runs in the CPU mode with the
# environment we provided.
use_GPU = False
if use_GPU:
    from utils_gpu import *
```

To train a network in PyTorch, we need 4 components:

1. **Dataset** - an object which can load the data and labels given an index.
2. **Model** - an object that contains the network architecture definition.
3. **Loss function** - a function that measures how far the network output is from the ground truth label.
4. **Optimizer** - an object that optimizes the network parameters to reduce the loss value.

This project has two main parts. In Part 1, you will train a deep network from scratch. In Part 2, you will "fine-tune" a trained network.

Part 0. Warm up! Training a Deep Network from Scratch

In [2]:
```
# Fix random seeds so that results will be reproducible
set_seed(0, use_GPU)
```

You do not need to code anything for this part. You will simply run the code we provided, but we want you to report the result you got. This section will also familiarize you with the steps of training a deep network from scratch.

In [3]:
```
# Training parameters.
input_size = (64, 64)
RGB = False
base_lr = 1e-2  # may try a smaller lr if not using batch norm
weight_decay = 5e-4
momentum = 0.9
```

We will first create our datasets, by calling the create_datasets function from student_code. This function returns a separate dataset loader for each split of the dataset (training and testing/validation). Each dataloader is used to load the datasets after appling some pre-processing transforms. In Part 1, you will be asked to add a few more pre-processing transforms to the dataloaders by modifying this function.

In [4]:
```
# Create the training and testing datasets.
train_dataset, test_dataset = sc.create_datasets(data_path=data_path, input_size=input_size, rgb=RGB)
assert test_dataset.classes == train_dataset.classes
```

```
Computing pixel mean and stdev...
Batch 0 / 30
Batch 20 / 30
Done, mean =
[0.45579668]
std =
[0.23624939]
Computing pixel mean and stdev...
Batch 0 / 60
Batch 20 / 60
Batch 40 / 60
Done, mean =
[0.45517009]
std =
[0.2350788]
```

Now we will create our network model using the SimpleNet class from student_code. The implementation provided in the SimpleNet class gives you a basic network. In Part 1, you will be asked to add a few more layers to this network.

In [5]:
```
# Create the network model.
model = sc.SimpleNet(num_classes=num_classes, rgb=False, verbose=False)
if use_GPU:
    model = model.cuda()
print(model)
```

```
SimpleNet(
  (features): Sequential(
    (0): Conv2d(1, 10, kernel_size=(9, 9), stride=(1, 1), bias=False)
    (1): MaxPool2d(kernel_size=7, stride=7, padding=0, dilation=1, ceil_mode=False)
    (2): ReLU()
  )
  (classifier): Conv2d(10, 15, kernel_size=(8, 8), stride=(1, 1))
)
```

Next we will create the loss function and the optimizer.

In [6]:
```
# Create the loss function.
# see http://pytorch.org/docs/0.3.0/nn.html#Loss-functions for a list of available loss functions
loss_function = nn.CrossEntropyLoss()
```

In [7]:
```
# Create the optimizer and a learning rate scheduler
optimizer = optim.SGD(params=model.parameters(), lr=base_lr, weight_decay=weight_decay, momentum=momen
tum)
# Currently a simple step scheduler.
# See http://pytorch.org/docs/0.3.0/optim.html#how-to-adjust-learning-rate for various LR schedulers
# and how to use them
lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=60, gamma=0.1)
```

Finally we are ready to train our network! We will start a local server to see the training progress of our network. Open a new terminal and activate the environment for this project. Then run the following command: **python -m visdom.server**. This will start a local server. The terminal output should give out a link like: "http://localhost:8097 (http://localhost:8097)". Open this link in your browser. After you run the following block, visit this link again, and you will be able to see graphs showing the progress of your training! If you do not see any graphs, select Part 1 on the top left bar where is says Environment (only select Part 1, do not check main or Part 2).

In [8]:
```
# train the network!
params = {'n_epochs': 100, 'batch_size': 50, 'experiment': 'part1'}
trainer = Trainer(train_dataset, test_dataset, model, loss_function, optimizer, lr_scheduler, params)
best_prec1 = trainer.train_val()
print('Best top-1 Accuracy = {:4.3f}'.format(best_prec1))
```

```
WARNING:root:Setting up a new session...
```

```
----------------------------------------
Experiment: part1
n_epochs: 100
batch_size: 50
do_val: True
shuffle: True
num_workers: 4
val_freq: 1
print_freq: 100
experiment: part1
checkpoint_file: None
resume_optim: True
----------------------------------------
part1 Epoch 0 / 100
train part1: batch 0/29, loss 2.709, top-1 accuracy 8.000, top-5 accuracy 28.000
train part1: loss 2.708517
val part1: batch 0/59, loss 2.720, top-1 accuracy 0.000, top-5 accuracy 0.000
val part1: loss 2.708254
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 1 / 100
train part1: batch 0/29, loss 2.706, top-1 accuracy 0.000, top-5 accuracy 42.000
train part1: loss 2.706191
val part1: batch 0/59, loss 2.649, top-1 accuracy 28.000, top-5 accuracy 100.000
val part1: loss 2.709384
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 2 / 100
train part1: batch 0/29, loss 2.703, top-1 accuracy 8.000, top-5 accuracy 34.000
train part1: loss 2.699871
val part1: batch 0/59, loss 2.666, top-1 accuracy 0.000, top-5 accuracy 22.000
val part1: loss 2.692278
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 3 / 100
train part1: batch 0/29, loss 2.656, top-1 accuracy 24.000, top-5 accuracy 44.000
train part1: loss 2.671104
val part1: batch 0/59, loss 2.608, top-1 accuracy 0.000, top-5 accuracy 32.000
val part1: loss 2.677120
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 4 / 100
train part1: batch 0/29, loss 2.571, top-1 accuracy 20.000, top-5 accuracy 64.000
train part1: loss 2.625427
val part1: batch 0/59, loss 2.588, top-1 accuracy 8.000, top-5 accuracy 42.000
val part1: loss 2.607369
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 5 / 100
train part1: batch 0/29, loss 2.434, top-1 accuracy 32.000, top-5 accuracy 56.000
train part1: loss 2.580147
val part1: batch 0/59, loss 2.632, top-1 accuracy 0.000, top-5 accuracy 40.000
val part1: loss 2.608572
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 6 / 100
train part1: batch 0/29, loss 2.526, top-1 accuracy 26.000, top-5 accuracy 54.000
train part1: loss 2.515079
val part1: batch 0/59, loss 2.843, top-1 accuracy 0.000, top-5 accuracy 32.000
val part1: loss 2.609508
Checkpoint saved
part1 Epoch 7 / 100
train part1: batch 0/29, loss 2.437, top-1 accuracy 24.000, top-5 accuracy 60.000
train part1: loss 2.465649
val part1: batch 0/59, loss 2.806, top-1 accuracy 0.000, top-5 accuracy 34.000
val part1: loss 2.471517
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 8 / 100
train part1: batch 0/29, loss 2.373, top-1 accuracy 26.000, top-5 accuracy 82.000
train part1: loss 2.355246
```

```
val part1: batch 0/59, loss 3.031, top-1 accuracy 0.000, top-5 accuracy 8.000
val part1: loss 2.449242
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 9 / 100
train part1: batch 0/29, loss 2.286, top-1 accuracy 32.000, top-5 accuracy 64.000
train part1: loss 2.317082
val part1: batch 0/59, loss 2.858, top-1 accuracy 0.000, top-5 accuracy 44.000
val part1: loss 2.435209
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 10 / 100
train part1: batch 0/29, loss 2.282, top-1 accuracy 32.000, top-5 accuracy 70.000
train part1: loss 2.214004
val part1: batch 0/59, loss 3.108, top-1 accuracy 0.000, top-5 accuracy 20.000
val part1: loss 2.414884
Checkpoint saved
part1 Epoch 11 / 100
train part1: batch 0/29, loss 2.143, top-1 accuracy 32.000, top-5 accuracy 74.000
train part1: loss 2.175274
val part1: batch 0/59, loss 2.661, top-1 accuracy 8.000, top-5 accuracy 50.000
val part1: loss 2.404811
Checkpoint saved
part1 Epoch 12 / 100
train part1: batch 0/29, loss 2.171, top-1 accuracy 24.000, top-5 accuracy 78.000
train part1: loss 2.105302
val part1: batch 0/59, loss 2.915, top-1 accuracy 8.000, top-5 accuracy 38.000
val part1: loss 2.462359
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 13 / 100
train part1: batch 0/29, loss 1.914, top-1 accuracy 42.000, top-5 accuracy 86.000
train part1: loss 2.043863
val part1: batch 0/59, loss 2.133, top-1 accuracy 22.000, top-5 accuracy 78.000
val part1: loss 2.359016
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 14 / 100
train part1: batch 0/29, loss 1.990, top-1 accuracy 40.000, top-5 accuracy 74.000
train part1: loss 1.962202
val part1: batch 0/59, loss 3.298, top-1 accuracy 2.000, top-5 accuracy 28.000
val part1: loss 2.367331
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 15 / 100
train part1: batch 0/29, loss 1.879, top-1 accuracy 32.000, top-5 accuracy 80.000
train part1: loss 1.921669
val part1: batch 0/59, loss 2.879, top-1 accuracy 16.000, top-5 accuracy 44.000
val part1: loss 2.353366
Checkpoint saved
part1 Epoch 16 / 100
train part1: batch 0/29, loss 1.852, top-1 accuracy 46.000, top-5 accuracy 80.000
train part1: loss 1.839318
val part1: batch 0/59, loss 2.816, top-1 accuracy 12.000, top-5 accuracy 52.000
val part1: loss 2.314891
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 17 / 100
train part1: batch 0/29, loss 1.827, top-1 accuracy 30.000, top-5 accuracy 84.000
train part1: loss 1.785016
val part1: batch 0/59, loss 2.155, top-1 accuracy 22.000, top-5 accuracy 76.000
val part1: loss 2.308254
Checkpoint saved
part1 Epoch 18 / 100
train part1: batch 0/29, loss 1.639, top-1 accuracy 50.000, top-5 accuracy 86.000
train part1: loss 1.729841
val part1: batch 0/59, loss 2.877, top-1 accuracy 10.000, top-5 accuracy 50.000
val part1: loss 2.282140
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 19 / 100
train part1: batch 0/29, loss 1.837, top-1 accuracy 42.000, top-5 accuracy 80.000
train part1: loss 1.698887
val part1: batch 0/59, loss 2.882, top-1 accuracy 10.000, top-5 accuracy 48.000
val part1: loss 2.203613
Checkpoint saved
part1 Epoch 20 / 100
train part1: batch 0/29, loss 1.426, top-1 accuracy 68.000, top-5 accuracy 86.000
train part1: loss 1.634228
val part1: batch 0/59, loss 2.792, top-1 accuracy 14.000, top-5 accuracy 64.000
val part1: loss 2.249531
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 21 / 100
train part1: batch 0/29, loss 1.380, top-1 accuracy 60.000, top-5 accuracy 92.000
train part1: loss 1.574890
val part1: batch 0/59, loss 2.463, top-1 accuracy 18.000, top-5 accuracy 70.000
val part1: loss 2.230502
Checkpoint saved
part1 Epoch 22 / 100
train part1: batch 0/29, loss 1.408, top-1 accuracy 52.000, top-5 accuracy 88.000
train part1: loss 1.536428
val part1: batch 0/59, loss 2.936, top-1 accuracy 12.000, top-5 accuracy 50.000
val part1: loss 2.168579
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 23 / 100
train part1: batch 0/29, loss 1.287, top-1 accuracy 58.000, top-5 accuracy 92.000
train part1: loss 1.506882
val part1: batch 0/59, loss 3.028, top-1 accuracy 14.000, top-5 accuracy 50.000
val part1: loss 2.252078
Checkpoint saved
part1 Epoch 24 / 100
train part1: batch 0/29, loss 1.475, top-1 accuracy 50.000, top-5 accuracy 90.000
train part1: loss 1.448166
val part1: batch 0/59, loss 2.616, top-1 accuracy 18.000, top-5 accuracy 62.000
val part1: loss 2.231408
Checkpoint saved
part1 Epoch 25 / 100
train part1: batch 0/29, loss 1.356, top-1 accuracy 60.000, top-5 accuracy 92.000
train part1: loss 1.423162
val part1: batch 0/59, loss 3.224, top-1 accuracy 16.000, top-5 accuracy 54.000
val part1: loss 2.271265
Checkpoint saved
part1 Epoch 26 / 100
train part1: batch 0/29, loss 1.569, top-1 accuracy 48.000, top-5 accuracy 92.000
train part1: loss 1.426793
val part1: batch 0/59, loss 3.013, top-1 accuracy 16.000, top-5 accuracy 56.000
val part1: loss 2.316300
Checkpoint saved
part1 Epoch 27 / 100
train part1: batch 0/29, loss 1.268, top-1 accuracy 64.000, top-5 accuracy 88.000
train part1: loss 1.397328
val part1: batch 0/59, loss 4.116, top-1 accuracy 6.000, top-5 accuracy 30.000
val part1: loss 2.249618
Checkpoint saved
part1 Epoch 28 / 100
train part1: batch 0/29, loss 1.155, top-1 accuracy 66.000, top-5 accuracy 92.000
train part1: loss 1.369537
val part1: batch 0/59, loss 3.726, top-1 accuracy 8.000, top-5 accuracy 46.000
val part1: loss 2.378837
Checkpoint saved
part1 Epoch 29 / 100
train part1: batch 0/29, loss 1.138, top-1 accuracy 60.000, top-5 accuracy 96.000
train part1: loss 1.317607
val part1: batch 0/59, loss 3.521, top-1 accuracy 12.000, top-5 accuracy 44.000
val part1: loss 2.362971
Checkpoint saved
part1 Epoch 30 / 100
train part1: batch 0/29, loss 1.279, top-1 accuracy 48.000, top-5 accuracy 98.000
train part1: loss 1.320762
val part1: batch 0/59, loss 2.914, top-1 accuracy 14.000, top-5 accuracy 60.000
val part1: loss 2.375174
Checkpoint saved
```

```
part1 Epoch 31 / 100
train part1: batch 0/29, loss 0.907, top-1 accuracy 70.000, top-5 accuracy 96.000
train part1: loss 1.306701
val part1: batch 0/59, loss 2.478, top-1 accuracy 28.000, top-5 accuracy 74.000
val part1: loss 2.394661
Checkpoint saved
part1 Epoch 32 / 100
train part1: batch 0/29, loss 1.109, top-1 accuracy 64.000, top-5 accuracy 96.000
train part1: loss 1.286650
val part1: batch 0/59, loss 2.521, top-1 accuracy 30.000, top-5 accuracy 68.000
val part1: loss 2.330312
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 33 / 100
train part1: batch 0/29, loss 1.302, top-1 accuracy 54.000, top-5 accuracy 90.000
train part1: loss 1.198231
val part1: batch 0/59, loss 3.228, top-1 accuracy 14.000, top-5 accuracy 60.000
val part1: loss 2.350145
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 34 / 100
train part1: batch 0/29, loss 1.168, top-1 accuracy 62.000, top-5 accuracy 92.000
train part1: loss 1.268760
val part1: batch 0/59, loss 2.581, top-1 accuracy 32.000, top-5 accuracy 74.000
val part1: loss 2.426174
Checkpoint saved
part1 Epoch 35 / 100
train part1: batch 0/29, loss 1.433, top-1 accuracy 56.000, top-5 accuracy 88.000
train part1: loss 1.249361
val part1: batch 0/59, loss 2.908, top-1 accuracy 24.000, top-5 accuracy 66.000
val part1: loss 2.362816
Checkpoint saved
part1 Epoch 36 / 100
train part1: batch 0/29, loss 1.040, top-1 accuracy 64.000, top-5 accuracy 92.000
train part1: loss 1.184333
val part1: batch 0/59, loss 3.480, top-1 accuracy 14.000, top-5 accuracy 60.000
val part1: loss 2.403720
Checkpoint saved
part1 Epoch 37 / 100
train part1: batch 0/29, loss 1.364, top-1 accuracy 56.000, top-5 accuracy 94.000
train part1: loss 1.176430
val part1: batch 0/59, loss 2.710, top-1 accuracy 28.000, top-5 accuracy 72.000
val part1: loss 2.503140
Checkpoint saved
part1 Epoch 38 / 100
train part1: batch 0/29, loss 0.881, top-1 accuracy 76.000, top-5 accuracy 98.000
train part1: loss 1.101327
val part1: batch 0/59, loss 3.351, top-1 accuracy 24.000, top-5 accuracy 56.000
val part1: loss 2.423237
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 39 / 100
train part1: batch 0/29, loss 0.984, top-1 accuracy 68.000, top-5 accuracy 94.000
train part1: loss 1.125916
val part1: batch 0/59, loss 2.918, top-1 accuracy 22.000, top-5 accuracy 68.000
val part1: loss 2.415714
Checkpoint saved
part1 Epoch 40 / 100
train part1: batch 0/29, loss 0.944, top-1 accuracy 64.000, top-5 accuracy 98.000
train part1: loss 1.068183
val part1: batch 0/59, loss 4.164, top-1 accuracy 12.000, top-5 accuracy 46.000
val part1: loss 2.477364
Checkpoint saved
part1 Epoch 41 / 100
train part1: batch 0/29, loss 1.349, top-1 accuracy 62.000, top-5 accuracy 86.000
train part1: loss 1.126097
val part1: batch 0/59, loss 3.528, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.421439
Checkpoint saved
part1 Epoch 42 / 100
train part1: batch 0/29, loss 1.004, top-1 accuracy 66.000, top-5 accuracy 100.000
train part1: loss 1.086174
val part1: batch 0/59, loss 2.932, top-1 accuracy 26.000, top-5 accuracy 72.000
val part1: loss 2.544626
Checkpoint saved
part1 Epoch 43 / 100
train part1: batch 0/29, loss 1.236, top-1 accuracy 68.000, top-5 accuracy 88.000
train part1: loss 1.078529
val part1: batch 0/59, loss 3.104, top-1 accuracy 22.000, top-5 accuracy 68.000
val part1: loss 2.524180
Checkpoint saved
part1 Epoch 44 / 100
train part1: batch 0/29, loss 1.057, top-1 accuracy 66.000, top-5 accuracy 96.000
train part1: loss 1.009818
val part1: batch 0/59, loss 4.239, top-1 accuracy 14.000, top-5 accuracy 42.000
val part1: loss 2.430260
Checkpoint saved
part1 Epoch 45 / 100
train part1: batch 0/29, loss 0.960, top-1 accuracy 70.000, top-5 accuracy 96.000
train part1: loss 0.981424
val part1: batch 0/59, loss 3.226, top-1 accuracy 26.000, top-5 accuracy 62.000
val part1: loss 2.550651
Checkpoint saved
part1 Epoch 46 / 100
train part1: batch 0/29, loss 0.820, top-1 accuracy 74.000, top-5 accuracy 96.000
train part1: loss 0.935307
val part1: batch 0/59, loss 3.239, top-1 accuracy 24.000, top-5 accuracy 64.000
val part1: loss 2.609238
Checkpoint saved
part1 Epoch 47 / 100
train part1: batch 0/29, loss 0.844, top-1 accuracy 76.000, top-5 accuracy 98.000
train part1: loss 0.926814
val part1: batch 0/59, loss 3.478, top-1 accuracy 20.000, top-5 accuracy 56.000
val part1: loss 2.479870
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 48 / 100
train part1: batch 0/29, loss 0.609, top-1 accuracy 86.000, top-5 accuracy 98.000
train part1: loss 0.921930
val part1: batch 0/59, loss 3.507, top-1 accuracy 22.000, top-5 accuracy 58.000
val part1: loss 2.555515
Checkpoint saved
part1 Epoch 49 / 100
train part1: batch 0/29, loss 0.599, top-1 accuracy 82.000, top-5 accuracy 100.000
train part1: loss 0.880634
val part1: batch 0/59, loss 3.963, top-1 accuracy 14.000, top-5 accuracy 52.000
val part1: loss 2.537087
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 50 / 100
train part1: batch 0/29, loss 0.834, top-1 accuracy 72.000, top-5 accuracy 98.000
train part1: loss 0.889839
val part1: batch 0/59, loss 4.294, top-1 accuracy 16.000, top-5 accuracy 56.000
val part1: loss 2.810930
Checkpoint saved
part1 Epoch 51 / 100
train part1: batch 0/29, loss 1.083, top-1 accuracy 66.000, top-5 accuracy 100.000
train part1: loss 0.852592
val part1: batch 0/59, loss 3.421, top-1 accuracy 22.000, top-5 accuracy 68.000
val part1: loss 2.660187
Checkpoint saved
part1 Epoch 52 / 100
train part1: batch 0/29, loss 0.832, top-1 accuracy 78.000, top-5 accuracy 94.000
train part1: loss 0.860566
val part1: batch 0/59, loss 4.710, top-1 accuracy 14.000, top-5 accuracy 44.000
val part1: loss 2.692811
Checkpoint saved
part1 Epoch 53 / 100
train part1: batch 0/29, loss 0.830, top-1 accuracy 74.000, top-5 accuracy 98.000
train part1: loss 0.839065
val part1: batch 0/59, loss 3.656, top-1 accuracy 20.000, top-5 accuracy 68.000
val part1: loss 2.764553
Checkpoint saved
part1 Epoch 54 / 100
```

```
train part1: batch 0/29, loss 0.866, top-1 accuracy 70.000, top-5 accuracy 98.000
train part1: loss 0.765641
val part1: batch 0/59, loss 4.626, top-1 accuracy 14.000, top-5 accuracy 42.000
val part1: loss 2.632757
Checkpoint saved
part1 Epoch 55 / 100
train part1: batch 0/29, loss 0.632, top-1 accuracy 80.000, top-5 accuracy 100.000
train part1: loss 0.745309
val part1: batch 0/59, loss 3.742, top-1 accuracy 20.000, top-5 accuracy 64.000
val part1: loss 2.825761
Checkpoint saved
part1 Epoch 56 / 100
train part1: batch 0/29, loss 0.953, top-1 accuracy 66.000, top-5 accuracy 98.000
train part1: loss 0.737354
val part1: batch 0/59, loss 3.869, top-1 accuracy 20.000, top-5 accuracy 64.000
val part1: loss 2.913077
Checkpoint saved
part1 Epoch 57 / 100
train part1: batch 0/29, loss 0.699, top-1 accuracy 74.000, top-5 accuracy 98.000
train part1: loss 0.682893
val part1: batch 0/59, loss 3.725, top-1 accuracy 20.000, top-5 accuracy 68.000
val part1: loss 2.745152
Checkpoint saved
part1 Epoch 58 / 100
train part1: batch 0/29, loss 0.451, top-1 accuracy 90.000, top-5 accuracy 100.000
train part1: loss 0.691142
val part1: batch 0/59, loss 4.262, top-1 accuracy 16.000, top-5 accuracy 60.000
val part1: loss 2.872676
Checkpoint saved
part1 Epoch 59 / 100
train part1: batch 0/29, loss 0.596, top-1 accuracy 82.000, top-5 accuracy 100.000
train part1: loss 0.698187
val part1: batch 0/59, loss 4.694, top-1 accuracy 16.000, top-5 accuracy 44.000
val part1: loss 2.857576
Checkpoint saved
part1 Epoch 60 / 100
train part1: batch 0/29, loss 0.748, top-1 accuracy 78.000, top-5 accuracy 96.000
train part1: loss 0.553523
val part1: batch 0/59, loss 4.140, top-1 accuracy 16.000, top-5 accuracy 56.000
val part1: loss 2.775068
Checkpoint saved
part1 Epoch 61 / 100
train part1: batch 0/29, loss 0.437, top-1 accuracy 90.000, top-5 accuracy 100.000
train part1: loss 0.514878
val part1: batch 0/59, loss 4.245, top-1 accuracy 16.000, top-5 accuracy 58.000
val part1: loss 2.781350
Checkpoint saved
part1 Epoch 62 / 100
train part1: batch 0/29, loss 0.500, top-1 accuracy 90.000, top-5 accuracy 100.000
train part1: loss 0.502227
val part1: batch 0/59, loss 4.226, top-1 accuracy 18.000, top-5 accuracy 58.000
val part1: loss 2.805928
Checkpoint saved
part1 Epoch 63 / 100
train part1: batch 0/29, loss 0.454, top-1 accuracy 90.000, top-5 accuracy 100.000
train part1: loss 0.499806
val part1: batch 0/59, loss 4.365, top-1 accuracy 18.000, top-5 accuracy 58.000
val part1: loss 2.796631
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 64 / 100
train part1: batch 0/29, loss 0.506, top-1 accuracy 84.000, top-5 accuracy 100.000
train part1: loss 0.491078
val part1: batch 0/59, loss 4.261, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.813836
Checkpoint saved
part1 Epoch 65 / 100
train part1: batch 0/29, loss 0.426, top-1 accuracy 86.000, top-5 accuracy 100.000
train part1: loss 0.488218
val part1: batch 0/59, loss 4.450, top-1 accuracy 16.000, top-5 accuracy 50.000
val part1: loss 2.791205
Checkpoint saved
part1 Epoch 66 / 100
train part1: batch 0/29, loss 0.561, top-1 accuracy 88.000, top-5 accuracy 98.000
train part1: loss 0.486210
val part1: batch 0/59, loss 4.222, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.807305
Checkpoint saved
part1 Epoch 67 / 100
train part1: batch 0/29, loss 0.444, top-1 accuracy 92.000, top-5 accuracy 100.000
train part1: loss 0.485753
val part1: batch 0/59, loss 4.309, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.819289
Checkpoint saved
part1 Epoch 68 / 100
train part1: batch 0/29, loss 0.585, top-1 accuracy 82.000, top-5 accuracy 98.000
train part1: loss 0.485771
val part1: batch 0/59, loss 4.295, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.800865
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 69 / 100
train part1: batch 0/29, loss 0.510, top-1 accuracy 88.000, top-5 accuracy 96.000
train part1: loss 0.478917
val part1: batch 0/59, loss 4.377, top-1 accuracy 16.000, top-5 accuracy 52.000
val part1: loss 2.813284
Checkpoint saved
part1 Epoch 70 / 100
train part1: batch 0/29, loss 0.493, top-1 accuracy 88.000, top-5 accuracy 100.000
train part1: loss 0.475962
val part1: batch 0/59, loss 4.286, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.816343
Checkpoint saved
part1 Epoch 71 / 100
train part1: batch 0/29, loss 0.308, top-1 accuracy 98.000, top-5 accuracy 98.000
train part1: loss 0.474362
val part1: batch 0/59, loss 4.359, top-1 accuracy 16.000, top-5 accuracy 56.000
val part1: loss 2.837672
Checkpoint saved
part1 Epoch 72 / 100
train part1: batch 0/29, loss 0.446, top-1 accuracy 86.000, top-5 accuracy 100.000
train part1: loss 0.474436
val part1: batch 0/59, loss 4.485, top-1 accuracy 16.000, top-5 accuracy 54.000
val part1: loss 2.814847
Checkpoint saved
part1 Epoch 73 / 100
train part1: batch 0/29, loss 0.463, top-1 accuracy 90.000, top-5 accuracy 100.000
train part1: loss 0.470581
val part1: batch 0/59, loss 4.553, top-1 accuracy 18.000, top-5 accuracy 52.000
val part1: loss 2.828946
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 74 / 100
train part1: batch 0/29, loss 0.469, top-1 accuracy 92.000, top-5 accuracy 98.000
train part1: loss 0.466536
val part1: batch 0/59, loss 4.294, top-1 accuracy 16.000, top-5 accuracy 58.000
val part1: loss 2.842773
Checkpoint saved
part1 Epoch 75 / 100
train part1: batch 0/29, loss 0.470, top-1 accuracy 90.000, top-5 accuracy 100.000
train part1: loss 0.466817
val part1: batch 0/59, loss 4.392, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.836829
Checkpoint saved
part1 Epoch 76 / 100
train part1: batch 0/29, loss 0.332, top-1 accuracy 92.000, top-5 accuracy 100.000
train part1: loss 0.461924
val part1: batch 0/59, loss 4.139, top-1 accuracy 20.000, top-5 accuracy 56.000
val part1: loss 2.864528
Checkpoint saved
part1 Epoch 77 / 100
train part1: batch 0/29, loss 0.312, top-1 accuracy 98.000, top-5 accuracy 100.000
train part1: loss 0.460124
val part1: batch 0/59, loss 4.502, top-1 accuracy 18.000, top-5 accuracy 54.000
```

```
val part1: loss 2.830907
Checkpoint saved
part1 Epoch 78 / 100
train part1: batch 0/29, loss 0.575, top-1 accuracy 90.000, top-5 accuracy 96.000
train part1: loss 0.459433
val part1: batch 0/59, loss 4.500, top-1 accuracy 18.000, top-5 accuracy 54.000
val part1: loss 2.842773
Checkpoint saved
part1 Epoch 79 / 100
train part1: batch 0/29, loss 0.399, top-1 accuracy 96.000, top-5 accuracy 100.000
train part1: loss 0.457860
val part1: batch 0/59, loss 4.238, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.857716
Checkpoint saved
part1 Epoch 80 / 100
train part1: batch 0/29, loss 0.353, top-1 accuracy 92.000, top-5 accuracy 100.000
train part1: loss 0.453134
val part1: batch 0/59, loss 4.378, top-1 accuracy 18.000, top-5 accuracy 54.000
val part1: loss 2.853244
Checkpoint saved
part1 Epoch 81 / 100
train part1: batch 0/29, loss 0.491, top-1 accuracy 84.000, top-5 accuracy 100.000
train part1: loss 0.456502
val part1: batch 0/59, loss 4.475, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.865270
Checkpoint saved
part1 Epoch 82 / 100
train part1: batch 0/29, loss 0.362, top-1 accuracy 94.000, top-5 accuracy 100.000
train part1: loss 0.452735
val part1: batch 0/59, loss 4.249, top-1 accuracy 18.000, top-5 accuracy 58.000
val part1: loss 2.860234
Checkpoint saved
part1 Epoch 83 / 100
train part1: batch 0/29, loss 0.336, top-1 accuracy 90.000, top-5 accuracy 100.000
train part1: loss 0.452950
val part1: batch 0/59, loss 4.152, top-1 accuracy 20.000, top-5 accuracy 58.000
val part1: loss 2.881641
Checkpoint saved
part1 Epoch 84 / 100
train part1: batch 0/29, loss 0.378, top-1 accuracy 96.000, top-5 accuracy 100.000
train part1: loss 0.447678
val part1: batch 0/59, loss 4.220, top-1 accuracy 18.000, top-5 accuracy 58.000
val part1: loss 2.866471
Checkpoint saved
part1 Epoch 85 / 100
train part1: batch 0/29, loss 0.378, top-1 accuracy 92.000, top-5 accuracy 98.000
train part1: loss 0.445843
val part1: batch 0/59, loss 4.593, top-1 accuracy 18.000, top-5 accuracy 52.000
val part1: loss 2.866296
Checkpoint saved
part1 Epoch 86 / 100
train part1: batch 0/29, loss 0.407, top-1 accuracy 92.000, top-5 accuracy 100.000
train part1: loss 0.446816
val part1: batch 0/59, loss 4.442, top-1 accuracy 16.000, top-5 accuracy 56.000
val part1: loss 2.885379
Checkpoint saved
part1 Epoch 87 / 100
train part1: batch 0/29, loss 0.451, top-1 accuracy 94.000, top-5 accuracy 100.000
train part1: loss 0.445781
val part1: batch 0/59, loss 4.287, top-1 accuracy 18.000, top-5 accuracy 58.000
val part1: loss 2.899134
Checkpoint saved
part1 Epoch 88 / 100
train part1: batch 0/29, loss 0.388, top-1 accuracy 94.000, top-5 accuracy 100.000
train part1: loss 0.439362
val part1: batch 0/59, loss 4.501, top-1 accuracy 18.000, top-5 accuracy 54.000
val part1: loss 2.869512
Checkpoint saved
part1 Epoch 89 / 100
train part1: batch 0/29, loss 0.480, top-1 accuracy 94.000, top-5 accuracy 98.000
train part1: loss 0.438773
val part1: batch 0/59, loss 4.213, top-1 accuracy 20.000, top-5 accuracy 60.000
val part1: loss 2.901951
Checkpoint saved
part1 Epoch 90 / 100
train part1: batch 0/29, loss 0.319, top-1 accuracy 98.000, top-5 accuracy 100.000
train part1: loss 0.438522
val part1: batch 0/59, loss 4.272, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.915737
Checkpoint saved
part1 Epoch 91 / 100
train part1: batch 0/29, loss 0.373, top-1 accuracy 94.000, top-5 accuracy 100.000
train part1: loss 0.437572
val part1: batch 0/59, loss 4.470, top-1 accuracy 18.000, top-5 accuracy 52.000
val part1: loss 2.891039
Checkpoint saved
part1 Epoch 92 / 100
train part1: batch 0/29, loss 0.382, top-1 accuracy 94.000, top-5 accuracy 100.000
train part1: loss 0.431352
val part1: batch 0/59, loss 4.566, top-1 accuracy 18.000, top-5 accuracy 52.000
val part1: loss 2.888387
Checkpoint saved
part1 Epoch 93 / 100
train part1: batch 0/29, loss 0.400, top-1 accuracy 92.000, top-5 accuracy 100.000
train part1: loss 0.431417
val part1: batch 0/59, loss 4.631, top-1 accuracy 18.000, top-5 accuracy 50.000
val part1: loss 2.896246
Checkpoint saved
part1 Epoch 94 / 100
train part1: batch 0/29, loss 0.391, top-1 accuracy 88.000, top-5 accuracy 100.000
train part1: loss 0.428473
val part1: batch 0/59, loss 4.665, top-1 accuracy 18.000, top-5 accuracy 54.000
val part1: loss 2.907819
Checkpoint saved
part1 Epoch 95 / 100
train part1: batch 0/29, loss 0.513, top-1 accuracy 90.000, top-5 accuracy 100.000
train part1: loss 0.427252
val part1: batch 0/59, loss 4.614, top-1 accuracy 18.000, top-5 accuracy 54.000
val part1: loss 2.908839
Checkpoint saved
part1 Epoch 96 / 100
train part1: batch 0/29, loss 0.421, top-1 accuracy 94.000, top-5 accuracy 98.000
train part1: loss 0.423548
val part1: batch 0/59, loss 4.149, top-1 accuracy 18.000, top-5 accuracy 60.000
val part1: loss 2.928829
Checkpoint saved
part1 Epoch 97 / 100
train part1: batch 0/29, loss 0.470, top-1 accuracy 92.000, top-5 accuracy 98.000
train part1: loss 0.423363
val part1: batch 0/59, loss 4.511, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.915327
Checkpoint saved
part1 Epoch 98 / 100
train part1: batch 0/29, loss 0.601, top-1 accuracy 82.000, top-5 accuracy 98.000
train part1: loss 0.420399
val part1: batch 0/59, loss 4.203, top-1 accuracy 18.000, top-5 accuracy 60.000
val part1: loss 2.939859
Checkpoint saved
part1 Epoch 99 / 100
train part1: batch 0/29, loss 0.453, top-1 accuracy 92.000, top-5 accuracy 100.000
train part1: loss 0.423848
val part1: batch 0/59, loss 4.454, top-1 accuracy 18.000, top-5 accuracy 56.000
val part1: loss 2.922803
Checkpoint saved
Best top-1 Accuracy = 36.281
```

Expect this code to take around 5 minutes on CPU or 3 minutes on GPU. Now you are ready to actually modify the functions we used to train our model. Before you move on, make sure to record the accuracy of your network from Part 0, and report it in your write up.

Part 1: Modifying the Dataloaders and the Simple Network create_datasets

In [9]:
```
# Fix random seeds so that results will be reproducible
set_seed(0, use_GPU)
```

Now you will modify the create_datasets function from student_code. You will add random left-right mirroring and normalization to the transformations applied to the training dataset. You will also add normalization to the transformations applied to the testing dataset.

In [10]:
```
# Create the training and testing datasets.
train_dataset, test_dataset = sc.create_datasets(data_path=data_path, input_size=input_size, rgb=RGB)
assert test_dataset.classes == train_dataset.classes
```

```
Computing pixel mean and stdev...
Batch 0 / 30
Batch 20 / 30
Done, mean =
[0.45579668]
std =
[0.23624939]
Computing pixel mean and stdev...
Batch 0 / 60
Batch 20 / 60
Batch 40 / 60
Done, mean =
[0.45517009]
std =
[0.2350788]
```

Now you will modify SimpleNet by adding droppout, batch normalization, and additional convolution/maxpool/relu layers. You should achieve an accuracy of at least **50%**. Make sure your network passes this threshold--it is required for full credit on this section!

You can also use the following two blocks to determine the stucture of your network.

In [11]:
```
# create the network model
model = sc.SimpleNet(num_classes=num_classes, rgb=False, verbose=False)
if use_GPU:
    model = model.cuda()
print(model)
```

```
SimpleNet(
  (features): Sequential(
    (0): Conv2d(1, 10, kernel_size=(9, 9), stride=(1, 1), bias=False)
    (1): BatchNorm2d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): MaxPool2d(kernel_size=7, stride=7, padding=0, dilation=1, ceil_mode=False)
    (3): ReLU()
    (4): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (5): BatchNorm2d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): Dropout(p=0.5)
  )
  (classifier): Conv2d(10, 15, kernel_size=(8, 8), stride=(1, 1))
)
```

In [12]:
```
# Use this block to determine the kernel size of the conv2d layer in the classifier
# first, set the kernel size of that conv2d layer to 1, and run this block
# then, use that size of input to the classifier printed by this block to
# go back and update the kernel size of the conv2d layer in the classifier
# Finally, run this block again and verify that the network output size is a scalar
# Don't forget to re-run the block above every time you update the SimpleNet class!
from torch.autograd import Variable
data, _ = train_dataset[0]
s = data.size()
data = Variable(data.view(1, *s))
if use_GPU:
    data = data.cuda()
out = model(data)
print('Network output size is ', out.size())
```

```
Network output size is  torch.Size([15])
```

Next we will create the loss function and the optimizer. You do not have to modify the custom_part1_trainer in student_code if you use the same loss_function, optimizer, scheduler and parameters (n_epoch, batch_size etc.) as provided in this notebook to hit the required threshold of 50% accuracy. If you changed any of these values, it is important that you modify this function in student_code since we will not be using the notebook you submit to evaluate.

In [13]:
```
# Set up the trainer. You can modify custom_part1_trainer in
# student_copy.py if you want to try different learning settings.
custom_part1_trainer = sc.custom_part1_trainer(model)

if custom_part1_trainer is None:
    # Create the loss function.
    # see http://pytorch.org/docs/0.3.0/nn.html#loss-functions for a list of available loss functions
    loss_function = nn.CrossEntropyLoss()

    # Create the optimizer and a learning rate scheduler.
    optimizer = optim.SGD(params=model.parameters(), lr=base_lr, weight_decay=weight_decay, momentum=m
omentum)
    # Currently a simple step scheduler, but you can get creative.
    # See http://pytorch.org/docs/0.3.0/optim.html#how-to-adjust-learning-rate for various LR schedule
rs
    # and how to use them
    lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=60, gamma=0.1)

    params = {'n_epochs': 100, 'batch_size': 50, 'experiment': 'part1'}

else:
    if 'loss_function' in custom_part1_trainer:
        loss_function = custom_part1_trainer['loss_function']
    if 'optimizer' in custom_part1_trainer:
        optimizer = custom_part1_trainer['optimizer']
    if 'lr_scheduler' in custom_part1_trainer:
        lr_scheduler = custom_part1_trainer['lr_scheduler']
    if 'params' in custom_part1_trainer:
        params = custom_part1_trainer['params']
```

We are ready to train our network! As before, we will start a local server to see the training progress of our network (if you server is already running, you should not start another one). Open a new terminal and activate the environment for this project. Then run the following command: **python -m visdom.server**. This will start a local server. The terminal output should give out a link like: "http://localhost:8097 (http://localhost:8097)". Open this link in your browser. After you run the following block, visit this link again, and you will be able to see graphs showing the progress of your training! If you do not see any graphs, select Part 1 on the top left bar where is says Environment (only select Part 1, do not check main or Part 2).

In [14]:
```
# Train the network!
trainer = Trainer(train_dataset, test_dataset, model, loss_function, optimizer, lr_scheduler, params)
best_prec1 = trainer.train_val()
print('Best top-1 Accuracy = {:4.3f}'.format(best_prec1))
```

```
WARNING:root:Setting up a new session...
```

```
----------------------------------------
Experiment: part1
n_epochs: 100
batch_size: 50
do_val: True
shuffle: True
num_workers: 4
val_freq: 1
print_freq: 100
experiment: part1
checkpoint_file: None
resume_optim: True
----------------------------------------
part1 Epoch 0 / 100
train part1: batch 0/29, loss 2.709, top-1 accuracy 10.000, top-5 accuracy 30.000
train part1: loss 2.526204
val part1: batch 0/59, loss 1.792, top-1 accuracy 38.000, top-5 accuracy 78.000
val part1: loss 2.607826
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 1 / 100
train part1: batch 0/29, loss 2.497, top-1 accuracy 28.000, top-5 accuracy 62.000
train part1: loss 2.197321
val part1: batch 0/59, loss 2.703, top-1 accuracy 16.000, top-5 accuracy 60.000
val part1: loss 2.405490
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 2 / 100
train part1: batch 0/29, loss 2.066, top-1 accuracy 28.000, top-5 accuracy 76.000
train part1: loss 1.998264
val part1: batch 0/59, loss 2.565, top-1 accuracy 20.000, top-5 accuracy 54.000
val part1: loss 2.119356
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 3 / 100
train part1: batch 0/29, loss 2.004, top-1 accuracy 30.000, top-5 accuracy 74.000
train part1: loss 1.876626
val part1: batch 0/59, loss 2.131, top-1 accuracy 26.000, top-5 accuracy 68.000
val part1: loss 1.827269
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 4 / 100
train part1: batch 0/29, loss 1.851, top-1 accuracy 34.000, top-5 accuracy 82.000
train part1: loss 1.767197
val part1: batch 0/59, loss 2.195, top-1 accuracy 24.000, top-5 accuracy 74.000
val part1: loss 1.744641
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 5 / 100
train part1: batch 0/29, loss 1.756, top-1 accuracy 48.000, top-5 accuracy 82.000
train part1: loss 1.746011
val part1: batch 0/59, loss 2.086, top-1 accuracy 24.000, top-5 accuracy 74.000
val part1: loss 1.802269
Checkpoint saved
part1 Epoch 6 / 100
train part1: batch 0/29, loss 1.672, top-1 accuracy 60.000, top-5 accuracy 88.000
train part1: loss 1.641729
val part1: batch 0/59, loss 2.201, top-1 accuracy 26.000, top-5 accuracy 70.000
val part1: loss 1.619978
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 7 / 100
train part1: batch 0/29, loss 1.906, top-1 accuracy 38.000, top-5 accuracy 76.000
train part1: loss 1.641038
val part1: batch 0/59, loss 2.004, top-1 accuracy 28.000, top-5 accuracy 80.000
val part1: loss 1.797812
Checkpoint saved
part1 Epoch 8 / 100
train part1: batch 0/29, loss 1.838, top-1 accuracy 50.000, top-5 accuracy 84.000
train part1: loss 1.620432
val part1: batch 0/59, loss 1.654, top-1 accuracy 44.000, top-5 accuracy 88.000
val part1: loss 1.743424
Checkpoint saved
part1 Epoch 9 / 100
train part1: batch 0/29, loss 1.425, top-1 accuracy 50.000, top-5 accuracy 90.000
train part1: loss 1.546619
val part1: batch 0/59, loss 1.810, top-1 accuracy 36.000, top-5 accuracy 80.000
val part1: loss 1.612047
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 10 / 100
train part1: batch 0/29, loss 1.594, top-1 accuracy 44.000, top-5 accuracy 88.000
train part1: loss 1.514281
val part1: batch 0/59, loss 1.912, top-1 accuracy 38.000, top-5 accuracy 76.000
val part1: loss 1.621186
Checkpoint saved
part1 Epoch 11 / 100
train part1: batch 0/29, loss 1.616, top-1 accuracy 32.000, top-5 accuracy 90.000
train part1: loss 1.514912
val part1: batch 0/59, loss 1.668, top-1 accuracy 42.000, top-5 accuracy 88.000
val part1: loss 1.553640
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 12 / 100
train part1: batch 0/29, loss 1.478, top-1 accuracy 56.000, top-5 accuracy 90.000
train part1: loss 1.451927
val part1: batch 0/59, loss 1.518, top-1 accuracy 44.000, top-5 accuracy 94.000
val part1: loss 1.642554
Checkpoint saved
part1 Epoch 13 / 100
train part1: batch 0/29, loss 1.442, top-1 accuracy 54.000, top-5 accuracy 88.000
train part1: loss 1.460499
val part1: batch 0/59, loss 2.086, top-1 accuracy 32.000, top-5 accuracy 70.000
val part1: loss 1.565526
Checkpoint saved
part1 Epoch 14 / 100
train part1: batch 0/29, loss 1.260, top-1 accuracy 64.000, top-5 accuracy 88.000
train part1: loss 1.446994
val part1: batch 0/59, loss 2.079, top-1 accuracy 30.000, top-5 accuracy 72.000
val part1: loss 1.570353
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 15 / 100
train part1: batch 0/29, loss 1.771, top-1 accuracy 44.000, top-5 accuracy 86.000
train part1: loss 1.412237
val part1: batch 0/59, loss 1.675, top-1 accuracy 46.000, top-5 accuracy 84.000
val part1: loss 1.620559
Checkpoint saved
part1 Epoch 16 / 100
train part1: batch 0/29, loss 1.425, top-1 accuracy 54.000, top-5 accuracy 88.000
train part1: loss 1.414859
val part1: batch 0/59, loss 1.857, top-1 accuracy 32.000, top-5 accuracy 84.000
val part1: loss 1.688472
Checkpoint saved
part1 Epoch 17 / 100
train part1: batch 0/29, loss 1.623, top-1 accuracy 52.000, top-5 accuracy 86.000
train part1: loss 1.374372
val part1: batch 0/59, loss 1.712, top-1 accuracy 44.000, top-5 accuracy 84.000
val part1: loss 1.603852
Checkpoint saved
part1 Epoch 18 / 100
train part1: batch 0/29, loss 1.396, top-1 accuracy 50.000, top-5 accuracy 92.000
train part1: loss 1.402285
val part1: batch 0/59, loss 1.880, top-1 accuracy 34.000, top-5 accuracy 80.000
val part1: loss 1.552677
Checkpoint saved
part1 Epoch 19 / 100
train part1: batch 0/29, loss 1.240, top-1 accuracy 56.000, top-5 accuracy 94.000
train part1: loss 1.355988
val part1: batch 0/59, loss 2.054, top-1 accuracy 24.000, top-5 accuracy 78.000
val part1: loss 1.599527
Checkpoint saved
part1 Epoch 20 / 100
```

```
train part1: batch 0/29, loss 1.259, top-1 accuracy 58.000, top-5 accuracy 94.000
train part1: loss 1.346608
val part1: batch 0/59, loss 2.239, top-1 accuracy 28.000, top-5 accuracy 68.000
val part1: loss 1.525281
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 21 / 100
train part1: batch 0/29, loss 1.240, top-1 accuracy 58.000, top-5 accuracy 90.000
train part1: loss 1.309893
val part1: batch 0/59, loss 1.883, top-1 accuracy 28.000, top-5 accuracy 82.000
val part1: loss 1.538270
Checkpoint saved
part1 Epoch 22 / 100
train part1: batch 0/29, loss 1.015, top-1 accuracy 66.000, top-5 accuracy 96.000
train part1: loss 1.285833
val part1: batch 0/59, loss 2.334, top-1 accuracy 28.000, top-5 accuracy 66.000
val part1: loss 1.644277
Checkpoint saved
part1 Epoch 23 / 100
train part1: batch 0/29, loss 1.225, top-1 accuracy 62.000, top-5 accuracy 92.000
train part1: loss 1.294786
val part1: batch 0/59, loss 1.920, top-1 accuracy 28.000, top-5 accuracy 74.000
val part1: loss 1.561747
Checkpoint saved
part1 Epoch 24 / 100
train part1: batch 0/29, loss 1.354, top-1 accuracy 62.000, top-5 accuracy 92.000
train part1: loss 1.281434
val part1: batch 0/59, loss 2.311, top-1 accuracy 28.000, top-5 accuracy 62.000
val part1: loss 1.485095
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 25 / 100
train part1: batch 0/29, loss 1.182, top-1 accuracy 70.000, top-5 accuracy 96.000
train part1: loss 1.277545
val part1: batch 0/59, loss 1.712, top-1 accuracy 36.000, top-5 accuracy 82.000
val part1: loss 1.545796
Checkpoint saved
part1 Epoch 26 / 100
train part1: batch 0/29, loss 1.203, top-1 accuracy 58.000, top-5 accuracy 96.000
train part1: loss 1.282829
val part1: batch 0/59, loss 2.116, top-1 accuracy 32.000, top-5 accuracy 74.000
val part1: loss 1.483925
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 27 / 100
train part1: batch 0/29, loss 1.131, top-1 accuracy 66.000, top-5 accuracy 96.000
train part1: loss 1.248637
val part1: batch 0/59, loss 2.133, top-1 accuracy 28.000, top-5 accuracy 74.000
val part1: loss 1.492317
Checkpoint saved
part1 Epoch 28 / 100
train part1: batch 0/29, loss 0.997, top-1 accuracy 78.000, top-5 accuracy 92.000
train part1: loss 1.218901
val part1: batch 0/59, loss 2.382, top-1 accuracy 20.000, top-5 accuracy 68.000
val part1: loss 1.580590
Checkpoint saved
part1 Epoch 29 / 100
train part1: batch 0/29, loss 1.314, top-1 accuracy 66.000, top-5 accuracy 90.000
train part1: loss 1.239330
val part1: batch 0/59, loss 2.284, top-1 accuracy 22.000, top-5 accuracy 72.000
val part1: loss 1.487132
Checkpoint saved
part1 Epoch 30 / 100
train part1: batch 0/29, loss 1.132, top-1 accuracy 60.000, top-5 accuracy 94.000
train part1: loss 1.184061
val part1: batch 0/59, loss 2.265, top-1 accuracy 22.000, top-5 accuracy 68.000
val part1: loss 1.504421
Checkpoint saved
part1 Epoch 31 / 100
train part1: batch 0/29, loss 1.107, top-1 accuracy 66.000, top-5 accuracy 94.000
train part1: loss 1.217686
val part1: batch 0/59, loss 2.036, top-1 accuracy 26.000, top-5 accuracy 84.000
val part1: loss 1.469148
Checkpoint saved
part1 Epoch 32 / 100
train part1: batch 0/29, loss 1.149, top-1 accuracy 62.000, top-5 accuracy 94.000
train part1: loss 1.162077
val part1: batch 0/59, loss 1.944, top-1 accuracy 36.000, top-5 accuracy 78.000
val part1: loss 1.502160
Checkpoint saved
part1 Epoch 33 / 100
train part1: batch 0/29, loss 1.850, top-1 accuracy 46.000, top-5 accuracy 82.000
train part1: loss 1.231292
val part1: batch 0/59, loss 2.045, top-1 accuracy 28.000, top-5 accuracy 76.000
val part1: loss 1.570987
Checkpoint saved
part1 Epoch 34 / 100
train part1: batch 0/29, loss 1.349, top-1 accuracy 60.000, top-5 accuracy 94.000
train part1: loss 1.177927
val part1: batch 0/59, loss 2.125, top-1 accuracy 32.000, top-5 accuracy 78.000
val part1: loss 1.537418
Checkpoint saved
part1 Epoch 35 / 100
train part1: batch 0/29, loss 1.179, top-1 accuracy 70.000, top-5 accuracy 92.000
train part1: loss 1.170895
val part1: batch 0/59, loss 2.110, top-1 accuracy 30.000, top-5 accuracy 76.000
val part1: loss 1.491258
Checkpoint saved
part1 Epoch 36 / 100
train part1: batch 0/29, loss 0.998, top-1 accuracy 62.000, top-5 accuracy 98.000
train part1: loss 1.148513
val part1: batch 0/59, loss 2.192, top-1 accuracy 32.000, top-5 accuracy 74.000
val part1: loss 1.466092
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 37 / 100
train part1: batch 0/29, loss 0.971, top-1 accuracy 64.000, top-5 accuracy 98.000
train part1: loss 1.113993
val part1: batch 0/59, loss 1.852, top-1 accuracy 38.000, top-5 accuracy 78.000
val part1: loss 1.427701
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 38 / 100
train part1: batch 0/29, loss 0.881, top-1 accuracy 70.000, top-5 accuracy 98.000
train part1: loss 1.127537
val part1: batch 0/59, loss 2.040, top-1 accuracy 28.000, top-5 accuracy 76.000
val part1: loss 1.429692
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 39 / 100
train part1: batch 0/29, loss 1.089, top-1 accuracy 60.000, top-5 accuracy 100.000
train part1: loss 1.122451
val part1: batch 0/59, loss 2.044, top-1 accuracy 24.000, top-5 accuracy 84.000
val part1: loss 1.474594
Checkpoint saved
part1 Epoch 40 / 100
train part1: batch 0/29, loss 1.459, top-1 accuracy 48.000, top-5 accuracy 94.000
train part1: loss 1.121451
val part1: batch 0/59, loss 2.124, top-1 accuracy 32.000, top-5 accuracy 82.000
val part1: loss 1.385009
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 41 / 100
train part1: batch 0/29, loss 1.107, top-1 accuracy 62.000, top-5 accuracy 94.000
train part1: loss 1.131356
val part1: batch 0/59, loss 1.994, top-1 accuracy 30.000, top-5 accuracy 88.000
val part1: loss 1.488342
Checkpoint saved
part1 Epoch 42 / 100
train part1: batch 0/29, loss 1.154, top-1 accuracy 60.000, top-5 accuracy 98.000
train part1: loss 1.095674
val part1: batch 0/59, loss 2.132, top-1 accuracy 30.000, top-5 accuracy 74.000
val part1: loss 1.484323
Checkpoint saved
```

```
part1 Epoch 43 / 100
train part1: batch 0/29, loss 0.997, top-1 accuracy 76.000, top-5 accuracy 94.000
train part1: loss 1.046971
val part1: batch 0/59, loss 2.300, top-1 accuracy 30.000, top-5 accuracy 82.000
val part1: loss 1.448524
Checkpoint saved
part1 Epoch 44 / 100
train part1: batch 0/29, loss 0.954, top-1 accuracy 64.000, top-5 accuracy 98.000
train part1: loss 1.070462
val part1: batch 0/59, loss 1.826, top-1 accuracy 40.000, top-5 accuracy 86.000
val part1: loss 1.368686
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 45 / 100
train part1: batch 0/29, loss 0.950, top-1 accuracy 68.000, top-5 accuracy 94.000
train part1: loss 1.052588
val part1: batch 0/59, loss 3.326, top-1 accuracy 10.000, top-5 accuracy 52.000
val part1: loss 1.474223
Checkpoint saved
part1 Epoch 46 / 100
train part1: batch 0/29, loss 1.026, top-1 accuracy 68.000, top-5 accuracy 100.000
train part1: loss 1.120148
val part1: batch 0/59, loss 2.341, top-1 accuracy 22.000, top-5 accuracy 68.000
val part1: loss 1.475900
Checkpoint saved
part1 Epoch 47 / 100
train part1: batch 0/29, loss 0.984, top-1 accuracy 60.000, top-5 accuracy 98.000
train part1: loss 1.071774
val part1: batch 0/59, loss 2.232, top-1 accuracy 24.000, top-5 accuracy 72.000
val part1: loss 1.472998
Checkpoint saved
part1 Epoch 48 / 100
train part1: batch 0/29, loss 0.855, top-1 accuracy 72.000, top-5 accuracy 100.000
train part1: loss 1.073832
val part1: batch 0/59, loss 2.280, top-1 accuracy 28.000, top-5 accuracy 68.000
val part1: loss 1.515829
Checkpoint saved
part1 Epoch 49 / 100
train part1: batch 0/29, loss 0.907, top-1 accuracy 66.000, top-5 accuracy 98.000
train part1: loss 1.097865
val part1: batch 0/59, loss 2.425, top-1 accuracy 28.000, top-5 accuracy 66.000
val part1: loss 1.519148
Checkpoint saved
part1 Epoch 50 / 100
train part1: batch 0/29, loss 1.203, top-1 accuracy 56.000, top-5 accuracy 92.000
train part1: loss 1.132024
val part1: batch 0/59, loss 2.203, top-1 accuracy 36.000, top-5 accuracy 74.000
val part1: loss 1.917147
Checkpoint saved
part1 Epoch 51 / 100
train part1: batch 0/29, loss 0.879, top-1 accuracy 66.000, top-5 accuracy 100.000
train part1: loss 1.096248
val part1: batch 0/59, loss 2.080, top-1 accuracy 32.000, top-5 accuracy 78.000
val part1: loss 1.452121
Checkpoint saved
part1 Epoch 52 / 100
train part1: batch 0/29, loss 1.177, top-1 accuracy 62.000, top-5 accuracy 96.000
train part1: loss 1.041504
val part1: batch 0/59, loss 2.598, top-1 accuracy 20.000, top-5 accuracy 70.000
val part1: loss 1.388009
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 53 / 100
train part1: batch 0/29, loss 1.041, top-1 accuracy 62.000, top-5 accuracy 94.000
train part1: loss 1.011943
val part1: batch 0/59, loss 2.226, top-1 accuracy 28.000, top-5 accuracy 74.000
val part1: loss 1.410900
Checkpoint saved
part1 Epoch 54 / 100
train part1: batch 0/29, loss 1.117, top-1 accuracy 66.000, top-5 accuracy 94.000
train part1: loss 1.027942
val part1: batch 0/59, loss 1.498, top-1 accuracy 46.000, top-5 accuracy 96.000
val part1: loss 1.427343
Checkpoint saved
part1 Epoch 55 / 100
train part1: batch 0/29, loss 1.079, top-1 accuracy 64.000, top-5 accuracy 94.000
train part1: loss 0.968187
val part1: batch 0/59, loss 2.345, top-1 accuracy 30.000, top-5 accuracy 74.000
val part1: loss 1.390273
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 56 / 100
train part1: batch 0/29, loss 0.843, top-1 accuracy 72.000, top-5 accuracy 96.000
train part1: loss 1.011828
val part1: batch 0/59, loss 2.667, top-1 accuracy 22.000, top-5 accuracy 66.000
val part1: loss 1.518940
Checkpoint saved
part1 Epoch 57 / 100
train part1: batch 0/29, loss 0.757, top-1 accuracy 80.000, top-5 accuracy 100.000
train part1: loss 0.985345
val part1: batch 0/59, loss 2.723, top-1 accuracy 16.000, top-5 accuracy 64.000
val part1: loss 1.379713
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 58 / 100
train part1: batch 0/29, loss 1.204, top-1 accuracy 60.000, top-5 accuracy 92.000
train part1: loss 1.021387
val part1: batch 0/59, loss 2.398, top-1 accuracy 16.000, top-5 accuracy 72.000
val part1: loss 1.392419
Checkpoint saved
part1 Epoch 59 / 100
train part1: batch 0/29, loss 1.005, top-1 accuracy 66.000, top-5 accuracy 94.000
train part1: loss 0.974134
val part1: batch 0/59, loss 2.183, top-1 accuracy 32.000, top-5 accuracy 80.000
val part1: loss 1.408351
Checkpoint saved
part1 Epoch 60 / 100
train part1: batch 0/29, loss 0.806, top-1 accuracy 72.000, top-5 accuracy 94.000
train part1: loss 0.940303
val part1: batch 0/59, loss 1.759, top-1 accuracy 38.000, top-5 accuracy 86.000
val part1: loss 1.308888
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 61 / 100
train part1: batch 0/29, loss 0.860, top-1 accuracy 70.000, top-5 accuracy 98.000
train part1: loss 0.872999
val part1: batch 0/59, loss 1.848, top-1 accuracy 36.000, top-5 accuracy 82.000
val part1: loss 1.295457
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 62 / 100
train part1: batch 0/29, loss 0.895, top-1 accuracy 74.000, top-5 accuracy 96.000
train part1: loss 0.850647
val part1: batch 0/59, loss 2.008, top-1 accuracy 30.000, top-5 accuracy 80.000
val part1: loss 1.290004
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 63 / 100
train part1: batch 0/29, loss 1.071, top-1 accuracy 66.000, top-5 accuracy 94.000
train part1: loss 0.850959
val part1: batch 0/59, loss 2.010, top-1 accuracy 32.000, top-5 accuracy 80.000
val part1: loss 1.286640
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 64 / 100
train part1: batch 0/29, loss 0.827, top-1 accuracy 76.000, top-5 accuracy 98.000
train part1: loss 0.855319
val part1: batch 0/59, loss 2.018, top-1 accuracy 32.000, top-5 accuracy 82.000
val part1: loss 1.279645
Checkpoint saved
part1 Epoch 65 / 100
train part1: batch 0/29, loss 0.946, top-1 accuracy 66.000, top-5 accuracy 100.000
train part1: loss 0.832714
val part1: batch 0/59, loss 2.065, top-1 accuracy 30.000, top-5 accuracy 80.000
```

```
val part1: loss 1.274930
Checkpoint saved
part1 Epoch 66 / 100
train part1: batch 0/29, loss 0.672, top-1 accuracy 74.000, top-5 accuracy 98.000
train part1: loss 0.861875
val part1: batch 0/59, loss 2.102, top-1 accuracy 28.000, top-5 accuracy 78.000
val part1: loss 1.290362
Checkpoint saved
part1 Epoch 67 / 100
train part1: batch 0/29, loss 0.820, top-1 accuracy 70.000, top-5 accuracy 96.000
train part1: loss 0.867240
val part1: batch 0/59, loss 2.126, top-1 accuracy 30.000, top-5 accuracy 80.000
val part1: loss 1.270799
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 68 / 100
train part1: batch 0/29, loss 0.718, top-1 accuracy 76.000, top-5 accuracy 100.000
train part1: loss 0.871294
val part1: batch 0/59, loss 2.098, top-1 accuracy 30.000, top-5 accuracy 80.000
val part1: loss 1.276019
Checkpoint saved
part1 Epoch 69 / 100
train part1: batch 0/29, loss 0.895, top-1 accuracy 72.000, top-5 accuracy 98.000
train part1: loss 0.851349
val part1: batch 0/59, loss 2.019, top-1 accuracy 32.000, top-5 accuracy 78.000
val part1: loss 1.272459
Checkpoint saved
part1 Epoch 70 / 100
train part1: batch 0/29, loss 1.072, top-1 accuracy 64.000, top-5 accuracy 96.000
train part1: loss 0.852901
val part1: batch 0/59, loss 2.064, top-1 accuracy 32.000, top-5 accuracy 80.000
val part1: loss 1.283682
Checkpoint saved
part1 Epoch 71 / 100
train part1: batch 0/29, loss 1.014, top-1 accuracy 72.000, top-5 accuracy 98.000
train part1: loss 0.869324
val part1: batch 0/59, loss 2.028, top-1 accuracy 30.000, top-5 accuracy 80.000
val part1: loss 1.269841
Checkpoint saved
part1 Epoch 72 / 100
train part1: batch 0/29, loss 0.851, top-1 accuracy 72.000, top-5 accuracy 100.000
train part1: loss 0.861725
val part1: batch 0/59, loss 2.000, top-1 accuracy 30.000, top-5 accuracy 82.000
val part1: loss 1.275700
Checkpoint saved
part1 Epoch 73 / 100
train part1: batch 0/29, loss 0.732, top-1 accuracy 78.000, top-5 accuracy 96.000
train part1: loss 0.844961
val part1: batch 0/59, loss 2.081, top-1 accuracy 32.000, top-5 accuracy 82.000
val part1: loss 1.278939
Checkpoint saved
part1 Epoch 74 / 100
train part1: batch 0/29, loss 0.758, top-1 accuracy 74.000, top-5 accuracy 98.000
train part1: loss 0.871322
val part1: batch 0/59, loss 1.982, top-1 accuracy 32.000, top-5 accuracy 82.000
val part1: loss 1.268409
Checkpoint saved
part1 Epoch 75 / 100
train part1: batch 0/29, loss 0.707, top-1 accuracy 82.000, top-5 accuracy 100.000
train part1: loss 0.833059
val part1: batch 0/59, loss 2.037, top-1 accuracy 32.000, top-5 accuracy 82.000
val part1: loss 1.267296
Checkpoint saved
part1 Epoch 76 / 100
train part1: batch 0/29, loss 0.792, top-1 accuracy 78.000, top-5 accuracy 100.000
train part1: loss 0.842629
val part1: batch 0/59, loss 1.960, top-1 accuracy 32.000, top-5 accuracy 82.000
val part1: loss 1.273829
Checkpoint saved
part1 Epoch 77 / 100
train part1: batch 0/29, loss 1.061, top-1 accuracy 78.000, top-5 accuracy 96.000
train part1: loss 0.831131
val part1: batch 0/59, loss 2.067, top-1 accuracy 30.000, top-5 accuracy 82.000
val part1: loss 1.268330
Checkpoint saved
part1 Epoch 78 / 100
train part1: batch 0/29, loss 0.906, top-1 accuracy 62.000, top-5 accuracy 98.000
train part1: loss 0.845808
val part1: batch 0/59, loss 2.059, top-1 accuracy 32.000, top-5 accuracy 82.000
val part1: loss 1.271593
Checkpoint saved
part1 Epoch 79 / 100
train part1: batch 0/29, loss 0.886, top-1 accuracy 76.000, top-5 accuracy 98.000
train part1: loss 0.827485
val part1: batch 0/59, loss 2.008, top-1 accuracy 34.000, top-5 accuracy 82.000
val part1: loss 1.271513
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 80 / 100
train part1: batch 0/29, loss 1.102, top-1 accuracy 66.000, top-5 accuracy 94.000
train part1: loss 0.803301
val part1: batch 0/59, loss 2.130, top-1 accuracy 28.000, top-5 accuracy 78.000
val part1: loss 1.276848
Checkpoint saved
part1 Epoch 81 / 100
train part1: batch 0/29, loss 0.818, top-1 accuracy 76.000, top-5 accuracy 98.000
train part1: loss 0.805922
val part1: batch 0/59, loss 1.930, top-1 accuracy 32.000, top-5 accuracy 82.000
val part1: loss 1.294209
Checkpoint saved
part1 Epoch 82 / 100
train part1: batch 0/29, loss 0.681, top-1 accuracy 78.000, top-5 accuracy 100.000
train part1: loss 0.804935
val part1: batch 0/59, loss 2.023, top-1 accuracy 26.000, top-5 accuracy 82.000
val part1: loss 1.266775
Checkpoint saved
part1 Epoch 83 / 100
train part1: batch 0/29, loss 0.816, top-1 accuracy 70.000, top-5 accuracy 98.000
train part1: loss 0.833163
val part1: batch 0/59, loss 2.048, top-1 accuracy 28.000, top-5 accuracy 82.000
val part1: loss 1.269629
Checkpoint saved
part1 Epoch 84 / 100
train part1: batch 0/29, loss 0.714, top-1 accuracy 76.000, top-5 accuracy 98.000
train part1: loss 0.881449
val part1: batch 0/59, loss 1.990, top-1 accuracy 28.000, top-5 accuracy 80.000
val part1: loss 1.276584
Checkpoint saved
part1 Epoch 85 / 100
train part1: batch 0/29, loss 0.922, top-1 accuracy 66.000, top-5 accuracy 98.000
train part1: loss 0.828473
val part1: batch 0/59, loss 2.082, top-1 accuracy 30.000, top-5 accuracy 82.000
val part1: loss 1.268415
Checkpoint saved
part1 Epoch 86 / 100
train part1: batch 0/29, loss 0.912, top-1 accuracy 72.000, top-5 accuracy 98.000
train part1: loss 0.838253
val part1: batch 0/59, loss 2.055, top-1 accuracy 30.000, top-5 accuracy 82.000
val part1: loss 1.268436
Checkpoint saved
part1 Epoch 87 / 100
train part1: batch 0/29, loss 0.680, top-1 accuracy 78.000, top-5 accuracy 98.000
train part1: loss 0.818667
val part1: batch 0/59, loss 1.986, top-1 accuracy 30.000, top-5 accuracy 84.000
val part1: loss 1.285363
Checkpoint saved
part1 Epoch 88 / 100
train part1: batch 0/29, loss 0.799, top-1 accuracy 76.000, top-5 accuracy 98.000
train part1: loss 0.813989
val part1: batch 0/59, loss 1.932, top-1 accuracy 36.000, top-5 accuracy 84.000
val part1: loss 1.269640
Checkpoint saved
part1 Epoch 89 / 100
train part1: batch 0/29, loss 0.822, top-1 accuracy 70.000, top-5 accuracy 98.000
```

```
train part1: loss 0.837765
val part1: batch 0/59, loss 2.063, top-1 accuracy 28.000, top-5 accuracy 82.000
val part1: loss 1.269923
Checkpoint saved
part1 Epoch 90 / 100
train part1: batch 0/29, loss 0.580, top-1 accuracy 82.000, top-5 accuracy 98.000
train part1: loss 0.836578
val part1: batch 0/59, loss 2.060, top-1 accuracy 32.000, top-5 accuracy 84.000
val part1: loss 1.273393
Checkpoint saved
part1 Epoch 91 / 100
train part1: batch 0/29, loss 0.825, top-1 accuracy 72.000, top-5 accuracy 100.000
train part1: loss 0.827001
val part1: batch 0/59, loss 1.928, top-1 accuracy 32.000, top-5 accuracy 84.000
val part1: loss 1.276233
Checkpoint saved
part1 Epoch 92 / 100
train part1: batch 0/29, loss 0.602, top-1 accuracy 76.000, top-5 accuracy 98.000
train part1: loss 0.802104
val part1: batch 0/59, loss 1.882, top-1 accuracy 34.000, top-5 accuracy 84.000
val part1: loss 1.284780
Checkpoint saved
part1 Epoch 93 / 100
train part1: batch 0/29, loss 0.725, top-1 accuracy 72.000, top-5 accuracy 96.000
train part1: loss 0.828365
val part1: batch 0/59, loss 2.085, top-1 accuracy 28.000, top-5 accuracy 84.000
val part1: loss 1.258868
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part1 Epoch 94 / 100
train part1: batch 0/29, loss 0.428, top-1 accuracy 84.000, top-5 accuracy 100.000
train part1: loss 0.821419
val part1: batch 0/59, loss 2.047, top-1 accuracy 28.000, top-5 accuracy 84.000
val part1: loss 1.267059
Checkpoint saved
part1 Epoch 95 / 100
train part1: batch 0/29, loss 0.722, top-1 accuracy 66.000, top-5 accuracy 100.000
train part1: loss 0.835623
val part1: batch 0/59, loss 1.983, top-1 accuracy 32.000, top-5 accuracy 84.000
val part1: loss 1.281335
Checkpoint saved
part1 Epoch 96 / 100
train part1: batch 0/29, loss 0.756, top-1 accuracy 72.000, top-5 accuracy 100.000
train part1: loss 0.833581
val part1: batch 0/59, loss 1.998, top-1 accuracy 30.000, top-5 accuracy 82.000
val part1: loss 1.269315
Checkpoint saved
part1 Epoch 97 / 100
train part1: batch 0/29, loss 0.706, top-1 accuracy 82.000, top-5 accuracy 98.000
train part1: loss 0.780962
val part1: batch 0/59, loss 2.074, top-1 accuracy 28.000, top-5 accuracy 80.000
val part1: loss 1.280181
Checkpoint saved
part1 Epoch 98 / 100
train part1: batch 0/29, loss 0.821, top-1 accuracy 68.000, top-5 accuracy 98.000
train part1: loss 0.785870
val part1: batch 0/59, loss 1.968, top-1 accuracy 36.000, top-5 accuracy 84.000
val part1: loss 1.281831
Checkpoint saved
part1 Epoch 99 / 100
train part1: batch 0/29, loss 0.744, top-1 accuracy 80.000, top-5 accuracy 96.000
train part1: loss 0.811295
val part1: batch 0/59, loss 2.005, top-1 accuracy 24.000, top-5 accuracy 84.000
val part1: loss 1.286770
Checkpoint saved
Best top-1 Accuracy = 60.972
```

Make sure you get at least 50% accuracy in this section! If you tried different settings than the ones provided to get 50%, you should modify custom_part1_trainer in student code to return a dictionary with your changed settings.

Part 2. Fine-Tuning a Pre-Trained Network

In [16]:
```
# Fix random seeds so that results will be reproducible
set_seed(0, use_GPU)
```

Training a network from scratch takes a lof of time. Instead of training from scratch, we can take a pre-trained model and fine tune it for our purposes. This is the goal of Part 2--you will train a pre-trained network, and achieve at least 80% accuracy.

In [17]:
```
# training parameters
input_size = (224, 224)
RGB = True
base_lr = 1e-3
weight_decay = 5e-4
momentum = 0.9
backprop_depth = 3
```

In [18]:
```
# Create the training and testing datasets.
train_dataset, test_dataset = sc.create_datasets(data_path=data_path, input_size=input_size, rgb=RGB)
assert test_dataset.classes == train_dataset.classes
```

```
Computing pixel mean and stdev...
Batch 0 / 30
Batch 20 / 30
Done, mean =
[0.45586014 0.45586014 0.45586014]
std =
[0.24787705 0.24787705 0.24787705]
Computing pixel mean and stdev...
Batch 0 / 60
Batch 20 / 60
Batch 40 / 60
Done, mean =
[0.45524448 0.45524448 0.45524448]
std =
[0.24699316 0.24699316 0.24699316]
```

Following block loads a pretrained AlexNet.

In [19]:
```
from torchvision.models import alexnet
```

In [20]:
```
# Create the network model.
model = alexnet(pretrained=True)
print(model)
```

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace)
    (3): Dropout(p=0.5)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

Now, you modify create_part2_model from student code in order to fine-tune AlexNet. As you can see in the docs (https://github.com/pytorch/vision/blob/master/torchvision/models/alexnet.py (https://github.com/pytorch/vision/blob/master/torchvision/models/alexnet.py)) and in the model printout above, AlexNet has 2 parts: 'features', which constists of conv layers that extract feature maps from the image, and 'classifier' which consists of FC layers that classify the features. We want to replace the last Linear layer in model.classifier.

In [21]:
```
model = sc.create_part2_model(model, num_classes)
if use_GPU:
    model = model.cuda()
print(model)
```

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace)
    (3): Dropout(p=0.5)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace)
    (6): Linear(in_features=4096, out_features=15, bias=True)
  )
)
```

Next we will create the loss function and the optimizer. Just as with part 1, if you modify any of the setttings to hit the required accuracy, you must modify custom_part2_trainer function to return a dictionary containing your changes.

In [22]:
```
# Set up the trainer. You can modify custom_part2_trainer in
# student_copy.py if you want to try different learning settings.
custom_part2_trainer = sc.custom_part2_trainer(model)

if custom_part2_trainer is None:
    # Create the loss function
    # see http://pytorch.org/docs/0.3.0/nn.html#loss-functions for a list of available loss functions
    loss_function = nn.CrossEntropyLoss()

    # Since we do not want to optimize the whole network, we must extract a list of parameters of interest that will be
    # optimized by the optimizer.
    params_to_optimize = []

    # List of modules in the network
    mods = list(model.features.children()) + list(model.classifier.children())

    # Extract parameters from the last `backprop_depth` modules in the network and collect them in
    # the params_to_optimize list.
    for m in mods[::-1][:backprop_depth]:
        params_to_optimize.extend(list(m.parameters()))

    # Construct the optimizer
    optimizer = optim.SGD(params=params_to_optimize, lr=base_lr, weight_decay=weight_decay, momentum=momentum)

    # Create a scheduler, currently a simple step scheduler, but you can get creative.
    # See http://pytorch.org/docs/0.3.0/optim.html#how-to-adjust-learning-rate for various LR schedulers
    # and how to use them
    lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.1)

    params = {'n_epochs': 4, 'batch_size': 10, 'experiment': 'part2'}

else:
    if 'loss_function' in custom_part2_trainer:
        loss_function = custom_part2_trainer['loss_function']
    if 'optimizer' in custom_part2_trainer:
        optimizer = custom_part2_trainer['optimizer']
    if 'lr_scheduler' in custom_part2_trainer:
        lr_scheduler = custom_part2_trainer['lr_scheduler']
    if 'params' in custom_part2_trainer:
        params = custom_part2_trainer['params']
```

We are ready to fine tune our network! Just like before, we will start a local server to see the training progress of our network. Open a new terminal and activate the environment for this project. Then run the following command: **python -m visdom.server**. This will start a local server. The terminal output should give out a link like: "http://localhost:8097 (http://localhost:8097)". Open this link in your browser. After you run the following block, visit this link again, and you will be able to see graphs showing the progress of your training! If you do not see any graphs, select Part 2 on the top left bar where is says Environment (only select Part 2, do not check main or Part 1).

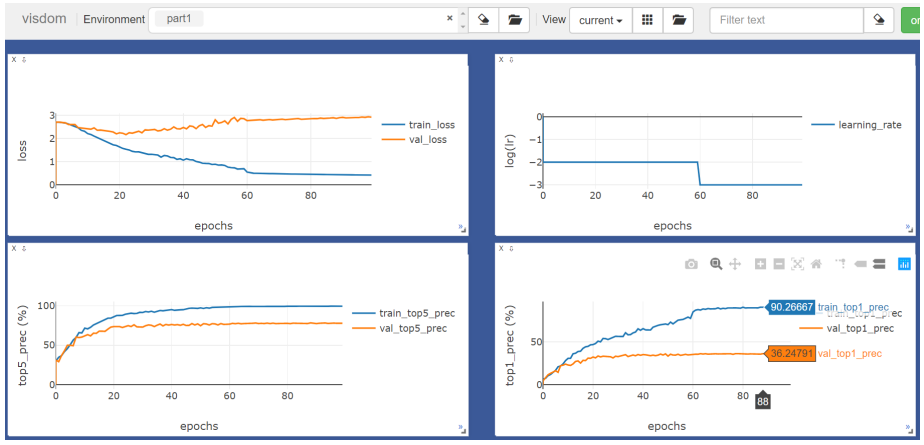
In [23]:
```
# Train the network!
trainer = Trainer(train_dataset, test_dataset, model, loss_function, optimizer, lr_scheduler, params)
best_prec1 = trainer.train_val()
print('Best top-1 Accuracy = {:4.3f}'.format(best_prec1))
```

```
WARNING:root:Setting up a new session...

---------------------------------------
Experiment: part2
n_epochs: 4
batch_size: 10
do_val: True
shuffle: True
num_workers: 4
val_freq: 1
print_freq: 100
experiment: part2
checkpoint_file: None
resume_optim: True
---------------------------------------
part2 Epoch 0 / 4
train part2: batch 0/149, loss 3.266, top-1 accuracy 10.000, top-5 accuracy 40.000
train part2: batch 100/149, loss 1.323, top-1 accuracy 60.000, top-5 accuracy 100.000
train part2: loss 1.062432
val part2: batch 0/298, loss 0.674, top-1 accuracy 70.000, top-5 accuracy 100.000
val part2: batch 100/298, loss 0.444, top-1 accuracy 90.000, top-5 accuracy 100.000
val part2: batch 200/298, loss 0.327, top-1 accuracy 90.000, top-5 accuracy 100.000
val part2: loss 0.481692
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
part2 Epoch 1 / 4
train part2: batch 0/149, loss 0.071, top-1 accuracy 100.000, top-5 accuracy 100.000
train part2: batch 100/149, loss 0.487, top-1 accuracy 80.000, top-5 accuracy 100.000
train part2: loss 0.503408
val part2: batch 0/298, loss 1.432, top-1 accuracy 60.000, top-5 accuracy 100.000
val part2: batch 100/298, loss 0.469, top-1 accuracy 90.000, top-5 accuracy 100.000
val part2: batch 200/298, loss 0.523, top-1 accuracy 80.000, top-5 accuracy 100.000
val part2: loss 0.507896
Checkpoint saved
part2 Epoch 2 / 4
train part2: batch 0/149, loss 0.105, top-1 accuracy 90.000, top-5 accuracy 100.000
train part2: batch 100/149, loss 0.145, top-1 accuracy 90.000, top-5 accuracy 100.000
train part2: loss 0.354122
val part2: batch 0/298, loss 0.938, top-1 accuracy 80.000, top-5 accuracy 90.000
val part2: batch 100/298, loss 0.289, top-1 accuracy 90.000, top-5 accuracy 100.000
val part2: batch 200/298, loss 0.585, top-1 accuracy 80.000, top-5 accuracy 100.000
val part2: loss 0.528443
Checkpoint saved
part2 Epoch 3 / 4
train part2: batch 0/149, loss 0.061, top-1 accuracy 100.000, top-5 accuracy 100.000
train part2: batch 100/149, loss 0.499, top-1 accuracy 70.000, top-5 accuracy 100.000
train part2: loss 0.269003
val part2: batch 0/298, loss 0.472, top-1 accuracy 70.000, top-5 accuracy 100.000
val part2: batch 100/298, loss 0.529, top-1 accuracy 90.000, top-5 accuracy 100.000
val part2: batch 200/298, loss 0.592, top-1 accuracy 80.000, top-5 accuracy 100.000
val part2: loss 0.491897
Checkpoint saved
BEST TOP1 ACCURACY SO FAR
Best top-1 Accuracy = 84.590
```

Expect this code to take around 10 minutes on CPU or 30 seconds on GPU. You should hit 80% accuracy.

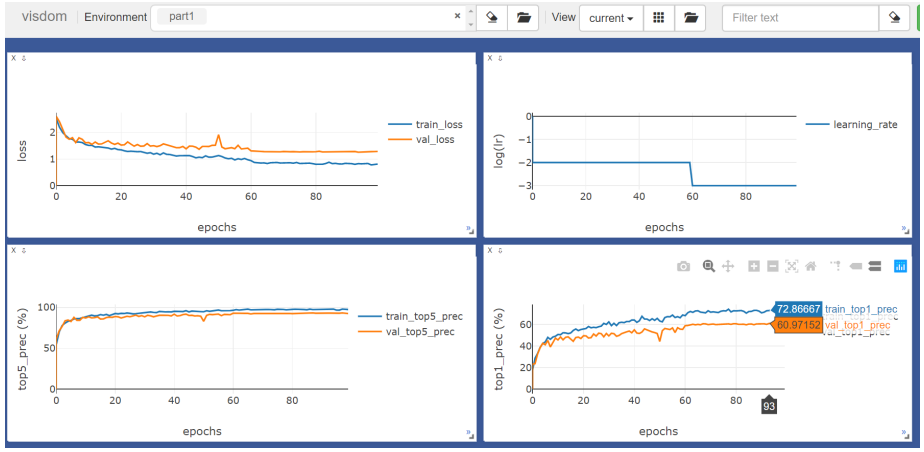
Write Up

Before jittering and only using a simple neural net

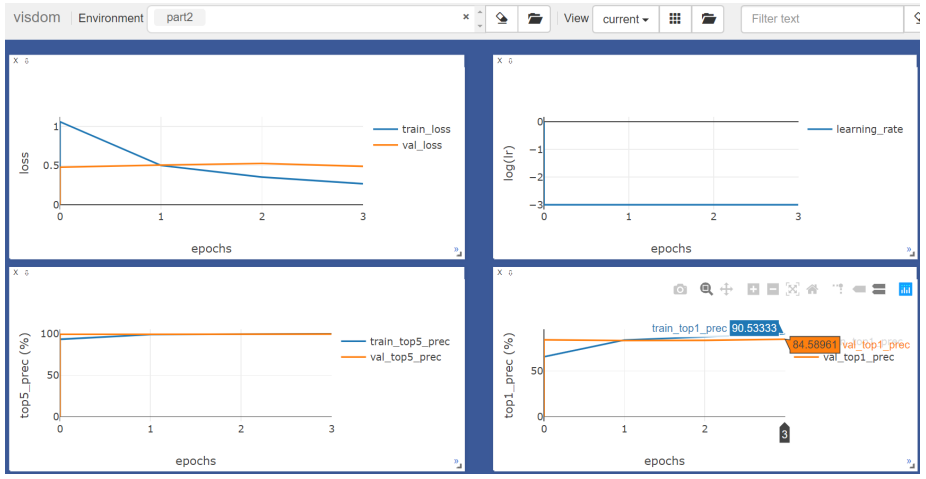


After jittering and using more CNN, BatchNorm, Dropout layers

Deeper network also better feature extractions and dropout minimize overfitting



Transfer Learning with Alexnet



Part 1

How did you do color jittering and nomalization? Why are they helpful?

How did you build your model? (what are sequential network, feature extractor, and classifier?)

Talk more about the operations (Conv2d, Maxpool, Relu etc.) you used. Explain why they are important for deep model.

How did you do data transformation? Also, disucss anything else you need to do to achieve require performance.

How did you do color jittering and nomalization? Why are they helpful?

I do the jittering by `train_data_tforms.append(transforms.RandomHorizontalFlip(p=0.5))` to randomly flip the image. When a kitchen image is flipped, it is still a kitchen. Jitering helps to create variety in the training dataset.

I do normalization by `train_data_tforms.append(transforms.Normalize(mean = train_mean, std = train_std))` . Normalization helps the model to converge in a more stable way as the boundary becomes less sensitive to weight variation.

How did you build your model? (what are sequential network, feature extractor, and classifier?)

Sequential Network is a set of NN layers connected in series. Typically, we have an activation layer after a linear or CNN layer.

Feature extractor for image is typically a set of CNN layers to extract feature with low-/mid-/high-level of abstraction. Trainning an effective feature extractor demands a very large dataset and deep neural network.

Classifier is to match the feature to the class (confident score) that we want to predict. Typically, it is a fully connected layer.

Talk more about the operations (Conv2d, Maxpool, Relu etc.) you used. Explain why they are important for deep model.

Conv2d is to learn the feature extraction parameters and to extract the parameters at the same time. It is a 3D version of the filter operation.

Maxpool is to down-sample the image and sometimes help with noise removal.

Relu is an activation layer to create non-linearity. Previously, sigmoid function is intensively employed, but with the invention of Relu function, we can train the deep net very stable.

BatchNormalization is to normalize the layer with mean and std. Also a great invention to stablize the trainning of deep net.

Dropout is to minimize over-fitting. A simple but very effective invention from Prof. Hinton's group.

How did you do data transformation? Also, disucss anything else you need to do to achieve require performance.

From an image path in a image folder, utils.py load the image and calculate the mean and std (critical step to stablize the training)

Pytorch `transform.Compose([])` do the following:

- Resize and Center crop such that all the image has the same input dimension (very critical)
- Transform to GrayScale if we want 1 channel image input
- Randomly flip horizonal the image to create a variety in the training data
- Change datatype to `Tensor` to perform operation on Pytorch
- Normalize the image with mean, std to zero-centered all input

Now the data is ready for training.

In [ ]:

Part 2. Fine-Tuning a Pre-Trained Network

What is the fine-tuning? What is the benifit of fine-tuing?

Fine-tuning is use the pretrained network to do transfer-learning. It use the learned weights from a pre-trained network on a very large dataset as a reference.

Typical, only the last few layers of the pre-trained network allows weight-updates.

The feature extraction works on the large dataset and its learned knowledge can be apply to current small dataset. Only the last layer need to be updated to fit with current class prediction.

In [ ]:

Extra Credits

Saliency map:

Visualize which part of an image is important?

In [24]:
```
train_mean = [0.45524448, 0.45524448, 0.45524448]
train_std =  [0.24699316, 0.24699316, 0.24699316]

detransform = transforms.Compose([
    transforms.Normalize(mean = (0,0,0), std = [1/s for s in train_std]),
    transforms.Normalize(mean = [-s for s in train_mean], std = (1,1,1)),
])
```

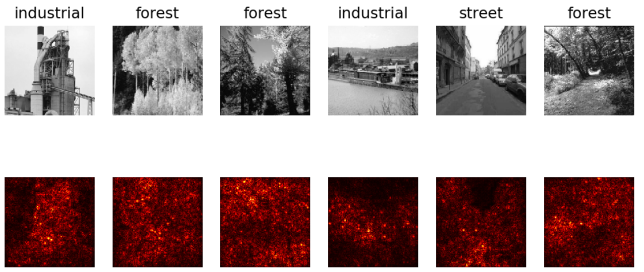
In [26]:
```
model.eval();
```

In [28]:
```
# Sample 100 images from the train dataset

indexes = np.random.permutation(len(train_dataset))[:100]
X,y,name = [],[],[]
for idx in indexes:
    img, label = train_dataset[idx][0],train_dataset[idx][1]
    X.append(img), y.append(label), name.append(train_dataset.classes[label])
X = torch.stack(X); y = torch.LongTensor(y); name = np.array(name)
```

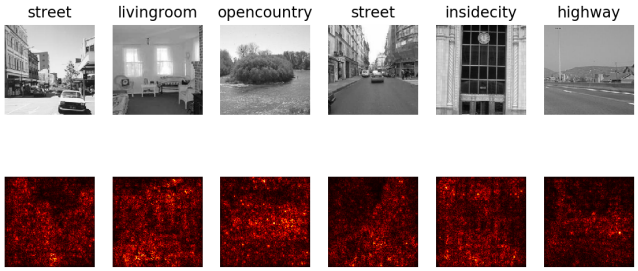
In [29]:
```
X.requires_grad_(True)
alex_cal = model(X)
score = torch.gather(alex_cal, 1, y.view(-1,1)).squeeze()
score.backward(torch.ones_like(score))

saliency = X.grad.clone().abs()
saliency, _ = torch.max(saliency, dim = 1)
saliency = saliency.numpy()
```

In [34]:
```
def plot_saliency_map(X, saliency, name, N=6, figsize = (9,4)):
    plt.figure(figsize=figsize)
    for ii in range(N):
        plt.subplot(2,N, ii+1); plt.title(name[ii])
        img = detransform(X.data[ii].clone()).numpy().transpose(1,2,0)
        plt.imshow(img/np.max(img)); plt.axis('off')

        plt.subplot(2,N, N+ii+1);
        plt.imshow(saliency[ii], cmap=plt.cm.hot); plt.axis('off')
    plt.show()
```

In [35]:
```
shuffle = np.random.permutation(len(X))[:6]
plot_saliency_map(X[shuffle], saliency[shuffle], name[shuffle])
```



In [37]:
```
shuffle = np.random.permutation(len(X))[:6]
plot_saliency_map(X[shuffle], saliency[shuffle], name[shuffle])
```



Visualize the feature extraction by CNN layers of Alexnet

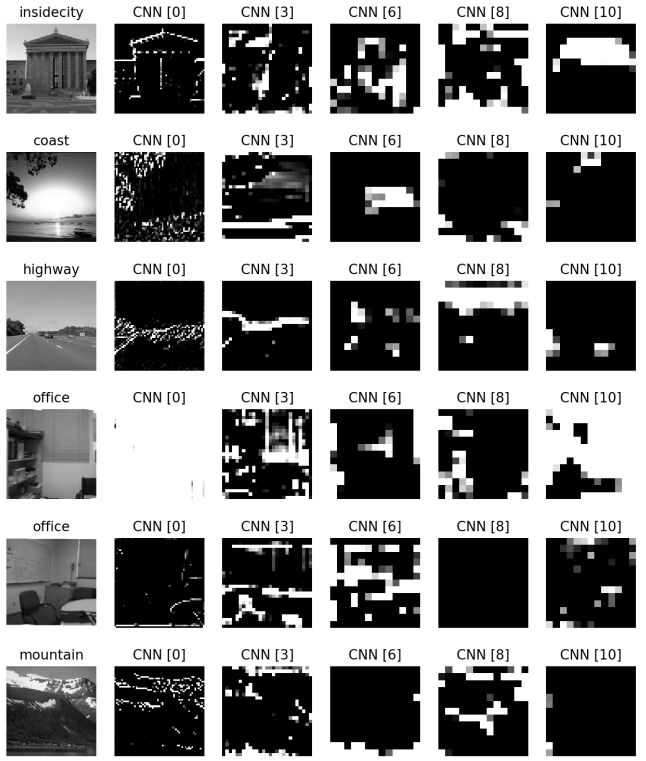
See the output of an image after passing over various CNN layers of Alexnet.

It can be seen very clearly that higher layers has higher abstraction level on the images information

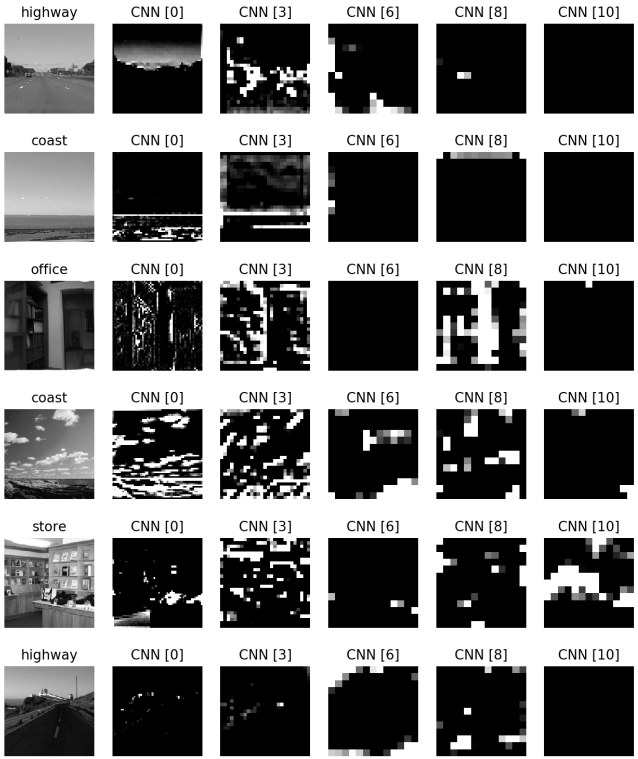
In [49]:
```
# Sample N images from the train dataset
indexes = np.random.permutation(len(train_dataset))[:100]
X,y,name = [],[],[]
for idx in indexes:
    img, label = train_dataset[idx][0],train_dataset[idx][1]
    X.append(img), y.append(label), name.append(train_dataset.classes[label])
X = torch.stack(X); y = torch.LongTensor(y); name = np.array(name)
```

In [51]:
```
def visualize_AlexNet_layers(N =6):
    shuffle = np.random.permutation(len(X))[:N]
    img, named_classes = X[shuffle].clone(), name[shuffle]

    # passing the images to Alex-net and record images at CNN layers
    # conv_layers are the CNN layers of Alex-net

    conv_layers = ['0','3','6','8','10']
    alex_cal = [img]
    for key, layer in model.features._modules.items():
        img = layer(img)
        if key in conv_layers:
            alex_cal.append(img)

    # Visualize
    plt.figure(figsize=(10,N*2))
    for row in range(N):
        visual = [img[row] for img in alex_cal]
        for ii in range(len(visual)):
            plt.subplot(N,6,row*6+ii+1)
            img_tensor = detransform(visual[ii].clone()) if ii == 0 else visual[ii].clone().squeeze(0)

            #Randomly sample only one CNN feature at a given layer
            img_np = img_tensor[np.random.choice(len(img_tensor))].numpy()
            plt.imshow(np.clip(img_np,0,1),cmap='gray'); plt.axis('off')
            plt.title(named_classes[row] if ii == 0 else 'CNN [%s]' %(conv_layers[ii-1]))
    plt.show()
```

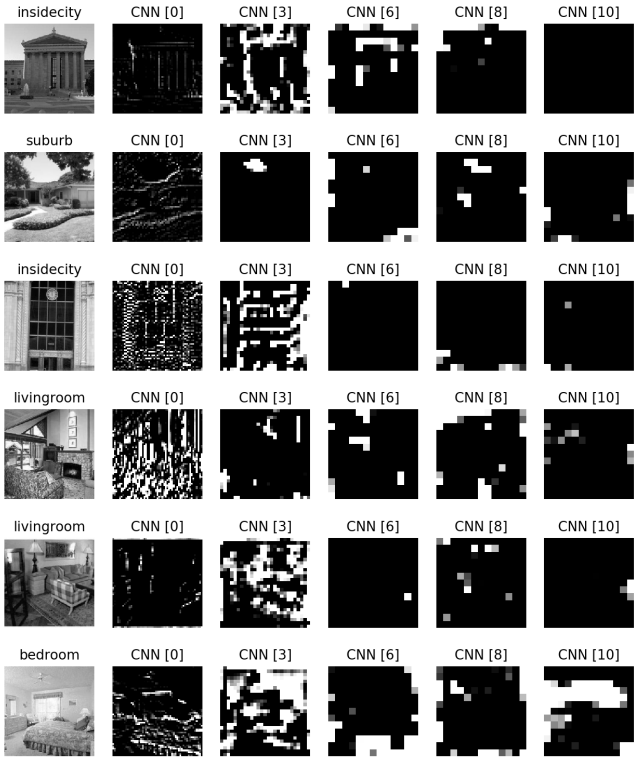
In [52]:
```
visualize_AlexNet_layers(N =6)
```



In [53]:
```
visualize_AlexNet_layers(N =6)
```



In [54]:
```
visualize_AlexNet_layers(N =6)
```



In [ ]: