High Level Computer Vision

Project Introduction | SS 2018

11/06/2018   - Rakshith Shetty
Logistics

- Start date: 11. 06
- Proposal presentation (5+5min) on 18. 06: (starts at 13:00)
  ‣ 5 slides due 17.06 – task / motivation / method / dataset

- Interim presentation (5+5min) on 02. 07: (starts at 13:00)
  ‣ Slides due on 01.07 - progress report / problems encountered / feedback

- Final presentation (15+5min) on 23. 07 *
  ‣ Slides due on 22.07 - Progress and presentation evaluation

- Written report submitted on 27. 07 (23:59)
  ‣ Report evaluation
Choose a dataset and task:

- **Datasets**: Caltech4, Caltech101, Buffy Stickmen, HOI, UKBench, MPII Human Pose, ImageNet, COCO, CelebA etc.
- **Tasks**: object class recognition, object detection/localization, person identification, gender recognition, scene classification, image captioning, visual question answering, image generation etc.

Apply methods to the task, present the analysis of your results

- Necessary simplifications are OK (e.g. additional annotations)
- Can you think of a new twist to the method?
Project goal

● Application:
  ○ Apply computer vision techniques to a real-world problem.

● Model
  ○ Build a new model/algorithm or a new variant of existing models for an existing computer vision task.

● Apply your methods to the task, present the analysis of your results.
Proposal Slides Structure

● Slide 1/2 – Task and motivation
  ○ Task statement and definitions
  ○ Motivation
  ○ Related work

● Slide 3/4 – Models, tools, novelty
  ○ Tentative material and methods
    ■ From coursework, open source and research code
  ○ Investigation
    ■ Feature, model, etc.

● Slide 5 – Analysis
  ○ Benchmark
    ■ Evaluation dataset and metric
  ○ Baselines
● Title
● Abstract
● Introduction
● Related work
● Proposed method explained
● Experimental results
● Conclusions and Future work
● References
● Reports to indicate assignments of each group member
● **Honor Code:** clearly cite your sources in your code and your report.
Conferences

● CVPR
● ICCV
● ECCV
● NIPS
● ICLR
● BMVC
● ACCV
● GCPR
Dataset

- Caltech-101
  - object class recognition
  - object localization
Dataset

● Pascal VOC:
  ○ Object detection
  ○ Segmentation
Dataset

  - Object instance retrieval
  - Object category classification
Dataset

- RGB-D Indoor Scenes Dataset ([http://cs.nyu.edu/~silberman/datasets/](http://cs.nyu.edu/~silberman/datasets/))
  - Scene classification
  - Object detection, recognition, segmentation
Dataset

- Leeds Sports Pose Dataset
  - sport scene recognition
Dataset

- ImageNet
- SUN Database
- Places Database
- MPII Human Pose
- Microsoft COCO
- Labeled Faces in the Wild
- ....
Dataset

● Or capture your own
  ○ Digital camera, mobile phone, Google glasses..
  ○ Microsoft Kinect
  ○ Web / Google image search
Methods

- **Bag-of-Words (BoW)**
  - Tools: VLFeat, SVM, Random Forests

- **Relevant paper**

- **Publicly available SVM implementations**
  - http://svmlight.joachims.org/
  - http://www.csie.ntu.edu.tw/~cjlin/liblinear/
Methods

● Interest points, vocabulary trees and geometry
  ○ Tools:
    ■ Hessian, Harris detectors
    ■ SIFT descriptor
    ■ Vocabulary tree

● Relevant paper
Methods

- **Image (IS) and video segmentation (VS)**
  - **Tools:**
    - IS ([http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html](http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html))
  - **Papers**
Methods

- **Convolutional Neural Networks**
  - **Tools:**
    - PyTorch: [https://pytorch.org/](https://pytorch.org/)
    - Tensorflow: [https://www.tensorflow.org/](https://www.tensorflow.org/)
  - **Papers**
    - .....
Methods

● Image Captioning
  ○ Tools:
    ■ Show and Tell: https://github.com/tensorflow/models/tree/master/research/im2txt
    ■ Neuraltalk2: https://github.com/ruotianluo/ImageCaptioning.pytorch
  ○ Papers
    ■ .....
For a good project

- **5 W’s**
  - What? (a problem)
  - Why? (motivation)
  - How? (proposed strategy)
  - Where? (dataset and benchmark)
  - Who? (team assignments)

- It is recommended
  - Baseline

- It is desired.. your considerations on
  - Influence of parameter and dataset choice
  - Results: what is expected and what is surprising.. not just numbers!
  - Observations must be substantiated by results or references
Example Projects

- Gender Recognition
- Age Recognition
- other recognition tasks....
- Object recognition or detection with the Kinect (RGB + Depth)
- Image retrieval for 3D objects
- Object retrieval in videos, on a mobile phone
- Person identification
- Image and video segmentation
- Detection and segmentation
- Tracking
- Vision and Language tasks (captioning, question answering, explanations)
- Image generation tasks (GANs, conditional generation, style transfer)
Previous year projects
Conditional Face Generation

- Generate faces of people conditioned on facial expression
- Generative adversarial networks
- Based on an BEGAN - added conditioning and good experiments and evaluation to test the model

Happy

![Happy Faces](image)

Angry

![Angry Faces](image)
TV series classification

- Classify tv series from short smartphone videos
- CNN frame level classification (designed based on related work)
- Collected own dataset !!
- Synthetic data augmentation
Visual question answering

- Answer simple questions about an image
- Novel extension to prior work to predict location of the object
- Combined the segmentation annotations with answers to obtain gt.

(a) Q: What is on the left side of sink?
1. cup [17.19%]
2. person [16.27%]
3. toilet [15.82%]
GT: cup

(d) Q: What is on the left side of laptop?
1. couch [70.49%]
2. person [14.31%]
3. bed [4.80%]
GT: couch
What am I working on?
Image Captioning

- Generate one sentence description of an input image or video

![Diagram of Image Captioning process]

- Input Image or Video
- Feature Extraction
- LSTM Generator
- Final Caption

Examples:
- C27: a man and a dog herding sheep in a field
- a bathroom with a sink, toilet, and bathtub
- a bottle of wine and a glass of wine
Image Captioning - Using Adversarial Training

➢ Use adversarial training
➢ Discriminator designed to promote diversity
➢ Adversarially trained model
  ○ Better diversity both per image and across images
  ○ Better match word usage statistics
  ○ Accuracy is on par with baseline

[Diagram showing Image Input, Generator, Set of Captions, Discriminator, Classify as real or fake, Backpropagate]
Adv-samp

a bathroom with a walk in shower and a sink

da dirty bathroom with a broken toilet and sink

a view of a very nice looking rest room

a white toilet in a public restroom stall

a small bathroom has a broken toilet and a broken sink

Base-samp

a bathroom with a toilet and a sink

Adv-samp

a person on skis jumping over a ramp

a skier is making a turn on a course

a cross country skier makes his way through the snow

a skier is headed down a steep slope

a person cross country skiing on a trail

Base-samp

a man riding skis down a snow covered slope
Interaction Free Image Manipulation

“Add a **stop-sign**, **skaters crossing the street** and change the color of the **bus to green**”

We tackle one aspect of this problem → **Automatic object removal**
Problem Formulation

**Input class label:**
- TV,
- Person,
- Chair

**Learn from unpaired data**

**Target class label:**
- Person,
- Chair

**Not available to train**
Overall Approach

- Two-staged editor network with a mask-generator ($G_M$) and an in-painter ($G_I$).
- $G_M$ fools the object classifier. $G_I$ fools the real-fake classifier.

$$L_{cls}(G_M) = -\mathbb{E}_x [\log(1 - D_{cls}(y, c_t))]$$
$$L_{rf}(G_I) = -\mathbb{E}_x [D_{rf}(y)]$$

- Still under-constrained and needs priors.

Two-stage generator avoids adversarial patterns
Mask Priors

- Object shapes are usually continuous coherent blobs.
  - Encourage generated masks to match a prior mask distribution to learn this.

- Wasserstein GAN $P(m^p|c_t)$ rated masks to match class ($c_t$) specific prior
  $$L_{prior}(G_M) = -\mathbb{E}_x [D_M(G_M(x, c_t), c_t)]$$

- Just need samples from the prior.
  - Can be random rectangular maks
  - Or class specific segmentation masks from a different dataset (unpaired)
Qualitative Results - COCO Dataset

Input image

M-RCCN based

Ours

dog  person  tv  airplane  person  cow  person
Qualitative Results - Logo Removal

Input image

Ours

mini  pepsi  heineken

Input image

Ours

nbc  citroen  bmw
High Level Computer Vision

PyTorch - Quick Introduction

Rakshith Shetty - 11/06/2017

Some slides borrowed from:
What is it?

Tensors and Dynamic neural networks in Python with strong GPU acceleration.

PyTorch is a deep learning framework that puts Python first.

We are in an early-release Beta. Expect some adventures.

Learn More
What is it?

- A library that allows tensor based computation (like matlab/ numpy)
  - Easily run on GPU or CPU.
  - **Do automatic differentiation! Very useful for backpropagation**
  - One of the fastest (maybe caffe is a bit faster)
  - Several library functions which allows you to quickly

- What’s different to other platforms?
  - **Dynamic computational graphs**
  - Very useful when dealing with recurrent networks or other wacky architectures

```python
from torch.autograd import Variable
x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```
Basics

```python
import numpy as np
import torch

# Task: compute matrix multiplication C = AB

# using numpy
A = np.random.rand(d, d).astype(np.float32)
B = np.random.rand(d, d).astype(np.float32)
C = A.dot(B)

# using torch with gpu
A = torch.rand(d, d).cuda()
B = torch.rand(d, d).cuda()
C = torch.mm(A, B)
```

350 ms

0.1 ms
Auto-differentiate

```python
import torch
from torch.autograd import Variable

# Task: compute d(||x||^2)/dx
x = Variable(torch.range(1, 5), requires_grad=True)
print(x.data)  # x.data = [1, 2, 3, 4, 5]

f = x.dot(x)
print(f.data)  # f.data = 55

f.backward()
print(x.grad)  # x.grad = [2, 4, 6, 8, 10]
```
Auto-differentiate

```python
import torch
from torch.autograd import Variable

# Task: compute d(||x||^2)/dx
x = Variable(torch.range(1, 5), requires_grad=True)
print(x.data)  # x.data = [1, 2, 3, 4, 5]

f = x.dot(x)
print(f.data)  # f.data = 55

f.backward()
print(x.grad)  # x.grad = [2, 4, 6, 8, 10]
```
# Components

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>torch</td>
<td>a Tensor library like NumPy, with strong GPU support</td>
</tr>
<tr>
<td>torch.autograd</td>
<td>a tape based automatic differentiation library that supports all differentiable Tensor operations in torch</td>
</tr>
<tr>
<td>torch.nn</td>
<td>a neural networks library deeply integrated with autograd designed for maximum flexibility</td>
</tr>
<tr>
<td>torch.optim</td>
<td>an optimization package to be used with torch.nn with standard optimization methods such as SGD, RMSProp, LBFGS, Adam etc.</td>
</tr>
<tr>
<td>torch.multiprocessing</td>
<td>python multiprocessing, but with magical memory sharing of torch Tensors across processes. Useful for data loading and hogwild training.</td>
</tr>
<tr>
<td>torch.utils</td>
<td>DataLoader, Trainer and other utility functions for convenience</td>
</tr>
</tbody>
</table>
Torchvision

- Consists of popular datasets, model architectures, and common image transformations.

- Imagenet pretrained models are available.

```python
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
```
```python
import torch.utils.data
import torch.nn as nn
from torch.autograd import Variable
from torch import tensor

class MLP_classifier(nn.Module):
    def __init__(self, params):
        super(MLP_classifier, self).__init__()
        # 1 is to allow pushing index
        self.output_size = params.out['num_output_layers'], 205
        self.hidden_dims = params.get('hidden_widths', [])
        self.rnn_size = params.get('rnn_size', 1)

        prev_size = self.rnn_size
        self.hidden_dims.append(self.output_size)
        self.rnn_layers = nn.ModuleList()
        self.hidden_layers = nn.ModuleList()
        self.rnn_dropout = nn.Dropout(p=params.get('drop_prob', 0.25))
        prev_size = self.hidden_dims[1]

        self.softmax = nn.LogSoftmax()
        self.init_weights()

    def init_weights(self):
        # Weight initializations for various parts.
        a = 0.0
        # LSTM forget gate could be initialized to high value (1.)
        for i in range(len(self.hidden_dims)):
            self.hidden_layers[i].weight.data.uniform_(-a, a)
            self.hidden_layers[i].bias.data.fill_(0)

    def forward(self, x, compute_softmax = False):
        x = Variable(x).cuda()
        prev_out = x

        for i in range(len(self.hidden_dims) - 1):
            prev_out = self.rnn_dropout[i](prev_out)
            prev_out = self.rnn_layers[i](prev_out)
            prev_out = self.rnn_dropout[i](prev_out)

        if compute_softmax:
            prob_out = self.softmax(prev_out)
        else:
            prob_out = prev_out

        return prob_out
```
Useful resources

● Official documentation
  ○ [http://pytorch.org/docs/](http://pytorch.org/docs/)

● Tutorials
  ○ [http://pytorch.org/tutorials/](http://pytorch.org/tutorials/)
  ○ [https://github.com/pytorch/tutorials](https://github.com/pytorch/tutorials)
  ○ [http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html](http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html) (Useful)

● Example projects
  ○ [https://github.com/pytorch/examples](https://github.com/pytorch/examples)