Self-labelling via Simultaneous Clustering and Representation Learning

Seminar on Current Works in Computer Vision

Presenter: Saurav Shanu

Authors
Asano YM.
Rupprecht C.
Vedaldi A.
Table of Contents

- Introduction
- Related Work
- Method
- Experiments
- Summary
Unsupervised learning

Self labelling

Unlabelled Dataset → Deep Network → Pseudo label 1, Pseudo label 2, Pseudo label 3, Pseudo label 4

Discovering labels (Simultaneous Clustering) + Learning features (Representational Learning)
Steps

Train CNN

Update the labels

Minimize cross-entropy loss
DeepCluster

- Neural network + k-means clustering.
- Trivial solution.
- Ad-hoc tricks for regularization.

Caron et al. 2019
Related Work – Self supervised learning

- Pathak et al. CVPR 2016
- Jenni and Favaro CVPR 2018
- Zhang et al. ECCV 2016
- Doersch et al. ICCV 2015
- Dosovitskiy et al. PAMI 2016
Table of Contents

- Introduction
- Related Work
- Method
- Experiments
- Summary
Classification with supervised learning

Minimizing cross-entropy loss.

\[ E(p|y_1, \ldots, y_N) = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i|x_i). \]
How to do in Unsupervised learning

- What’s the best way to minimize the loss?

Image source: towardsdatascience.com
Change the loss function?

- Introduce a distribution “q” of labels over the data:

  \[ E(p|y_1, \ldots, y_N) = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i|x_i) \]

  \[ E(p, q) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{y=1}^{K} q(y|x_i) \log p(y|x_i) \]

- Goal \( \rightarrow \) Find the label assignment distribution “q”.
Avoid degeneracy

- Add constraints on q:

\[
\min_{p,q} E(p, q) \quad \text{subject to} \quad \forall y : q(y|x_i) \in \{0, 1\} \quad \text{and} \quad \sum_{i=1}^{N} q(y|x_i) = \frac{N}{K}
\]

Each datapoint is assigned to exactly one label.

Equipartition data.
Optimal transport problem

Kantorovich Problem

Optimal transport course by Cuturi
Optimal transport problem

Kantorovich Problem

Easy solution: split the task with proportions
120:90:90 = 4:3:3

Optimal transport course by Cuturi
Optimal transport problem

Kantorovich Problem

Easy solution: split the task with proportions
120:90:90 = 4:3:3

Optimal transport course by Cuturi
Optimal transport problem

Kantorovich Problem

Easy solution: split the task with proportions 120:90:90 = 4:3:3

Optimal transport course by Cuturi
Optimal transport problem

Kantorovich Problem

Easy solution: split the task with proportions
120:90:90 = 4:3:3

Optimal transport course by Cuturi
Kantorovich Problem

Easy solution: split the task with proportions
120:90:90 = 4:3:3
Optimal transport problem

Kantorovich Problem

Naive approach results in too many displacements.

Goal: find a cheaper alternative

Easy solution: split the task with proportions
120:90:90 = 4:3:3
Optimal transport problem

Kantorovich Problem

Optimal transport course by Cuturi
Application to label assignment

Cost function (class probabilities)
Optimal transport problem - Formally

Formally, let $U(r, c) := \{ Q \in \mathbb{R}^{K \times N}_+ | Q1 = r, Q^\top 1 = c \}$

\[
\min_{Q \in U(r,c)} \langle Q, -\log P \rangle
\]

Sought distribution “Q”

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>L3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$-\log P$ is log likelihood
Optimizing the same objective

\[ E(p, q) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{y=1}^{K} q(y|x_i) \log p(y|x_i) \]

Cross-entropy loss

\[ E(p, q) + \log N = \langle Q, -\log P \rangle \]

where \( \langle \cdot \rangle \) is the Frobenius dot-product between two matrices

\[ \min_{Q \in \mathcal{U}(r,c)} \langle Q, -\log P \rangle \]
Solving optimal transport problem

\[
\min_{Q \in U(r,c)} \langle Q, - \log P \rangle
\]

- This is a linear program and can be solved in polynomial time.
- Scales badly when Q is huge.
Introducing regularizer (Cuturi et al. 2013)

\[
\min_{Q \in U(r,c)} \langle Q, -\log P \rangle + \frac{1}{\lambda} \text{KL}(Q\|r_c^\top) \]

Strictly convex regularizer

\[ Q = \text{diag}(\alpha) P^\lambda \text{diag}(\beta) \]

Obtained by solving the Euler-Lagrange equation

- Unique optimal approximate solution.
- \(\lambda\) trades off convergence speed with closeness to the optimal transport problem.
Iterative optimizing of the labels

\[ \forall y : \alpha_y \leftarrow \left[ P^\lambda \beta \right]_y^{-1} \]

\[ \forall i : \beta_i \leftarrow \left[ \alpha^T P^\lambda \right]_i^{-1} \]

\[ Q = \text{diag}(\alpha) P^\lambda \text{diag}(\beta) \]
Entire method briefly

Aug → CNN → $h \circ \Phi$

- Initialize $Q$ (equipartitioned data)
- Clustering along multiple heads

$\forall y : \alpha_y \leftarrow [P^\lambda y]^{-1}_y$

$\forall i : \beta_i \leftarrow [\alpha^\top P^\lambda]^{-1}_i$

- Minimizing cross-entropy loss

- Optimizing labels

$\text{CNN}$

$\Phi$

$P$

$Q$

(label assignments)

(equipartitioned data)
Table of Contents

- Introduction
- Related Work
- Method
- Experiments
- Summary
Experiments

- REPRESENTATIONAL LEARNING STUDY.
- LINEAR PROBES – FEATURES CLASSIFICATION AT $C_x$.
- DIFFERENT ARCHITECTURES.
- DIFFERENT DATASETS.
Ablation: number of self-labelling steps

- “#opts” → Number of times label optimization is being run.

<table>
<thead>
<tr>
<th>Method</th>
<th>#opt.</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeLa [3k × 1]</td>
<td>0</td>
<td>20.8</td>
<td>18.3</td>
<td>13.4</td>
</tr>
<tr>
<td>SeLa [3k × 1]</td>
<td>40</td>
<td>42.7</td>
<td>43.4</td>
<td>39.2</td>
</tr>
<tr>
<td>SeLa [3k × 1]</td>
<td>80</td>
<td>43.0</td>
<td>44.7</td>
<td>40.9</td>
</tr>
<tr>
<td>SeLa [3k × 1]</td>
<td>160</td>
<td>42.4</td>
<td>44.6</td>
<td>40.7</td>
</tr>
</tbody>
</table>
Number of clusters K

- Increasing the number of clusters.

<table>
<thead>
<tr>
<th>Method</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeLa [1k × 1]</td>
<td>40.1</td>
<td>42.1</td>
<td>38.8</td>
</tr>
<tr>
<td>SeLa [3k × 1]</td>
<td>43.0</td>
<td>44.7</td>
<td>40.9</td>
</tr>
<tr>
<td>SeLa [5k × 1]</td>
<td>42.5</td>
<td>43.9</td>
<td>40.2</td>
</tr>
<tr>
<td>SeLa [10k × 1]</td>
<td>42.2</td>
<td>43.8</td>
<td>39.7</td>
</tr>
</tbody>
</table>
Number of heads $T$

- Number of clustering heads.
- +2% for AlexNet
- +10% for ResNet-50

<table>
<thead>
<tr>
<th>Method</th>
<th>Architecture</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeLa [3k × 1]</td>
<td>AlexNet</td>
<td>44.7</td>
</tr>
<tr>
<td>SeLa [3k × 10]</td>
<td>AlexNet</td>
<td>46.7</td>
</tr>
<tr>
<td>SeLa [3k × 1]</td>
<td>ResNet-50</td>
<td>51.8</td>
</tr>
<tr>
<td>SeLa [3k × 10]</td>
<td>ResNet-50</td>
<td>61.5</td>
</tr>
</tbody>
</table>
Label Transfer

- **Source** → labels assigned by SeLa.
- **Target** → supervised training with the assigned labels.

<table>
<thead>
<tr>
<th>Method</th>
<th>Source (Top-1)</th>
<th>Target (Top-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeLa [3k × 10] AlexNet (46.7)</td>
<td>AlexNet (46.5)</td>
<td></td>
</tr>
<tr>
<td>SeLa [3k × 1] ResNet-50 (51.8)</td>
<td>AlexNet (45.0)</td>
<td></td>
</tr>
<tr>
<td>SeLa [3k × 10] ResNet-50 (61.5)</td>
<td>AlexNet (48.4)</td>
<td></td>
</tr>
</tbody>
</table>
### Imbalanced datasets

<table>
<thead>
<tr>
<th>Training data/Method</th>
<th>kNN</th>
<th>Linear/conv5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIFAR-10</td>
<td>CIFAR-100</td>
</tr>
<tr>
<td><strong>CIFAR-10, full</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervised</td>
<td>92.1</td>
<td>24.0</td>
</tr>
<tr>
<td>ours (K-means) [128 × 1]</td>
<td>64.7 (−17.4)</td>
<td>19.3 (−4.7)</td>
</tr>
<tr>
<td>ours (SK) [128 × 1]</td>
<td>72.9 (−9.2)</td>
<td>28.9 (+4.0)</td>
</tr>
<tr>
<td><strong>CIFAR-10, light imbalance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervised</td>
<td>92.0</td>
<td>24.0</td>
</tr>
<tr>
<td>ours (K-means) [128 × 1]</td>
<td>64.2 (−17.8)</td>
<td>18.1 (−5.9)</td>
</tr>
<tr>
<td>ours (SK) [128 × 1]</td>
<td>71.7 (−10.3)</td>
<td>28.2 (+4.2)</td>
</tr>
<tr>
<td><strong>CIFAR-10, heavy imbalance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervised</td>
<td>86.7</td>
<td>22.6</td>
</tr>
<tr>
<td>ours (K-means) [128 × 1]</td>
<td>60.7 (−16.0)</td>
<td>17.8 (−4.8)</td>
</tr>
<tr>
<td>ours (SK) [128 × 1]</td>
<td>67.6 (−9.1)</td>
<td>26.7 (+3.9)</td>
</tr>
</tbody>
</table>
## Finetuning

<table>
<thead>
<tr>
<th>Method</th>
<th>PASCAL VOC Task</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cls.</td>
<td>Det.</td>
<td>Seg.</td>
</tr>
<tr>
<td></td>
<td>fc6-8</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td>ImageNet labels</td>
<td>78.9</td>
<td>79.9</td>
<td>59.1</td>
</tr>
<tr>
<td>Random</td>
<td>53.3</td>
<td>43.4</td>
<td></td>
</tr>
<tr>
<td>Random Rescaled</td>
<td>56.6</td>
<td>45.6</td>
<td>32.6</td>
</tr>
<tr>
<td>BiGAN</td>
<td>52.3</td>
<td>60.1</td>
<td>46.9</td>
</tr>
<tr>
<td>Context*</td>
<td>55.1</td>
<td>65.3</td>
<td>51.1</td>
</tr>
<tr>
<td>Context 2</td>
<td>69.6</td>
<td>55.8</td>
<td>41.4</td>
</tr>
<tr>
<td>CC+VGG</td>
<td>72.5</td>
<td>56.5</td>
<td>42.6</td>
</tr>
<tr>
<td>RotNet</td>
<td>70.9</td>
<td>73.0</td>
<td>54.4</td>
</tr>
<tr>
<td>DeepCluster*</td>
<td>72.0</td>
<td>73.4</td>
<td>55.4</td>
</tr>
<tr>
<td>RotNet+retrieval*</td>
<td>72.5</td>
<td>74.7</td>
<td>58.0</td>
</tr>
<tr>
<td>SeLa* [3k × 10]</td>
<td>73.1</td>
<td>75.3</td>
<td>55.9</td>
</tr>
<tr>
<td>SeLa* [3k × 10]−</td>
<td>74.4</td>
<td>75.9</td>
<td>57.8</td>
</tr>
<tr>
<td>SeLa* [3k × 10]−+Rot</td>
<td><strong>75.6</strong></td>
<td><strong>77.2</strong></td>
<td><strong>59.2</strong></td>
</tr>
</tbody>
</table>
Linear probing - AlexNet

<table>
<thead>
<tr>
<th>Method</th>
<th>ILSVRC-12</th>
<th></th>
<th>Places</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet supervised, (Zhang et al., 2017)</td>
<td>19.3</td>
<td>36.3 44.2 48.3 50.5</td>
<td>22.7</td>
<td>34.8 38.4 39.4 38.7</td>
</tr>
<tr>
<td>Places supervised, (Zhang et al., 2017)</td>
<td>-</td>
<td>-    - -</td>
<td>22.1</td>
<td>35.1 40.2 43.3 44.6</td>
</tr>
<tr>
<td>Random, (Zhang et al., 2017)</td>
<td>11.6</td>
<td>17.1 16.9 16.3 14.1</td>
<td>15.7</td>
<td>20.3 19.8 19.1 17.5</td>
</tr>
<tr>
<td>Inpainting, (Pathak et al., 2016)</td>
<td>14.1</td>
<td>20.7 21.0 19.8 15.5</td>
<td>18.2</td>
<td>23.2 23.4 21.9 18.4</td>
</tr>
<tr>
<td>BiGAN, (Donahue et al., 2017)</td>
<td>17.7</td>
<td>24.5 31.0 29.9 28.0</td>
<td>22.0</td>
<td>28.7 31.8 31.3 29.7</td>
</tr>
<tr>
<td>Instance retrieval, (Wu et al., 2018)</td>
<td>16.8</td>
<td>26.5 31.8 34.1 35.6</td>
<td>18.8</td>
<td>24.3 31.9 34.5 33.6</td>
</tr>
<tr>
<td>RotNet, (Gidaris et al., 2018)</td>
<td>18.8</td>
<td>31.7 38.7 38.2 36.5</td>
<td>21.5</td>
<td>31.0 35.1 34.6 33.7</td>
</tr>
<tr>
<td>AND*, (Huang et al., 2019)</td>
<td>15.6</td>
<td>27.0 35.9 39.7 37.9</td>
<td>-</td>
<td>-  -  -  -</td>
</tr>
<tr>
<td>CMC*, (Tian et al., 2019)</td>
<td>18.4</td>
<td>33.5 38.1 40.4 42.6</td>
<td>-</td>
<td>-  -  -  -</td>
</tr>
<tr>
<td>AET*, (Zhang et al., 2019)</td>
<td>19.3</td>
<td>35.4 44.0 43.6 42.4</td>
<td>22.1</td>
<td>32.9 37.1 36.2 34.7</td>
</tr>
<tr>
<td>RotNet+retrieval*, (Feng et al., 2019)</td>
<td>20.8</td>
<td>35.2 41.8 44.3 44.4</td>
<td>24.0</td>
<td>33.8 37.5 39.3 38.9</td>
</tr>
</tbody>
</table>

**1-crop evaluation**
### Linear probes - ResNet

<table>
<thead>
<tr>
<th>Method</th>
<th>Architecture</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised, (Donahue &amp; Simonyan, 2019)</td>
<td>ResNet-50</td>
<td>76.3</td>
<td>93.1</td>
</tr>
<tr>
<td>Jigsaw, (Kolesnikov et al., 2019)</td>
<td>ResNet-50</td>
<td>38.4</td>
<td>–</td>
</tr>
<tr>
<td>Rotation, (Kolesnikov et al., 2019)</td>
<td>ResNet-50</td>
<td>43.8</td>
<td>–</td>
</tr>
<tr>
<td>CPC, (Oord et al., 2018)</td>
<td>ResNet-101</td>
<td>48.7</td>
<td>73.6</td>
</tr>
<tr>
<td>BigBiGAN, (Donahue &amp; Simonyan, 2019)</td>
<td>ResNet-50</td>
<td>55.4</td>
<td>77.4</td>
</tr>
<tr>
<td>LocalAggregation, (Zhuang et al., 2019)</td>
<td>ResNet-50</td>
<td>60.2</td>
<td>–</td>
</tr>
<tr>
<td>Efficient CPC v2.1, (Hénaff et al., 2019)</td>
<td>ResNet-50</td>
<td>(63.8)</td>
<td>(85.3)</td>
</tr>
<tr>
<td>CMC, (Tian et al., 2019)</td>
<td>ResNet-50</td>
<td>(64.1)</td>
<td>(85.4)</td>
</tr>
<tr>
<td>MoCo, (He et al., 2019)</td>
<td>ResNet-50</td>
<td>60.6</td>
<td>–</td>
</tr>
<tr>
<td>PIRL, (Misra &amp; van der Maaten, 2019)*</td>
<td>ResNet-50</td>
<td><strong>63.6</strong></td>
<td>–</td>
</tr>
<tr>
<td>SeLa [3k × 10]</td>
<td>ResNet-50</td>
<td>61.5</td>
<td>84.0</td>
</tr>
</tbody>
</table>

*other architectures*

<table>
<thead>
<tr>
<th>Method</th>
<th>Architecture</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo, (He et al., 2019)</td>
<td>RevNet-50×4</td>
<td>68.6</td>
<td>–</td>
</tr>
<tr>
<td>Efficient CPC v2.1, (Hénaff et al., 2019)</td>
<td>ResNet-161</td>
<td><strong>71.5</strong></td>
<td>90.1</td>
</tr>
</tbody>
</table>
Visualization of pseudoclasses on the validation set.
A random sample of ImageNet validation set images associated to random pseudoclasses
Summary

- Self-supervised learning might run into degenerate solution.
- Equipartitioning the data.
- Optimizing same objective during feature learning and clustering.
- State-of-the-art on SVHN, CIFAR-10/100 and ImageNet for AlexNet and ResNet-50.
Thank you!

Questions?
Learning details

Unless otherwise noted, we train all our self-supervised models with SGD and initial learning rate 0.05 for 400 epochs with two learning rate drops where we divide the rate by ten at 150 and 300 epochs.
if first_epoch == 0:
    # initiate labels as shuffled.
    self.L = np.zeros((self.hc, N), dtype=np.int32)
    for nh in range(self.hc):
        for _i in range(N):
            self.L[nh, _i] = _i % self.outs[nh]
        self.L[nh] = np.random.permutation(self.L[nh])
Label optimization

```python
def optim(self):
    self.PS = self.PS.T  # now it is K x N
    r = np.ones((self.outs[nh], 1), dtype=self.dtype) / self.outs[nh]
    c = np.ones((N, 1), dtype=self.dtype) / N
    self.PS += self.lamb  # K x N
    inv_K = self.dtype(1./self.outs[nh])
    inv_N = self.dtype(1./N)
    err = 1e-6
    _counter = 0
    while err > 1e-1:
        r = inv_K / (self.PS @ c)  # (KxN)(N,1) = K x 1
        c_new = inv_N / (r.T @ self.PS).T  # ((1,K)(KxN)).T() = N x 1
        if _counter % 10 == 0:
            err = np.nansum(np.abs(c / c_new - 1))
        c = c_new
        _counter += 1
    print("error: ", err, " step ", _counter, flush=True)  # ", sum(T), flush=True")
    # inplace calculations.
    self.PS = np.squeeze(c)
    self.PS = self.PS.T
    self.PS *= np.squeeze(r)
    self.PS = self.PS.T
    argmaxes = np.nanargmax(self.PS, 0)  # size N
    newl = torch.LongTensor(argmaxes)
```
Optimization schedule

```python
# optimization times (spread exponentially), can also just be linear in practice (i.e. every n-th epoch)
self.optimize_times = [(self.num_epochs+2)*N] + \
  ((self.num_epochs+1.01)*N*(np.linspace(0, 1, args.nopts)**2)[::1]).tolist()

optimizer = torch.optim.SGD(filter(lambda p: p.requires_grad, self.model.parameters()),
    weight_decay=self.weight_decay,
    momentum=self.momentum,
    lr=self.lr)

if self.checkpoint_dir is not None and self.resume:
    self.L, first_epoch = files.load_checkpoint_all(self.checkpoint_dir, self.model, optimizer)
    print('found first epoch to be', first_epoch, flush=True)
    include = [(qq/N >= first_epoch) for qq in self.optimize_times]
    self.optimize_times = (np.array(self.optimize_times)[include]).tolist()
    print('We will optimize L at epochs:', [np.round(1.0 * t / N, 2) for t in self.optimize_times], flush=True)
```
Sample of images associated to the lowest entropy pseudoclasses
A random sample of ImageNet training set images associated to the random pseudoclasses.
Dieses Folienlayout bitte nur verwenden, um großformatige Abbildungen wie in diesem Beispiel zu zeigen.

Es ist nicht für Texte etc. vorgesehen.