Getting the most out of your annotations with Transfer Learning

Róger Bermúdez-Chacón

Computer Vision Laboratory - EPFL
The problem

Learn to recognize some pattern* in images

* Classification, Segmentation, Detection, ...
The learning

(Supervised?) Machine Learning

Feed examples, update model
The data

Electron Microscopy Imaging
(3D “stacks”)
The training data painstakingly annotated manually

Image data

Annotations
“But wait, it gets worse!”
postsynaptic cell (no vesicles)

presynaptic cell (vesicles)

synaptic cleft

context!

synapse segmentation
Annotating is hard

Student
(5 min into annotating)

Student
(1 month into annotating)

Neuroscientist
(after being asked to annotate)
Trained models

Recycling. More fun in Switzerland
Domain shift

*Reference data*
- a.k.a. **Source Domain** \( X_i^S \in \mathcal{D}^S \)
- Plenty of labeled data \( Y_S \)
- Can train a model \( f^S(X_i^S) \approx Y_i^S \)

*New data*
- a.k.a. **Target Domain** \( \mathcal{D}^S \neq \mathcal{D}^T \)
- Little (if any) annotated data \( Y^T ??? \)
- Needs different model \( f^S \neq f^T \)
Domain shift

Reference data
- a.k.a. Source Domain \( X_i^S \in \mathcal{D}^S \)

New data
- a.k.a. Target Domain \( \mathcal{D}^S \neq \mathcal{D}^T \)
Domain shift

Reference data
- a.k.a. Source Domain

New data
- a.k.a. Target Domain

\[ X_i^S \in D^S \]
\[ D^S \neq D^T \]

Same underlying problem.
Transfer learning

- Find some aspect of the problem that is shared
- Tinker with the data
- Tinker with the model
Two example approaches

Traditional machine learning
- Exploit visual correspondences
- Model adaptation

Deep learning
- Two-stream architecture
- Co-training + Regularization
Model adaptation from visual correspondences
Boosted Decision Trees

(stumps)
Adapt decision thresholds

$X_{ik} > T_k$
Visual correspondences

Objects of interest look similar across domains

Not similar enough to use same model, though
Multiple Instance Learning

(bags)
If we had perfect correspondences...

\[ \ell(X_i^S, X_i^T) = L_\delta \left( f(X_i^S; \theta^S) - f(X_i^T; \theta^T) \right) \]

Make labels agree

\[ \hat{\Gamma}^t = \arg \min_{\Gamma^t} \frac{1}{|X^S|} \sum_{X_i^S \in X^S} \sum_{X_i^T \in X^T} \ell(X_i^S, X_i^T) \]
Find the **new thresholds** for the **Target Domain** such that for all elements from the **source domain** the label is consistent with that predicted for the bag.

\[
\ell_{ij} = L_\delta \left( f_{\theta_s}(c^S_i) - f_{\theta_t}(c^t_{i,j}) \right)
\]

\[
\hat{\Gamma}^t = \arg \min_{\Gamma^t} \frac{1}{|C^s|} \sum_{c^s_i \in C^s} \text{softmin} [\ell_{i1}, \ell_{i2}, \ldots, \ell_{ik}]
\]

*softmin* \([\ell_1, \ldots, \ell_k] = -\frac{1}{r} \ln \frac{1}{k} \sum_{j=1}^{k} \exp(-r\ell_j)\)

Find the **new thresholds** for the **Target Domain** such that for all elements from the **source domain** the label is consistent with that predicted for the bag.
Does it work?
Mitochondria

Synapses
### Mitochondria

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Synapse</td>
<td>0.22</td>
<td>0.32</td>
<td>0.39</td>
<td>0.13</td>
<td>0.39</td>
<td>0.57</td>
</tr>
<tr>
<td>Mitochondria</td>
<td>0.50</td>
<td>0.39</td>
<td>0.57</td>
<td>0.24</td>
<td>0.59</td>
<td>0.62</td>
</tr>
</tbody>
</table>
Some caveats

- Getting candidate correspondences is slow
- Assumes same weights in the learned basis
- Features are hand designed
Coupled U-Nets
Two U-Nets together?
Strategy

- Co-train
- Get “inspiration” from source stream
  - Parameter regularization / sharing
- Enforce correlated outputs
  - Feature space alignment
Getting inspiration from the source stream

- Don’t let corresponding parameters arbitrarily diverge
- Share parameters on upstream layers
Enforcing correlated outputs

Can’t expect both distributions to match, but features should be correlated
\[ \mathcal{L} = \text{Source prediction} + \text{Target prediction} + \text{Weight constrains} + \text{Output correlation} \]
The soft Jaccard Index

Jaccard based in counts

Replace counts with soft indicator

\[ J(y, \hat{y}) = \frac{|y_+ \cap \hat{y}_+|}{|y_+ \cup \hat{y}_+|} \]

\[ J(y, \hat{y}) = \frac{|y_+ \cap \hat{y}_+|}{|y_+| + |y_- \cap \hat{y}_+|} \]

\[ J_{soft} = \frac{\sum_i y_i \ s(y_i, \hat{y}_i)}{\sum_i y_i + \sum_i (1 - y_i) \ s(1 - y_i, \hat{y}_i)} \]

\[ *s(a, b) = \exp(-\|a - b\|^2 / \sigma) \]
Does it work?
Synapses

Mitochondria
<table>
<thead>
<tr>
<th>Experiment</th>
<th>Full TD</th>
<th>NoDA</th>
<th>Tuning</th>
<th>SA+B</th>
<th>SLSB</th>
<th>MIVC</th>
<th>CORAL</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synapses Cerebellum → Cortex</td>
<td>0.7511</td>
<td>0.2224</td>
<td>0.1973</td>
<td>0.1350</td>
<td>0.6705</td>
<td>0.5709</td>
<td>0.6862</td>
<td>0.7230</td>
</tr>
<tr>
<td>Mitochondria Hippocampus → Striatum</td>
<td>0.8060</td>
<td>0.5054</td>
<td>0.6979</td>
<td>0.2417</td>
<td>0.6336</td>
<td>0.5933</td>
<td>0.6700</td>
<td>0.7456</td>
</tr>
</tbody>
</table>
Some caveats

● Is a Deep Net
  ○ You will need a GPU
  ○ You will need plenty of memory
  ○ You will need plenty of patience

● Works best with some labeling
To take home

- We can repurpose trained models
  - Yes, beyond fine-tuning
  - Not all is perfect
- “What unites us is greater than what divides us”
  - Extract and leverage the shared semantics of the data
- Time is money
  - We all should be using TL in our pipelines