Motivation

- The Information Bottleneck (IB) has been suggested to be the governing principle behind Deep Nets
- However, the claim that maximizing the IB Lagrangian in each layer leads to optimal performance was never validated
- We train deep nets layer-by-layer using the IB Lagrangian and show that it does lead to competitive performance

Theory

- Each layer should maximize the IB Lagrangian:
  \[ L_{IB} = I(Y; L_i) - \beta I(X; L_i) \]
- \( I(Y; L_i) \) - Mutual Information of latent representation \( L_i \) with target output \( Y \). Higher \( I(Y; L_i) \) \( \Rightarrow \) better classification
- \( I(X; L_i) \) - Mutual Information of latent representation \( L_i \) with input \( X \). Lower \( I(X; L_i) \) \( \Rightarrow \) better generalization
- \( I(X; L_i) = \infty \) for deterministic nets, we opt for \( I(X; L_i) + \epsilon \)

Mutual Information Neural Estimator

- Donsker-Varadhan Representation
  \[ I(U; V) = \sup_{D: \mathbb{R}^U \times \mathbb{R}^V \to \mathbb{R}} \mathbb{E} [D(U, V)] - \log \mathbb{E} [D(U, V)|U \cdot V] \]
- Maximize \( DV \) with DNN to estimate MI

Architectural Aware MINE

- MINE is inaccurate in high dimensions
- For example, it does not estimate infinite MI with the input
- Solution: Inform \( D(X, L_i) \) of the relation between \( X \) and \( L_i \) by implementing an internal copy of layers 1,…,\( i \) within it

How to train your IB deep-net

- Adversarial training – the MINEs are the discriminators while the Deep Net is the Generator
- Train Layer-By-Layer – freeze the parameters of each layer after training it

Results

<table>
<thead>
<tr>
<th>Data Set (%)</th>
<th>Training method</th>
<th>Test acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST 1%</td>
<td>Cross-Entropy</td>
<td>85.76</td>
</tr>
<tr>
<td></td>
<td>( L = I(Y; L_i) )</td>
<td>85.12</td>
</tr>
<tr>
<td></td>
<td>( L = I(Y; L_i) - \beta I(X; L_i) )</td>
<td>85.67</td>
</tr>
<tr>
<td>MNIST 100%</td>
<td>Cross-Entropy</td>
<td>97.77</td>
</tr>
<tr>
<td></td>
<td>( L = I(Y; L_i) )</td>
<td>97.31</td>
</tr>
<tr>
<td></td>
<td>( L = I(Y; L_i) - \beta I(X; L_i) )</td>
<td>98.05</td>
</tr>
<tr>
<td>CIFAR-10 100%</td>
<td>Cross-Entropy</td>
<td>60.59</td>
</tr>
<tr>
<td></td>
<td>( L = I(Y; L_i) )</td>
<td>61.39</td>
</tr>
<tr>
<td></td>
<td>( L = I(Y; L_i) - \beta I(X; L_i) )</td>
<td>62.78</td>
</tr>
</tbody>
</table>

Bottleneck Effect

- Higher bottleneck leads to a discretized and therefore more compressed latent representation

Information Plane Dynamics

- Two phase dynamics only appear in vanilla training with weight decay
- Two phase training dynamics are apparent with a succeeding 'compression' phase when training with the IB functional