Various Training Extensions

Yossi Keshet
Bar-Ilan University

June 7, 2021
Outline

- Multi-task learning
- Adversarial training
- Mixup
- Hyper Networks
Outline

• Multi-task learning
• Adversarial training
  • Mixup
• Hyper Networks
Multi-task Learning

- Several related predictions tasks
- Leverage the information in one of the tasks in order to improve the accuracy on the other tasks.
- How can we apply it to a NN?
Multi-task Learning

• A model that was trained on specific language can’t generalize to other languages.

• Why?
  – The acoustic representation isn't language invariant
Multi-task Learning

English

Hebrew
Multi-task Learning
Multi-task Learning

- Assuming the triangles (△) and squares (□), represent task-specific features, and the circles (○) are features which can be shared.
Multi-task Learning

- Assuming the triangles (△) and squares (□), represent task-specific features, and the circles (○) are features which can be shared.

- If we only care about a specific task, when leveraging other tasks information, we “suffer” from irrelevant features.
Multi-task Learning
Multi-task Learning
Multi-task Learning
Multi-task Learning

![Diagram of Multi-task Learning](image)
Multi-task Learning
Multi-task Learning

$$\text{Loss}_1$$

Multi-Class Language Classifier

Gradient reversal layer

$$-\lambda \frac{\partial L_1}{\partial w}$$

$$\text{Loss}_2$$

Acoustic Representation

RNN

$$\frac{\partial L_2}{\partial w}$$

Features Extraction
Multi-task Learning
Multi-task Learning

Diagram:
- Loss_1
- Multi-Class Language Classifier
- Gradient reversal layer
- Loss_2
- Acoustic Representation
- RNN
- Features Extraction
When the training is finished, we “prune” the side branch.
Figure 1: An illustration of our architecture. We first feed the network a sequence of acoustic features, then we take the output of some intermediate representation and use it also to classify the speaker.

Multi-task Learning: examples

Figure 2: Our structured prediction model combining both classification and adversary branches.

Shrem et al, “Dr.VOT: Measuring Positive and Negative Voice Onset Time in the Wild”, 2019
Outline

- Multi-task learning
- Adversarial training
- Mixup
- Hyper Networks
Outline

• Multi-task learning
• Adversarial training
  • Mixup
• Hyper Networks
**TL;DR** *mixup* constructs virtual training examples:

\[
\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \quad \text{where } x_i, x_j \text{ are raw input vectors}
\]

\[
\tilde{y} = \lambda y_i + (1 - \lambda)y_j, \quad \text{where } y_i, y_j \text{ are one-hot label encodings}
\]

where \( \lambda \in [0,1] \)
Empirical Risk Minimization (ERM)

Recall that we would like to minimize the Risk:

\[ R(f) = \mathbb{E}_{(x,y) \sim P} [\ell(f(x), y)] \quad \quad \quad \quad \quad R(f) = \int \ell(f(x), y) dP(x, y). \]
Empirical Risk Minimization (ERM)

Recall that we would like to minimize the Risk:

\[ R(f) = \mathbb{E}_{(x,y) \sim P} [\ell(f(x), y)] \]

Unfortunately, the distribution \( P \) is unknown in most practical situations. Instead, we usually have access to a set of training data \( D = \{(x_i, y_i)\}_{i=1}^n \) where \( (x_i, y_i) \sim P \).

We approximate \( P \) by the empirical distribution:

\[ P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i) \]
Empirical Risk Minimization (ERM)

Recall that we would like to minimize the Risk:

\[ R(f) = \mathbb{E}_{(x,y) \sim P} [\ell(f(x), y)] \]
\[ R(f) = \int \ell(f(x), y) dP(x, y). \]

Unfortunately, the distribution \( P \) is unknown in most practical situations. Instead, we usually have access to a set of training data \( \mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \) where \( (x_i, y_i) \sim P \)

We approximate \( P \) by the empirical distribution:

\[ P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^n \delta(x = x_i, y = y_i) \]

We approximate the expected risk by the empirical risk:

\[ R_\delta(f) = \int \ell(f(x), y) dP_\delta(x, y) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i) \]
Vicinal Risk Minimization (VRM)

But we could choose $P_\delta$ differently. In the Vicinal Risk Minimization (VRM) principle (Chapelle et al., 2000), the distribution $P$ is approximated by

$$P_\nu(\tilde{x}, \tilde{y}) = \frac{1}{n} \sum_{i=1}^{n} \nu(\tilde{x}, \tilde{y} | x_i, y_i),$$

where $\nu$ is a vicinity distribution that measures the probability of finding the virtual feature-target pair $(\tilde{x}, \tilde{y})$ in the vicinity of the training feature-target pair $(x_i, y_i)$. 
Vicinal Risk Minimization (VRM)

In particular, Chapelle et al. (2000) considered Gaussian vicinities

$$\nu(\tilde{x}, \tilde{y} | x_i, y_i) = \mathcal{N}(\tilde{x} - x_i, \sigma^2) \delta(\tilde{y} = y_i)$$

which is equivalent to augmenting the training data with additive Gaussian noise.
In particular, Chapelle et al. (2000) considered Gaussian vicinities

\[ \nu(\tilde{x}, \tilde{y}|x_i, y_i) = \mathcal{N}(\tilde{x} - x_i, \sigma^2)\delta(\tilde{y} = y_i) \]

which is equivalent to augmenting the training data with additive Gaussian noise.

To learn using VRM, we sample the vicinal distribution to construct a dataset \( \mathcal{D}_\nu := \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m \) and minimize the vicinal risk:

\[ R_\nu(f) = \frac{1}{m} \sum_{i=1}^m \ell(f(\tilde{x}_i), \tilde{y}_i) \]
mixup is a generic vicinal distribution:

\[
\mu(\bar{x}, \bar{y}|x_i, y_i) = \frac{1}{n} \sum_{j=1}^{n} \mathbb{E}_{\lambda}[\delta(\bar{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \bar{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j)]
\]

where \( \lambda \sim \text{Beta}(\alpha, \alpha) \), for \( \alpha \in (0, \infty) \).
mixup is a generic vicinal distribution:

\[
\mu(\tilde{x}, \tilde{y}|x_i, y_i) = \frac{1}{n} \sum_{j} \mathbb{E}_{\lambda}[\delta(\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j)]
\]

where \( \lambda \sim \text{Beta}(\alpha, \alpha) \), for \( \alpha \in (0, \infty) \).
Sampling from the *mixup* vicinal distribution produces virtual feature-target vectors

\[
\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \\
\tilde{y} = \lambda y_i + (1 - \lambda)y_j,
\]

where \((x_i, y_i)\) and \((x_j, y_j)\) are two feature-target vectors drawn at random from the training data, and \(\lambda \in [0,1]\).
mixup: a generic vicinal distribution

Sampling from the mixup vicinal distribution produces virtual feature-target vectors

\[ \tilde{x} = \lambda x_i + (1 - \lambda)x_j, \]
\[ \tilde{y} = \lambda y_i + (1 - \lambda)y_j, \]

where \((x_i, y_i)\) and \((x_j, y_j)\) are two feature-target vectors drawn at random from the training data, and \(\lambda \in [0,1]\).

The mixup hyper-parameter \(\alpha\) controls the strength of interpolation between feature-target pairs, recovering the ERM principle as \(\alpha \to 0\).

The authors found that \(\alpha \in [0.1, 0.4]\) leads to improved performance over ERM, whereas for large \(\alpha\) leads to underfitting.
mixup: analysis

The mixup vicinal distribution can be understood as a form of data augmentation that encourages the model $f$ to behave linearly in-between training examples.
mixup leads to more robust model behaviors in-between the training data.

Prediction errors in-between training data. Evaluated at $x = \lambda x_i + (1 - \lambda) x_j$, a prediction is counted as a “miss” if it does not belong to \{y_i, y_j\}. The model trained with mixup has fewer misses.
## mixup: performance

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Epochs</th>
<th>Top-1 Error</th>
<th>Top-5 Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>ERM (Goyal et al., 2017)</td>
<td>90</td>
<td>23.5</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>mixup $\alpha = 0.2$</td>
<td></td>
<td><strong>23.3</strong></td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>ResNet-101</td>
<td>ERM (Goyal et al., 2017)</td>
<td>90</td>
<td>22.1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>mixup $\alpha = 0.2$</td>
<td></td>
<td><strong>21.5</strong></td>
<td><strong>5.0</strong></td>
</tr>
<tr>
<td>ResNeXt-101</td>
<td>ERM (Xie et al., 2016)</td>
<td>100</td>
<td>21.2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ERM</td>
<td>90</td>
<td>21.2</td>
<td><strong>5.6</strong></td>
</tr>
<tr>
<td></td>
<td>mixup $\alpha = 0.4$</td>
<td>90</td>
<td><strong>20.7</strong></td>
<td><strong>5.3</strong></td>
</tr>
<tr>
<td>ResNeXt-101</td>
<td>ERM (Xie et al., 2016)</td>
<td>100</td>
<td>20.4</td>
<td><strong>5.3</strong></td>
</tr>
<tr>
<td></td>
<td>mixup $\alpha = 0.4$</td>
<td>90</td>
<td><strong>19.8</strong></td>
<td><strong>4.9</strong></td>
</tr>
</tbody>
</table>

Table 1: Validation errors for ERM and mixup on the development set of ImageNet-2012.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>ERM</th>
<th>mixup</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>PreAct ResNet-18</td>
<td>5.6</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td>WideResNet-28-10</td>
<td>3.8</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>DenseNet-BC-190</td>
<td>3.7</td>
<td>2.7</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>PreAct ResNet-18</td>
<td>25.6</td>
<td>21.1</td>
</tr>
<tr>
<td></td>
<td>WideResNet-28-10</td>
<td>19.4</td>
<td>17.5</td>
</tr>
<tr>
<td></td>
<td>DenseNet-BC-190</td>
<td>19.0</td>
<td>16.8</td>
</tr>
</tbody>
</table>

Test errors for ERM and mixup on the CIFAR experiments.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Validation set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet</td>
<td>ERM</td>
<td>9.8</td>
<td>10.3</td>
</tr>
<tr>
<td></td>
<td>mixup $\alpha = 0.1$</td>
<td>10.1</td>
<td>10.8</td>
</tr>
<tr>
<td></td>
<td>mixup $\alpha = 0.2$</td>
<td>10.2</td>
<td>11.3</td>
</tr>
<tr>
<td>VGG-11</td>
<td>ERM</td>
<td>5.0</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>mixup $\alpha = 0.1$</td>
<td>4.0</td>
<td>3.8</td>
</tr>
<tr>
<td></td>
<td>mixup $\alpha = 0.2$</td>
<td>3.9</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Classification errors of ERM and mixup on the Google commands dataset.
Manifold mixup

- On each update, select a random layer uniformly (including the input).

- Sample $\lambda \sim \text{Beta}(\alpha, \alpha)$

- Mix between two random examples from the minibatch at the selected layer with weights $\lambda$ and $(1-\lambda)$.

- Mix the labels for those two examples in the same way to construct a soft target, yielding the manifold mixup loss, which compares the soft target with the output obtained with the mixed layer.
Outline

- Multi-task learning
- Adversarial training
- Mixup
- Hyper Networks
Outline

• Multi-task learning

• Adversarial training

• Mixup

  • Hyper Networks
Hyper Network or Dynamic Networks

**TL;DR** one network is used to find the weights of another network.

Littlwin and Wolf, “Deep Meta Functionals for Shape Representation”, 2019
Hyper Network

\[ \theta_I = f(I, \theta_f) \]

\( f \) is a mapping from the input image \( I \) to the parameters \( \theta_I \) of network \( g \).

\[ g(p, \theta_I) \]

\( g \) is a classification function that maps a point \( p \) with coordinates \((x, y, z)\) in 3D into a score \( s^p_I \in [0, 1] \), such that the shape is defined by the classifier’s decision boundary.

Littlwin and Wolf, “Deep Meta Functionals for Shape Representation”, 2019
\[ \theta_I = f(I, \theta_f) \]

\[ s_I^p = g(p, \theta_I) \]

Littlwin and Wolf, “Deep Meta Functionals for Shape Representation”, 2019
\( s_I^p \) represents Bernoulli distributions \([1 - g(p, \theta_I), g(p, \theta_I)]\)

\[ \theta_I = f(I, \theta_f) \]

\[ s_I^p = g(p, \theta_I) \]

Littlwin and Wolf, “Deep Meta Functionals for Shape Representation”, 2019
Hyper Network

\[ s_I^p \text{ represents Bernoulli distributions } [1 - g(p, \theta_I), g(p, \theta_I)] \]

\[
\theta_I = f(I, \theta_f)
\]

\[ y(p) \in \{0, 1\} \text{ ground truth label: } p \text{ inside } y(p) = 1 \text{ or outside } y(p) = 0 \text{ the shape.} \]

\[ s_I^p = g(p, \theta_I) \]

Littelwin and Wolf, “Deep Meta Functionals for Shape Representation”, 2019
Hyper Network

\( s^p_I \) represents Bernoulli distributions \([1 - g(p, \theta_I), g(p, \theta_I)]\)

\[ \theta_I = f(I, \theta_f) \]

\( y(p) \in \{0,1\} \) ground truth label: \( p \) inside \( y(p) = 1 \) or outside \( y(p) = 0 \) the shape.

Training is done using the cross-entropy classification loss:

\[ s^p_I = g(p, \theta_I) \]
$s_i^p$ represents Bernoulli distributions $[1 - g(p, \theta_I), g(p, \theta_I)]$

$\theta_I = f(I, \theta_f)$

$y(p) \in \{0, 1\}$ ground truth label: $p$ inside $y(p) = 1$ or outside $y(p) = 0$ the shape.

Training is done using the cross-entropy classification loss:

\[
H(\theta_f, I) = -\int_V y(p)\log(g(p, f(I, \theta_f))) + (1 - y(p))\log(1 - g(p, f(I, \theta_f)))dp
\]

Littlwin and Wolf, “Deep Meta Functionals for Shape Representation”, 2019
Hyper Networks

Reconstructing the 3D model (inference):

Evaluate $s^p_I = g(p, \theta_I)$ using a grid of points $p \in [-1,1]^3$.
Use an algorithm (like the marching cube algorithm) to obtain a polygon mesh.
Evaluate $s^p_I = g(p, \theta_I)$ using a grid of points $p \in [-1,1]^3$.

Use an algorithm (like the marching cube algorithm) to obtain a polygon mesh.
Hyper Networks

Reconstructing the 3D model (inference):

Evaluate $s^p_I = g(p, \theta_I)$ using a grid of points $p \in [-1,1]^3$. Use an algorithm (like the marching cube algorithm) to obtain a polygon mesh.
Hyper Networks

State-of-the-art results in the shape reconstruction task

<table>
<thead>
<tr>
<th>Method</th>
<th>airplane</th>
<th>bench</th>
<th>cabinet</th>
<th>car</th>
<th>cellphone</th>
<th>chair</th>
<th>couch</th>
<th>firearm</th>
<th>lamp</th>
<th>monitor</th>
<th>speaker</th>
<th>table</th>
<th>watercraft</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D-R2N2 [8]</td>
<td>51.3</td>
<td>42.1</td>
<td>71.6</td>
<td>79.8</td>
<td>66.1</td>
<td>46.6</td>
<td>62.8</td>
<td>54.4</td>
<td>38.1</td>
<td>46.8</td>
<td>66.2</td>
<td>51.3</td>
<td>51.3</td>
<td>56.0</td>
</tr>
<tr>
<td>OGN [35]</td>
<td>58.7</td>
<td>48.1</td>
<td>72.9</td>
<td>81.6</td>
<td>70.2</td>
<td>48.3</td>
<td>64.6</td>
<td>59.3</td>
<td>39.8</td>
<td>50.2</td>
<td>63.7</td>
<td>53.6</td>
<td>63.2</td>
<td>59.6</td>
</tr>
<tr>
<td>PSGN [10]</td>
<td>60.1</td>
<td>55.0</td>
<td>77.1</td>
<td>83.1</td>
<td>74.9</td>
<td>54.4</td>
<td>70.8</td>
<td>60.4</td>
<td>46.2</td>
<td>55.2</td>
<td>73.7</td>
<td>60.6</td>
<td>61.1</td>
<td>64.0</td>
</tr>
<tr>
<td>VTN [32]</td>
<td>67.1</td>
<td>63.7</td>
<td>76.7</td>
<td>82.1</td>
<td>74.2</td>
<td>55.0</td>
<td>69.0</td>
<td>62.6</td>
<td>43.6</td>
<td>53.4</td>
<td>68.1</td>
<td>57.3</td>
<td>59.9</td>
<td>64.1</td>
</tr>
<tr>
<td>MTN [32]</td>
<td>64.7</td>
<td>57.7</td>
<td>77.6</td>
<td>85.0</td>
<td>75.6</td>
<td>54.7</td>
<td>68.1</td>
<td>61.6</td>
<td>40.8</td>
<td>53.2</td>
<td>70.1</td>
<td>57.3</td>
<td>59.1</td>
<td>63.5</td>
</tr>
<tr>
<td>PCDI [37]</td>
<td>61.2</td>
<td>60.9</td>
<td>68.3</td>
<td>83.2</td>
<td>74.4</td>
<td>57.2</td>
<td>69.9</td>
<td>69.5</td>
<td>46.4</td>
<td>61.4</td>
<td>69.8</td>
<td>61.5</td>
<td>58.5</td>
<td>64.8</td>
</tr>
<tr>
<td>Ours</td>
<td>71.4</td>
<td>65.9</td>
<td>79.3</td>
<td>87.1</td>
<td>79.1</td>
<td>60.7</td>
<td>74.8</td>
<td>68.0</td>
<td>48.6</td>
<td>61.7</td>
<td>73.8</td>
<td>62.8</td>
<td>65.4</td>
<td>69.1</td>
</tr>
</tbody>
</table>

Table 2. Shape reconstruction from a single image on ShapeNet-core at $32^3$ grid resolution. Mean IOU (%) per category is reported as well as the average IOU (%) over all 13 categories. Dataset provided by Choy et al. [8]
Questions?