Learning feed-forward one-shot learners

Luca Bertinetto,* João F. Henriques,* Jack Valmadre,* Philip H.S. Torr, and Andrea Vedaldi
One-shot learning:
Learning a new concept from 1 or few samples.

Examples:

- Specializing OCR to new writers or new alphabets.
- Single-object tracking.
Standard discriminative learning

**Starting point:** Standard SGD/back-propagation learning

Test image $z$

```
\[ \cdot \rightarrow \sigma \rightarrow \cdot \rightarrow \sigma \rightarrow \cdot \]
```

Prediction $y$ (same/different)

Exemplar $x$

```
\[ \cdot \rightarrow \cdot \rightarrow \sigma \rightarrow \cdot \rightarrow \cdot \]
```

Stochastic Gradient Descent / Back-propagation (minimize loss of exemplar)
Starting point: Standard SGD/back-propagation learning

### Problems:
- Scarce data/overfitting
- Lengthy optimization process
- No priors ("learning to learn")
**Parameter prediction**

**Idea:** Re-interpret “training” as “parameter prediction”

Test image $z$

Exemplar $x$

- **Red:** dynamic convolution
- **Green:** standard convolution
**Idea:** Re-interpret “training” as “parameter prediction”

- The second network (learnnet) has meta-parameters $w'_i$.
- Represent prior knowledge about how to “learn” (predict) parameters, from one exemplar.
Feed-forward one-shot learning:

- Learns a classifier (predicts $w_i$ from $x$) and evaluates it (on $z$) in one pass.

Because this is a standard computational graph, it can be differentiated with back-prop.

Training “meta-parameters” $w'_i$:

- Draw exemplar/test-image/label triplets ($x, z, y$).
- Back-propagate through graph, and update $w'_i$ with SGD.
Technical challenges

Typical number of parameters:
- Fully-connected: $4096 \times 4096 \approx 2 \times 10^7$
- Convolutional: $3 \times 3 \times 192 \times 256 \approx 4 \times 10^6$

The output-space for parameter prediction can be very large.

To predict this many outputs from a 4096-dim. vector:
$4096 \times 4 \times 10^6 \approx 1 \times 10^{10}$ parameters (15.2 GB)

- Storage issues
- Overfitting
Factorization

• We start with the **fully-connected** case (easier).

\[ y = W(z)x + b(z) \]

• The number of dynamic weights \( W(z) \) scales **quadratically** with the size of \( x \).

• Inspired by SVD, factorize:

\[ y = M'\text{diag}(w(z))Mx + b(z) \]
Factorization

- We start with the **fully-connected** case (easier).
  \[ y = W(z)x + b(z) \]

- The number of dynamic weights \( W(z) \) scales **quadratically** with the size of \( x \).

- Inspired by SVD, factorize:
  \[ y = M' \text{diag}(w(z))Mx + b(z) \]

  - **Learned offline and fixed.**
  - **Projection into space with independent factors of variation.**
  - **Predicted dynamically.**
  - **Scales linearly with size of \( x \).**
To be broadly useful, we need to generalize for convolutional layers.

**Factorized convolution:**
- $1 \times 1$ convolution ($M$)
- Diagonal convolution with $w(z)$
- $1 \times 1$ convolution ($M'$)

Diagonal convolution applies $k$ independent filters to $k$ input channels. In the fully-connected case ($1 \times 1$ output) reduces to $\text{diag}(w(z))$. 
The proposed architecture is reminiscent of **siamese networks**.

\[ f(z, x) = \Gamma(\varphi(x; W), \varphi(z; W)) \]

**Key differences:**

- Siamese net applies **same model** with **shared weights** to \( x \) and \( z \).
- The proposed “learnert” changes **intermediate** representations of another net (red).
Siamese networks

- To highlight that they are not mutually exclusive, a learnet can be used to **dynamically change** the parameters of a siamese net.
Experiments

Omniglot dataset

- 30 training alphabets, 20 testing alphabets.
- Resized to 28x28 pixels.
- Find match among 20 characters from same alphabet (chance is 95% error).
- Architecture: 3 conv layers, final layer $\Gamma$ is a weighted L1 distance.
- Learnet predicts parameters of 2$^{nd}$ conv.
# Experiments

## Omniglot dataset – results

<table>
<thead>
<tr>
<th>Model</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siamese</td>
<td>41.8</td>
</tr>
<tr>
<td>Siamese (unshared)</td>
<td>34.6</td>
</tr>
<tr>
<td>Learnet</td>
<td>28.6</td>
</tr>
<tr>
<td>Siamese learnet</td>
<td>31.4</td>
</tr>
</tbody>
</table>
Single-object tracking

- Can be naturally posed as a **one-shot learning** problem:
  1. **Learn** classifier, with the initial object patch as the exemplar.
  2. **Classify** patches over remaining video into object/background.

- Possible to update online, but in our experiments this was not needed.
Experiments

Training

- Used ImageNet Video dataset.
  - 4,500 videos / 1,200,000 bounding boxes.
  - 30 classes: mostly animals (~75%) and some vehicles (~25%). Class data ignored.

- Pos. pairs: patches near in time (same video).
- Neg. pairs: patches far away/different videos.

- Architecture: slim AlexNet (less channels, for speed).
Experiments

Fully convolutional architecture

- To efficiently classify many patches, the net is applied convolutionally to a larger image.
- Standard trick from detection; produces a heat map of possible object locations.
Experiments

Visual Object Tracking (VOT) 2015 benchmark – results

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (IoU)</th>
<th>Num failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siamese</td>
<td>0.465</td>
<td>105</td>
</tr>
<tr>
<td>Siamese (unshared)</td>
<td>0.447</td>
<td>131</td>
</tr>
<tr>
<td>Siamese learnet</td>
<td><strong>0.500</strong></td>
<td><strong>87</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (IoU)</th>
<th>Num failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSST</td>
<td>0.483</td>
<td>163</td>
</tr>
<tr>
<td>MEEM</td>
<td>0.458</td>
<td><strong>107</strong></td>
</tr>
<tr>
<td>MUSTer</td>
<td>0.471</td>
<td>132</td>
</tr>
<tr>
<td>DAT</td>
<td>0.442</td>
<td>113</td>
</tr>
<tr>
<td>SO-DLT</td>
<td><strong>0.540</strong></td>
<td>108</td>
</tr>
</tbody>
</table>
Experiments

Visualization – predicted filters and activations

Omniglot:

<table>
<thead>
<tr>
<th>z</th>
<th>x</th>
<th>Predicted filters $w(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Activations

ImageNet

Video:

<table>
<thead>
<tr>
<th>z</th>
<th>x</th>
<th>Predicted filters $w(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Activations
Conclusions

• It is possible to obtain the parameters of a deep network by a single feed-forward prediction.
• Related to siamese nets, more general.
• “Learning-to-learn” direction: Train meta-parameters by solving millions of small learning tasks offline, as feed-forward computations.