Learning Invariant Weights in Neural Networks (Supplementary material)

Tycho F.A. van der Ouderaa\textsuperscript{1} \hspace{2cm} Mark van der Wilk\textsuperscript{1}

\textsuperscript{1}Imperial College London, UK
APPE N DIX A: DETAILED DERIVATION OF VARIATIONAL INVERENCE

Applying Variational Inference (VI) \cite{Blei2017}, we maximise the marginal likelihood w.r.t. parameters \( \theta = \text{vec}(W_2) \) by minimizing the \( D_{KL}(.\|.) \)-divergence between approximate posterior \( q(W_2|\mu, \Sigma) \) and true posterior distribution of weights \( p(W_2|D) \), equivalent to maximizing the evidence lower bound (ELBO) denoted by \( \mathcal{L} \):

\[
\begin{align*}
\text{arg min} & \quad D_{KL}(q(W_2|\mu, \Sigma)||p(W_2|D)) \\
= & \quad \text{arg min} \quad \mathbb{E}_{q(W_2|\mu, \Sigma)} \left[ \log \frac{q(W_2|\mu, \Sigma)}{p(W_2|D)} \right] \\
= & \quad \text{arg min} \quad \mathbb{E}_{q(W_2|\mu, \Sigma)} \left[ \log \frac{q(W_2|\mu, \Sigma)}{p(W_2)p(D|W_2)} \right] + \log p(D) \\
= & \quad \text{arg min} \quad \mathbb{E}_{q(W_2|\mu, \Sigma)} \left[ \log q(W_2|\mu, \Sigma) \right] - \log p(W_2) - \mathbb{E}_{q(W_2|\mu, \Sigma)} \left[ \log p(D|W_2) \right] \\
= & \quad \text{arg max} \quad \mathbb{E}_{q(W_2|\mu, \Sigma)} \left[ \log p(D|W_2) \right] - D_{KL}(q(W_2|\mu, \Sigma)||p(W_2)) \\
= & \quad \text{arg max} \quad \mathbb{L}
\end{align*}
\]

We independently model the weight \( w_2^c \) for each class \( c \) with a full co-variance multivariate Gaussian distribution \( \mathcal{N}(w_2^c|\mu^c, \Sigma^c) \), parameterised by mean vector \( \mu^c \) and lower-triangular (Cholesky) decomposition of the co-variance \( (L^c)^T L^c = \Sigma^c \) to avoid computational issues, following \cite{Blei2017}. We can view the variational posterior \( q(W_2|\mu, \Sigma) \) as multivariate Gaussian over all classes with concatenated mean and block-diagonally stacked covariances from which we sample flattened matrix \( W_2 \) in one go, or -equivalently- sample row vectors \( w_2^c \) for each class and concatenate them to obtain matrix \( W_2 \). By sampling \( L \) times from variational approximation \( W_2^{(1)}, W_2^{(2)} \ldots W_2^{(L)} \sim q(W_2|\mu, \Sigma) \) we obtain a Monte Carlo estimate of \( \mathbb{E}_{W} := \mathbb{E}_{W_2 \sim q(W_2|\mu, \Sigma)} \) required to compute the final ELBO or negative loss \( \mathcal{L}(\theta, D) \):

\[
\begin{align*}
\mathcal{L}(\theta, D) &= \mathbb{E}_{q(W_2|\mu, \Sigma)} \left[ \log p(D|W_2) \right] - D_{KL}(p(W_2|\mu, \Sigma)||p(W_2)) \\
&= \mathbb{E}_{q(W_2|\mu, \Sigma)} \left[ \log p(D|W_2) \right] - \sum_c D_{KL}(\mathcal{N}(w_2^c|\mu^c, \Sigma^c)||p(w_2^c)) \\
&= \mathbb{E}_{q(W_2|\mu, \Sigma)} \left[ \log p(D|W_2) \right] - \sum_c D_{KL}(\mathcal{N}(w_2^c|\mu^c, \Sigma^c)||\mathcal{N}(0; \Sigma_p)) \\
&= - \sum_c \sum_i \sum_j \log \sigma_{y_{c,i}} (\mathbb{E}_{T \sim p_T(T)} \left[ W_2 \circ \phi \left( W_1 \circ T \circ x^{(i)} \right) \right] ) - \sum_c \frac{1}{2} \left[ \log \frac{\Sigma_p}{\Sigma_p} - D + \text{tr} \{ \Sigma_p \Sigma_p^{-1} \mu \} \right]
\end{align*}
\]

for every input \( x^{(i)} \), log soft-argmax output \( \sigma_{y_{c,i}} \) for class of corresponding label \( y_{c,i} \), fixed first layer weights \( W_1 \), prior weights \( \Sigma_p = I \alpha \), input dimensionality \( D \), and trace \( \text{tr}(.). \) To allow for mini-batching, we use the Stochastic Variational Bayes Estimate (SGVB) from \cite{Blei2017} of the ELBO or negative loss \( \hat{\mathcal{L}}(\theta, D) \):

\[
\begin{align*}
\hat{\mathcal{L}}(\theta, D) &= -N \frac{1}{M} \sum_l \sum_i \sum_j \log \sigma_{y_{c,i}} (\mathbb{E}_{T \sim p_T(T)} \left[ W_2 \circ \phi \left( W_1 \circ T \circ x^{(i)} \right) \right] ) - \sum_c \frac{1}{2} \left[ \log \frac{\Sigma_p}{\Sigma_p} - D + \text{tr} \{ \Sigma_p \Sigma_p^{-1} \mu \} \right]
\end{align*}
\]

where we can choose \( L = 1 \) if we use a sufficiently large batch size.
APPENDIX B: WEIGHT VISUALIZATIONS OF LEARNED ROTATIONAL INVARIANCE

Figure 1: Illustration of the features banks over training iterations. Features are randomly initialised with almost no rotational invariance and converge to particular filters with full rotational invariance when trained on fully rotated MNIST data.
APPENDIX C.1: ROTATIONAL INVARIANCE IN RFF NEURAL NETWORK

Figure 2: Predicted invariance over training iterations for different initial invariances for RFF neural network.

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Accuracy</th>
<th>ELBO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fully rotated</td>
<td>Partially rotated</td>
</tr>
<tr>
<td>Fixed 5°</td>
<td>79.29</td>
<td>86.71</td>
</tr>
<tr>
<td>Fixed 45°</td>
<td>87.35</td>
<td>91.13</td>
</tr>
<tr>
<td>Fixed 90°</td>
<td>90.33</td>
<td>91.69</td>
</tr>
<tr>
<td>Fixed 135°</td>
<td>91.19</td>
<td>91.04</td>
</tr>
<tr>
<td>Fixed 175°</td>
<td>91.57</td>
<td>90.47</td>
</tr>
<tr>
<td>Learned (5° Init)</td>
<td>91.72</td>
<td>92.34</td>
</tr>
<tr>
<td>Learned (45° Init)</td>
<td>91.65</td>
<td>92.31</td>
</tr>
<tr>
<td>Learned (90° Init)</td>
<td>91.65</td>
<td>92.37</td>
</tr>
<tr>
<td>Learned (135° Init)</td>
<td>91.66</td>
<td>92.37</td>
</tr>
<tr>
<td>Learned (175° Init)</td>
<td>91.68</td>
<td>91.69</td>
</tr>
</tbody>
</table>

Table 1: Table containing Test Accuracy and ELBO scores after training for experiments with RFF network. In bold: the best scores for fixed invariance and, for learned invariances, all scores that surpass the best score using fixed invariance.
APPENDIX C.2: ROTATIONAL INVARIANCE IN RELU NEURAL NETWORK

Figure 3: Predicted invariance over training iterations for different initial invariances of ReLU neural network with both input and output layer weights trained.

Table 2: Table containing Test Accuracy and ELBO scores after training for experiments of ReLU neural network with both input and output layer weights trained. In bold: the best scores for fixed invariance and, for learned invariances, all scores that surpass the best score using fixed invariance.
### APPENDIX C.3: DIFFERENT TRANSFORMATIONS IN RFF NETWORK

<table>
<thead>
<tr>
<th>Model</th>
<th>Fully rotated MNIST</th>
<th>Translated MNIST</th>
<th>Scaled MNIST</th>
<th>Regular MNIST</th>
<th>Fully rotated MNIST</th>
<th>Translated MNIST</th>
<th>Scaled MNIST</th>
<th>Regular MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular MLP</td>
<td>79.29</td>
<td>66.07</td>
<td>89.25</td>
<td>95.16</td>
<td>-1.14</td>
<td>-1.49</td>
<td>-0.69</td>
<td>-0.39</td>
</tr>
<tr>
<td>+ Rotation</td>
<td>92.59</td>
<td>75.06</td>
<td>88.66</td>
<td>96.59</td>
<td>-0.43</td>
<td>-1.08</td>
<td>-0.62</td>
<td>-0.26</td>
</tr>
<tr>
<td>+ Translation</td>
<td>83.66</td>
<td>87.81</td>
<td>86.15</td>
<td>96.78</td>
<td>-0.82</td>
<td>-0.64</td>
<td>-0.72</td>
<td>-0.24</td>
</tr>
<tr>
<td>+ Scale</td>
<td>82.77</td>
<td>75.48</td>
<td>91.31</td>
<td>96.52</td>
<td>-0.84</td>
<td>-1.08</td>
<td>-0.49</td>
<td>-0.26</td>
</tr>
<tr>
<td>+ Affine</td>
<td>92.64</td>
<td>87.77</td>
<td>90.58</td>
<td>97.38</td>
<td>-0.43</td>
<td>-0.64</td>
<td>-0.54</td>
<td>-0.21</td>
</tr>
</tbody>
</table>

Table 3: Test Accuracy and ELBO for learned invariance using different transformations in a shallow RFF neural network.

### APPENDIX C.4: DIFFERENT TRANSFORMATION IN RELU NETWORK

<table>
<thead>
<tr>
<th>Model</th>
<th>Fully rotated MNIST</th>
<th>Translated MNIST</th>
<th>Scaled MNIST</th>
<th>Regular MNIST</th>
<th>Fully rotated MNIST</th>
<th>Translated MNIST</th>
<th>Scaled MNIST</th>
<th>Regular MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular MLP</td>
<td>90.35</td>
<td>89.34</td>
<td>96.61</td>
<td>98.10</td>
<td>-0.06</td>
<td>-0.06</td>
<td>-0.03</td>
<td>-0.02</td>
</tr>
<tr>
<td>+ Rotation</td>
<td>98.05</td>
<td>94.08</td>
<td>97.62</td>
<td>98.64</td>
<td>-0.05</td>
<td>-0.06</td>
<td>-0.03</td>
<td>-0.02</td>
</tr>
<tr>
<td>+ Translation</td>
<td>93.59</td>
<td>97.87</td>
<td>97.98</td>
<td>98.76</td>
<td>-0.09</td>
<td>-0.06</td>
<td>-0.03</td>
<td>-0.02</td>
</tr>
<tr>
<td>+ Scale</td>
<td>93.80</td>
<td>94.30</td>
<td>98.06</td>
<td>98.35</td>
<td>-0.06</td>
<td>-0.06</td>
<td>-0.03</td>
<td>-0.02</td>
</tr>
<tr>
<td>+ Affine</td>
<td>98.14</td>
<td>97.66</td>
<td>98.31</td>
<td>98.93</td>
<td>-0.05</td>
<td>-0.06</td>
<td>-0.03</td>
<td>-0.02</td>
</tr>
</tbody>
</table>

Table 4: Test Accuracy and ELBO for learned invariance using different transformations in a shallow ReLU neural network.

### APPENDIX C.4: DIFFERENT TRANSFORMATION IN RELU NETWORK ON DATASETS WITH COMBINATIONS OF TWO INVARIANCES.

<table>
<thead>
<tr>
<th>Model</th>
<th>Fully rotated MNIST</th>
<th>Translated MNIST</th>
<th>Scaled MNIST</th>
<th>Regular MNIST</th>
<th>Fully rotated MNIST</th>
<th>Translated MNIST</th>
<th>Scaled MNIST</th>
<th>Regular MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular MLP</td>
<td>53.36</td>
<td>80.71</td>
<td>75.50</td>
<td>98.10</td>
<td>-0.26</td>
<td>-0.10</td>
<td>-0.12</td>
<td>-0.02</td>
</tr>
<tr>
<td>+ Rotation</td>
<td>85.35</td>
<td>95.66</td>
<td>85.42</td>
<td>98.64</td>
<td>-0.31</td>
<td>-0.10</td>
<td>-0.27</td>
<td>-0.02</td>
</tr>
<tr>
<td>+ Translation</td>
<td>83.84</td>
<td>83.40</td>
<td>91.77</td>
<td>98.76</td>
<td>-0.42</td>
<td>-0.16</td>
<td>-0.19</td>
<td>-0.02</td>
</tr>
<tr>
<td>+ Scale</td>
<td>55.63</td>
<td>89.81</td>
<td>86.04</td>
<td>98.35</td>
<td>-0.39</td>
<td>-0.12</td>
<td>-0.17</td>
<td>-0.02</td>
</tr>
<tr>
<td>+ Affine</td>
<td>89.37</td>
<td>95.88</td>
<td>91.95</td>
<td>98.93</td>
<td>-0.37</td>
<td>-0.09</td>
<td>-0.18</td>
<td>-0.02</td>
</tr>
</tbody>
</table>

Table 5: Test Accuracy and ELBO for learned invariance using different transformations in a shallow ReLU neural network on datasets augmented by two subsequent transformations (rotation+translation, rotation+scaling and translation+scaling). Surprisingly, the regular MLP ends up with the best ELBO in this experiment. We did not consistently observe the best ELBO for the regular MLP throughout optimization, and find that we can still use our method and the ELBO to learn invariances in this case. Again, we observe that models with learned invariances achieve the highest test accuracy.
APPENDIX D: DATASET DETAILS

All datasets have 60000 training examples and 10000 test examples and are created by taking regular MNIST or CIFAR-10 and applying random transformations:

**Regular MNIST Dataset:** MNIST handwritten digit database [?].

**Regular CIFAR-10 Dataset:** CIFAR-10 dataset with 10 classes [?].

**Partially rotated dataset:** Every sample rotated by radian angle $\theta$, sampled from $\theta \sim U[-\frac{\pi}{2}, \frac{\pi}{2}]$.

**Fully rotated dataset:** Every sample rotated by radian angle $\theta$, sampled from $\theta \sim U[-\pi, \pi]$.

**Translated dataset:** Translated samples relatively by $dx$ and $dy$ pixels, sampled from $dx, dy \sim U[-8, 8]$.

**Scaled dataset:** Every sample scaled around center with $\exp(s)$, sampled from $s \sim U[-\log(2), \log(2)]$. 
APPENDIX E: LIE GROUP GENERATORS

We follow [1] and, similarly, utilise six matrix generators:

\[
G_{\text{transx}} = G_1 = \begin{bmatrix}
0 & 0 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\]

\[
G_{\text{transy}} = G_2 = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix},
\]

\[
G_{\text{rot}} = G_3 = \begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\]

\[
G_{\text{scalex}} = G_4 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\]

\[
G_{\text{scaley}} = G_5 = \begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{bmatrix},
\]

\[
G_{\text{shear}} = G_6 = \begin{bmatrix}
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}.
\]

To parameterise affine transformations we compute the following matrix exponential [2]:

\[
T_\epsilon = \exp \left( \sum_i \epsilon_i \eta_i G_i \right),
\]

\[\epsilon \sim U[-1,1]^k\] (1)

Optionally, the values of \(\eta\) can be constrained to a positive range by passing them through a ‘softplus’-function, or in case of \(\eta_3 = \eta_{\text{rot}}\) to \([-\pi, \pi]\) using a scaled ‘tanh’ function, preventing double coverage on the unit circle. In practice, however, we did not find such constraints necessary as long as \(\eta_{\text{rot}}\) is reasonably initialised (e.g. \(\eta = 0\)).

By fixing certain \(\eta_i\) at 0, subsets of the generator matrices parameterise rotation, translation and scaling:

For rotation only:
- Learn \(\eta_3\).
- Fix \(\eta_i = 0\) for all \(i \neq 3\).

\[
T_\epsilon^{(\text{rot})} = \exp \left( \sum_i \epsilon_i \eta_i G_i \right)
\]

\[
= \exp \left( \epsilon_3 \eta_3 G_3 \right)
\]

\[
= \begin{bmatrix}
\cos(\epsilon_3 \eta_3) & -\sin(\epsilon_3 \eta_3) & 0 \\
\sin(\epsilon_3 \eta_3) & \cos(\epsilon_3 \eta_3) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

For translation only:
- Learn \(\eta_1\) and \(\eta_2\).
- Fix \(\eta_i = 0\) for all \(i > 2\).

\[
T_\epsilon^{(\text{trans})} = \exp \left( \sum_i \epsilon_i \eta_i G_i \right)
\]

\[
= \exp \left( \epsilon_1 \eta_1 G_1 + \epsilon_2 \eta_2 G_2 \right)
\]

\[
= \begin{bmatrix}
1 & 0 & \epsilon_1 \eta_1 \\
0 & 1 & \epsilon_2 \eta_2 \\
0 & 0 & 1
\end{bmatrix}
\]

For scaling only:
- Learn \(\eta_4\) and \(\eta_5\).
- Fix \(\eta_i = 0\) for all \(i \notin \{4, 5\}\).

\[
T_\epsilon^{(\text{scale})} = \exp \left( \sum_i \epsilon_i \eta_i G_i \right)
\]

\[
= \exp \left( \epsilon_4 \eta_4 G_4 + \epsilon_5 \eta_5 G_5 \right)
\]

\[
= \begin{bmatrix}
\exp(\epsilon_4 \eta_4) & 0 & 0 \\
0 & \exp(\epsilon_5 \eta_5) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]