A. Derivation of Likelihood Lower Bounds

In this appendix we detail the derivations of the log-likelihood lower bounds that were provided in Section 2.

EQUIVAE is relevant when a non-empty set of labelled data is available. We write the data set as

$$\mathcal{D} = \mathcal{D}_{\text{lab}} \cup \mathcal{D}_{\text{unlab}} = \{x_n, y_n\}_{n=1}^{N_{\text{lab}}} \cup \{x_n\}_{n=1}^{N_{\text{unlab}}} \tag{8}$$

We also decompose $\mathcal{D}_{\text{lab}} = \cup_y \mathcal{D}_{\text{lab}}^y$, where $\mathcal{D}_{\text{lab}}^y$ is the set of labelled instantiations x with label y. In particular, in what follows we think of $\mathcal{D}_{\text{lab}}^y$ as containing only the images x, not the labels, since they are specified by the index on the set. We require at least two labelled data points from each class, so that $|\mathcal{D}_{\text{lab}}^y| \geq 2 \; \forall y$.

We would like to maximise the log likelihood that our model generates both the labelled and the unlabelled data, which we write as:

$$\log p(\mathcal{D}) \;=\; \log p(\mathcal{D}_{\text{lab}}) \;+\; \log p(\mathcal{D}_{\text{unlab}}|\mathcal{D}_{\text{lab}}) \tag{9}$$

where we make explicit here the usage of labelled data in the unlabelled generative model.

For convenience, we begin by repeating the generative model for the labelled data in Equation 1, except with the deterministic integral over r_n completed:

$$p\big(\{x_n, y_n\}_{n=1}^{N_{\text{lab}}}\big) \;=\; \prod_{n=1}^{N_{\text{lab}}} \int dv_n \tag{10}$$
$$\times\, p_\theta\big(x_n | r(\mathcal{D}_{\text{lab}}^{y_n} \setminus \{x_n\}), v_n\big)\, p(v_n)\, p(y_n)$$

We will simplify the notation by writing $\widehat{x_n} = \mathcal{D}_{\text{lab}}^{y_n} \setminus \{x_n\}$ and $r_{y_n, \widehat{x_n}} = r(\mathcal{D}_{\text{lab}}^{y_n} \setminus \{x_n\})$, but keep all other details explicit.

We seek to construct a lower bound on $\log p(\{x_n, y_n\}_{n=1}^{N_{\text{lab}}})$, namely the log likelihood of the labelled data, using the following variational distribution over v_n (Equation 3):

$$q_{\phi_{\text{cov}}}(v_n | r_{y_n,\widehat{x_n}}, x_n) \tag{11}$$
$$= \mathcal{N}\big(\mu_{\phi_{\text{cov}}}(r_{y_n,\widehat{x_n}}, x_n), \sigma^2_{\phi_{\text{cov}}}(r_{y_n,\widehat{x_n}}, x_n) I\big)$$

Indeed,

$$\log p\big(\{x_n, y_n\}_{n=1}^{N_{\text{lab}}}\big) = \sum_{n=1}^{N_{\text{lab}}}$$
$$\times \log \mathbb{E}_{q_{\phi_{\text{cov}}}(v_n|r_{y_n,\widehat{x_n}},x_n)} \frac{p_\theta\big(x_n|r_{y_n,\widehat{x_n}}, v_n\big) p(v_n) p(y_n)}{q_{\phi_{\text{cov}}}\big(v_n|r_{y_n,\widehat{x_n}}, x_n\big)}$$
$$\overset{\text{Jensen's}}{\geq} \sum_{n=1}^{N_{\text{lab}}} \tag{12}$$
$$\times\, \mathbb{E}_{q_{\phi_{\text{cov}}}(v_n|r_{y_n,\widehat{x_n}},x_n)} \log \frac{p_\theta\big(x_n|r_{y_n,\widehat{x_n}}, v_n\big) p(v_n) p(y_n)}{q_{\phi_{\text{cov}}}\big(v_n|r_{y_n,\widehat{x_n}}, x_n\big)}$$
$$= \sum_{n=1}^{N_{\text{lab}}} \mathbb{E}_{q_{\phi_{\text{cov}}}(v_n|r_{y_n,\widehat{x_n}},x_n)} \log p_\theta\big(x_n|r_{y_n,\widehat{x_n}}, v_n\big)$$
$$- D_{\text{KL}}\big[q_{\phi_{\text{cov}}}(v_n|r_{y_n,\widehat{x_n}}, x_n)\big|\big|p(v_n)\big] + \log p(y_n)$$

Which coincides with the notationally simplified lower bound objective function given in Equation 4.

We now turn to the lower bound on the unlabelled data. To start, we marginalise over the labels on the unlabelled dataset:

$$p\big(\{x_n\}_{n=1}^{N_{\text{unlab}}}|\mathcal{D}_{\text{lab}}\big) = \prod_{n=1}^{N_{\text{unlab}}} \sum_{y_n} \int dv_n \tag{13}$$
$$\times\, p_\theta\big(x_n|r(\mathcal{D}_{\text{lab}}^{y_n}), v_n\big)\, p(v_n)\, p(y_n)$$

where we no longer need to remove x_n from $\mathcal{D}_{\text{lab}}^{y_n}$ in $r(\cdot)$ since for the unlabelled data, $x_n \notin \mathcal{D}_{\text{lab}}^{y_n}$.

As was done for the labelled data, we construct a lower bound using variational inference. However, in this case, we require a variational distribution over both y_n and v_n. We take:

$$q(v_n, y_n|x_n, \mathcal{D}_{\text{lab}}) \tag{14}$$
$$= q_{\phi_{\text{cov}}}\big(v_n\big|r(\mathcal{D}_{\text{lab}}^{y_n}), x_n\big)\; q_{\phi_{y\text{-post}}}(y_n|x_n)$$

which gives

$$\log p\big(\{x_n\}_{n=1}^{N_{\text{unlab}}}\big|\mathcal{D}_{\text{lab}}\big)$$
$$= \log \prod_{n=1}^{N_{\text{unlab}}} \mathbb{E}_{q_{\phi_{y\text{-post}}}(y_n|x_n)} \mathbb{E}_{q_{\phi_{\text{cov}}}(v_n|r(\mathcal{D}_{\text{lab}}^{y_n}),x_n)}$$
$$\times \frac{p_\theta\big(x_n|r(\mathcal{D}_{\text{lab}}^{y_n}), v_n\big)\, p(v_n)\, p(y_n)}{q_{\phi_{\text{cov}}}\big(v_n\big|r(\mathcal{D}_{\text{lab}}^{y_n}), x_n\big)\; q_{\phi_{y\text{-post}}}(y_n|x_n)}$$
$$\overset{\text{Jensen's}}{\geq} \sum_{n=1}^{N_{\text{unlab}}} \mathbb{E}_{q_{\phi_{y\text{-post}}}(y_n|x_n)} \mathbb{E}_{q_{\phi_{\text{cov}}}(v_n|r(\mathcal{D}_{\text{lab}}^{y_n}),x_n)} \tag{15}$$
$$\times \log \frac{p_\theta\big(x_n|r(\mathcal{D}_{\text{lab}}^{y_n}), v_n\big)\, p(v_n)\, p(y_n)}{q_{\phi_{\text{cov}}}\big(v_n\big|r(\mathcal{D}_{\text{lab}}^{y_n}), x_n\big)\; q_{\phi_{y\text{-post}}}(y_n|x_n)}$$
$$= \sum_{n=1}^{N_{\text{unlab}}} \mathbb{E}_{q_{\phi_{y\text{-post}}}(y_n|x_n)} \Bigg[$$
$$\mathbb{E}_{q_{\phi_{\text{cov}}}(v_n|r(\mathcal{D}_{\text{lab}}^{y_n}),x_n)} \log p_\theta\big(x_n|r(\mathcal{D}_{\text{lab}}^{y_n}), v_n\big)$$
$$- D_{\text{KL}}\big[q_{\phi_{\text{cov}}}(v_n|r(\mathcal{D}_{\text{lab}}^{y_n}), x_n)\big|\big|p(v_n)\big]\Bigg]$$
$$- D_{\text{KL}}\big[q_{\phi_{y\text{-post}}}(y_n|x_n)\big|\big|p(y_n)\big]$$

Thus, we have Equation 6 augmented with the notational decorations that were omitted in Section 2.

Therefore, the objective

$$\mathcal{L} \;=\; \sum_{n=1}^{N_{\text{unlab}}} \mathcal{L}_{\text{unlab}}^{(n)} \;+\; \sum_{n=1}^{N_{\text{lab}}} \mathcal{L}_{\text{lab}}^{(n)} \tag{16}$$

given in Equation 7, with $\mathcal{L}^{(n)}_{\text{lab}}$ given in Equation 4 and $\mathcal{L}^{(n)}_{\text{unlab}}$ given in Equation 6, is a lower bound on the data log likelihood.

B. Experimental Setup

In this appendix we provide details of the experimental setup that was used to generate the results from Section 3.

For our implementation of EQUIVAE, we use relatively standard neural networks. All of our experiments use implementations with well under 1 million parameters in total, converge within a few hours (on a Tesla K80 GPU), and are exposed to minimal hyperparameter tuning.

In particular, for the deterministic class-representation vector r_y given in Equation 2, we parametrise $f_{\theta_{\text{inv}}}(x)$ using a 5-layer, stride-2 (stride-1 first layer), with 5x5 kernal size, convolution network, followed by a dense hidden layer. The mean of these m embeddings $f_{\theta_{\text{inv}}}(x^i_y)$ is taken, followed then by another dense hidden layer, and the final linear dense output layer. This is shown for a $y = 6$ MNIST digit in the top shaded box of Figure 2. Our implementation uses $(8, 16, 32, 64, 64)$ filters in the convolution layers, and $(128, 64)$ hidden units in the two subsequent dense layers for a 16 dimensional latent (the number of units in the dense layers are halved when using 8 dimensional latents, as in our semi-supervised experiments on MNIST).

We parametrise the approximate posterior distribution $q_{\phi_{\text{cov}}}(v|r_y, x)$ over the equivariant latent as a diagonal-covariance normal distribution, $\mathcal{N}(\mu_{\phi_{\text{cov}}}(r_y, x), \sigma^2_{\phi_{\text{cov}}}(r_y, x))$, following the SGVB algorithm (Kingma & Welling, 2014; Rezende et al., 2014). For $\mu_{\phi_{\text{cov}}}(r_y, x)$ and $\sigma^2_{\phi_{\text{cov}}}(r_y, x)$, we use the identical convolution architecture as for the invariant embedding network as an initial embedding for the data point x. This embedding is then concatenated with the output of a single dense layer that transforms r_y, the output of which is then passed to one more dense hidden layer for each μ and σ^2 separately. This is shown in the bottom shaded box of Figure 2.

The generative model $p_\theta(x|r_y, v)$ is based on the DCGAN-style transposed convolutions (Radford et al., 2016), and is assumed to be a Bernoulli distribution for MNIST (Gaussian distribution for SVHN) over the conditionally independent image pixels. Both the invariant representation r_y and the equivariant representation v, are separately passed through a single-layer dense network before being concatenated and passed through another dense layer. This flat embedding that combines both representations is then transpose convolved to get the output image in a way the mirrors the 5-layer convolution network used to embed the representations in the first place. That is, we use $(64, 128)$ hidden units in the first two dense layers, and then $(64, 32, 16, 8, n_{\text{colours}})$ filters in each transpose convolution layer, all with 5x5 kernals and stride 2, except the last layer, which is a stride-1 convolution layer (with padding to accommodate different image sizes).

In our semi-supervised experiments, we implement $q_{\phi_{y\text{-post}}}(y|x)$ using the same (5-CNN, 1-dense) encoding block to provide an initial embedding for x. This is then concatenated with $\text{stop\_grad}(f_{\theta_{\text{inv}}}(x))$ and passed to a 2-layer dense dropout network with (128, 64) units. The use of $\text{stop\_grad}(f_{\theta_{\text{inv}}}(x))$ is simply that $f_{\theta_{\text{inv}}}(x)$ is learning a highly relevant, invariant representation of x that $q_{\phi_{y\text{-post}}}(y|x)$ might as well get access to. However, we do not allow gradients to pass through this operation since $f_{\theta_{\text{inv}}}(x)$ is meant to learn from the complementary data of known same-class members only.

As discussed in Section 2, the number of complementary samples m used to reconstruct r_y (see Equation 2) is chosen randomly at each training step in order to ensure that r_y is insensitive to m. For our supervised experiments where labelled data are plentiful, m is randomly select between 1 and m_{max} with $m_{\text{max}} = 7$ for MNIST ($m_{\text{max}} = 10$ for SVHN), whereas in the semi-supervised case $m_{\text{max}} = 4$ for MNIST ($m_{\text{max}} = 10$ for SVHN).

We perform standard, mild preprocessing on our data sets. MNIST is normalised so that each pixel value lies between 0 and 1. SVHN is normalised so that each pixel has zero mean and unit standard deviation over the entire dataset.

Finally, all activation functions that are not fixed by model outputs are taken to be rectified linear units. We use Adam (Kingma & Ba, 2015) for training with default settings, and choose a batch size of 32 at the beginning of training, which we double successively throughout training.