A DERIVATIONS

A.1 ANALYTIC SOLUTIONS TO LINEAR VAES

In this section, we present the analysis from Lucas et al. (2019) with additional details. Recall that the linear VAE is defined as:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{D}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \\ q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{E}(\mathbf{x} - \boldsymbol{\mu}), \mathbf{C}), \end{aligned} \tag{12}$$

where $\mathbf{D}$ is the decoder weight, $\mathbf{E}$ is the encoder weight, and $\mathbf{C}$ is the diagonal covariance matrix. Also recall that the β-VAE objective is defined as:

$$\mathcal{L}_\beta(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{p_d(\mathbf{x})}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[-\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]] + \beta\mathbb{E}_{p_d(\mathbf{x})}[D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}), p(\mathbf{z}))]. \tag{13}$$

For linear VAEs, each component of the β-VAE objective can be expressed in closed form. For simplicity, we assume a fixed identity covariance matrix $\mathbf{I}$ for the decoder as modifying β gives a similar effect as changing the observation noise (Rybkin et al., 2020). We also assume that the data covariance matrix is full rank. The second KL term can be expressed as:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}), p(\mathbf{z})) = \frac{1}{2}\left(-\log\det\mathbf{C} + (\mathbf{x} - \boldsymbol{\mu})^\top\mathbf{E}^\top\mathbf{E}(\mathbf{x} - \boldsymbol{\mu}) + \mathrm{tr}(\mathbf{C}) - d\right). \tag{14}$$

The first term can be expressed as:

$$\begin{aligned} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-(\mathbf{D}\mathbf{z} - (\mathbf{x} - \boldsymbol{\mu}))^\top(\mathbf{D}\mathbf{z} - (\mathbf{x} - \boldsymbol{\mu}))/2 - c\right] && (15) \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\frac{-(\mathbf{D}\mathbf{z})^\top(\mathbf{D}\mathbf{z}) + 2(\mathbf{x} - \boldsymbol{\mu})^\top\mathbf{D}\mathbf{z} - (\mathbf{x} - \boldsymbol{\mu})^\top(\mathbf{x} - \boldsymbol{\mu})}{2} - c\right], && (16) \end{aligned}$$

where $c = {}^d\!/_2 \log 2\pi$ is a constant term. Because $\mathbf{D}\mathbf{z} \sim \mathcal{N}(\mathbf{D}\mathbf{E}(\mathbf{x} - \boldsymbol{\mu}), \mathbf{D}\mathbf{C}\mathbf{D}^\top)$, we can further simplify the equation as:

$$\begin{aligned} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] = \frac{1}{2}(&-\mathrm{tr}(\mathbf{D}\mathbf{C}\mathbf{D}^\top) - (\mathbf{x} - \boldsymbol{\mu})^\top\mathbf{E}^\top\mathbf{D}^\top\mathbf{D}\mathbf{E}(\mathbf{x} - \boldsymbol{\mu}) && (17) \\ &+ 2(\mathbf{x} - \boldsymbol{\mu})^\top\mathbf{W}\mathbf{V}(\mathbf{x} - \boldsymbol{\mu}) - (\mathbf{x} - \boldsymbol{\mu})^\top(\mathbf{x} - \boldsymbol{\mu})) - c. && (18) \end{aligned}$$

After setting $\boldsymbol{\mu} = \boldsymbol{\mu}_{\mathrm{MLE}}$ (which is the global minimum for $\boldsymbol{\mu}$ as discussed in Lucas et al. (2019)), we take the gradient with respect to $\mathbf{C}$ and $\mathbf{E}$ to compute critical points of encoder weights and covariance matrix. These gradients can be written as:

$$\frac{\partial\mathcal{L}_\beta}{\partial\mathbf{C}}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{N}{2}\left(\beta\mathbf{C}^{-1} - \beta\mathbf{I} - \mathrm{diag}(\mathbf{D}^\top\mathbf{D})\right). \tag{19}$$

$$\frac{\partial\mathcal{L}_\beta}{\partial\mathbf{E}}(\boldsymbol{\phi}, \boldsymbol{\theta}) = N\left(\mathbf{D}^\top\mathbf{S} - \mathbf{D}^\top\mathbf{D}\mathbf{E}\mathbf{S} - \beta\mathbf{E}\mathbf{S}\right), \tag{20}$$

where $\mathbf{S}$ is the sample covariance matrix. By setting these gradients to equal to $\mathbf{0}$, the critical points can be expressed as:

$$\mathbf{C}^\star = \beta(\mathrm{diag}(\mathbf{D}^\top\mathbf{D}) + \beta\mathbf{I})^{-1}. \tag{21}$$

$$\mathbf{E}^\star = (\mathbf{D}^\top\mathbf{D} + \beta\mathbf{I})^{-1}\mathbf{D}^\top, \tag{22}$$

which recovers the true posterior mean and posterior covariance when the columns of $\mathbf{D}$ are orthogonal. Moreover, as the global minimum of the decoder weight coincides with the solution to the probabilistic PCA (pPCA) (Dai et al., 2018; Lucas et al., 2019; Sicks et al., 2021), we can express the optimal decoder weight as:

$$\mathbf{D}^\star = \mathbf{U}(g(\boldsymbol{\Lambda} - \beta\mathbf{I}))^{1/2}\mathbf{R}, \tag{23}$$

where $\mathbf{U}$ corresponds to the first k eigenvectors of the sample covariance $\mathbf{S}$ with corresponding top eigenvalues $\lambda_1 \geq ... \geq \lambda_k$ stored in the $k \times k$ diagonal matrix $\boldsymbol{\Lambda}$. The matrix $\mathbf{R}$ is some rotation matrix and the function $g(x) = \max(0, x)$ clips negative values to 0 and is applied elementwise.

A.2 PROOF OF THEOREM 1

In this section, we show that the MR-VAE parameterization can represent the response functions for linear VAEs. Throughout this section, we use $\eta = \log \beta$ for simplicity in notation. First, notice that we can simplify the equation for the encoder response function by using the singular value decomposition $\mathbf{D} = \mathbf{ASB}^\top$:

$$\mathbf{E}^\star(\eta) = \mathbf{B}(\mathbf{S}^2 + \exp(\eta)\mathbf{I})^{-1}\mathbf{SA}^\top. \tag{24}$$

Suppose we apply the MR-VAE parameterization to two-layer encoders $\mathbf{E}_{\boldsymbol{\psi}}(\eta) = \mathbf{E}^{(2)}_{\boldsymbol{\psi}}(\eta)\mathbf{E}^{(1)}_{\boldsymbol{\psi}}(\eta)$, where each component can be expressed as:

$$\mathbf{E}^{(1)}_{\boldsymbol{\psi}}(\eta) = \sigma_E(\mathbf{v}^{(1)}\eta + \mathbf{s}^{(1)}) \odot_{\text{row}} \mathbf{E}^{(1)}_{\text{base}} \tag{25}$$

$$\mathbf{E}^{(2)}_{\boldsymbol{\psi}}(\eta) = \sigma_E(\mathbf{v}^{(2)}\eta + \mathbf{s}^{(2)}) \odot_{\text{row}} \mathbf{E}^{(2)}_{\text{base}}, \tag{26}$$

where $\sigma_E(\cdot)$ is the sigmoid activation function. Consider setting $\mathbf{E}^{(1)}_{\text{base}} = \mathbf{A}^\top$, $\mathbf{v}^{(1)} = -\mathbf{1}$, and $\mathbf{s}^{(1)}_i = 2\log(\mathbf{S}_{ii})$ for all i in dimension of $\mathbf{s}^{(1)}$. Then, each dimension of the scaling term can be expressed as:

$$\sigma_E(\mathbf{v}^{(1)}\eta + \mathbf{s}^{(1)})_i = \frac{1}{1 + \exp(\eta - 2\log(\mathbf{S}_{ii}))} = \frac{\mathbf{S}^2_{ii}}{\mathbf{S}^2_{ii} + \exp(\eta)}. \tag{27}$$

By further setting $\mathbf{E}^{(2)}_{\boldsymbol{\psi}}(\eta) = \mathbf{BS}^{-1}$, which can be achieved by setting both scaling parameters ($\mathbf{v}^{(2)}$ and $\mathbf{s}^{(2)}$) to $\mathbf{0}$ and setting $\mathbf{E}^{(2)}_{\text{base}} = 2\mathbf{BS}^{-1}$, we recover the response function for the encoder weight:

$$\mathbf{E}^{(2)}_{\boldsymbol{\psi}}(\eta)\mathbf{E}^{(1)}_{\boldsymbol{\psi}}(\eta) = \mathbf{B}(\mathbf{S}^2 + \exp(\eta)\mathbf{I})^{-1}\mathbf{SA}^\top. \tag{28}$$

The response function for the diagonal covariance can be expressed as:

$$\mathbf{C}^\star(\eta) = \exp(\eta)(\text{diag}(\mathbf{D}^\top\mathbf{D}) + \exp(\eta)\mathbf{I})^{-1}. \tag{29}$$

Our hypernetwork parameterization corresponds to $\mathbf{C}_{\boldsymbol{\psi}}(\eta) = \mathbf{C}^{(2)}_{\boldsymbol{\psi}}(\eta)\mathbf{C}^{(1)}_{\boldsymbol{\psi}}(\eta)$, where each component can be written as:

$$\mathbf{C}^{(1)}_{\boldsymbol{\psi}}(\eta) = \sigma_E(\mathbf{t}^{(1)}\eta + \mathbf{p}^{(1)}) \odot_{\text{row}} \mathbf{C}^{(1)}_{\text{base}} \tag{30}$$

$$\mathbf{C}^{(2)}_{\boldsymbol{\psi}}(\eta) = \sigma_E(\mathbf{t}^{(2)}\eta + \mathbf{p}^{(2)}) \odot_{\text{row}} \mathbf{C}^{(2)}_{\text{base}}, \tag{31}$$

We describe the procedure for constructing the diagonal matrix $\mathbf{C}^{(1)}_{\text{base}}$: in the i-th dimension, if $\text{diag}(\mathbf{D}^\top\mathbf{D})_{ii} = 0$, we set the corresponding diagonal entry to 2 and the scaling (hyper) parameters to be $\mathbf{t}^{(1)}_i = \mathbf{p}^{(1)}_i = 0$.

Otherwise, if $\text{diag}(\mathbf{D}^\top\mathbf{D})_{ii} \neq 0$, we set the diagonal entry to 1. We set $\mathbf{t}^{(1)}_i = 1$ and $\mathbf{p}^{(1)}_i = -\log(\text{diag}(\mathbf{D}^\top\mathbf{D})_{ii})$ for each dimension. Finally, by letting $\mathbf{C}^{(2)}_{\boldsymbol{\psi}}(\eta) = \mathbf{I}$, we can achieve:

$$\mathbf{C}^{(2)}_{\boldsymbol{\psi}}(\eta)\mathbf{C}^{(1)}_{\boldsymbol{\psi}}(\eta) = \exp(\eta)(\text{diag}(\mathbf{D}^\top\mathbf{D}) + \exp(\eta)\mathbf{I})^{-1}. \tag{32}$$

Lastly, recall that the response function for decoder weight can be expressed as:

$$\mathbf{D}^\star(\eta) = \mathbf{U}(g(\boldsymbol{\Lambda} - \exp(\eta)\mathbf{I}))^{1/2}\mathbf{R}. \tag{33}$$

For notational simplicity, we represent $\boldsymbol{\Lambda}_{ii} = \exp(\boldsymbol{\xi}_i)$ for $i = 1, ..., k$ using the assumption above that the data covariance matrix is full rank. Then, the diagonal term in the response function can be represented as:

$$(g(\boldsymbol{\Lambda} - \exp(\eta)\mathbf{I}))^{1/2}_{ii} = \max(0, (\exp(\boldsymbol{\xi}_i) - \exp(\eta))^{1/2}) \tag{34}$$

$$= \exp(\boldsymbol{\xi}_i)^{1/2}\max(0, 1 - \exp(-\boldsymbol{\xi}_i)\exp(\eta))^{1/2} \tag{35}$$

$$= \exp(\boldsymbol{\xi}_i)^{1/2}\max(0, 1 - \exp(\eta - \boldsymbol{\xi}_i))^{1/2}. \tag{36}$$

Again, recall that we can use MR-VAE parameterization $\mathbf{D}_{\psi}(\eta) = \mathbf{D}^{(2)}_{\psi}(\eta)\mathbf{D}^{(1)}_{\psi}(\eta)$. Specifically, we have:

$$\mathbf{D}^{(1)}_{\psi}(\eta) = \sigma_D(\mathbf{h}^{(1)}\eta + \mathbf{q}^{(1)}) \odot_{\text{row}} \mathbf{D}^{(1)}_{\text{base}} \tag{37}$$

$$\mathbf{D}^{(2)}_{\psi}(\eta) = \sigma_D(\mathbf{h}^{(2)}\eta + \mathbf{q}^{(2)}) \odot_{\text{row}} \mathbf{D}^{(2)}_{\text{base}}, \tag{38}$$

We highlight that we use a different activation function $\sigma_D(\cdot)$ for representing the decoder weight as described in Eqn. 7. We propose to set $\mathbf{D}^{(1)}_{\text{base}} = \mathbf{R}$, $\mathbf{h}^{(1)} = \mathbf{1}$, $\mathbf{q}^{(1)}_i = -\boldsymbol{\xi}_i$, and $\mathbf{D}^{(2)}_{\psi}(\eta) = \mathbf{U}\boldsymbol{\Lambda}^{1/2}$ for all i. Then, the response hypernetwork can be expressed as:

$$\mathbf{D}^{(2)}_{\psi}(\eta)\mathbf{D}^{(1)}_{\psi}(\eta) = \mathbf{U}(g(\boldsymbol{\Lambda} - \exp(\eta)\mathbf{I}))^{1/2}\mathbf{R}. \tag{39}$$

which matches the response function for the decoder weight.

B Additional Implementation Details

B.1 Convolution Layers

In this section, we present the hypernetwork parameterization for convolutional layers. Consider the i-th layer of a convolutional VAE with C_i filters and kernel size K_i. We denote $\mathbf{W}^{(i,c)} \in \mathbb{R}^{C_{i-1} \times K_i \times K_i}$ and $\mathbf{b}^{(i,c)} \in \mathbb{R}$ to be the weight and bias of the c-th filter, where $c \in \{1, ..., C_i\}$. We formulate the response hypernetwork for the weight and bias as:

$$\mathbf{W}^{(i,c)}_{\psi}(\beta) = \sigma^{(i)}\left(\mathbf{w}^{(i,c)}_{\text{hyper}} \log \beta + \mathbf{b}^{(i,c)}_{\text{hyper}}\right) \odot \mathbf{W}^{(i,c)}_{\text{base}} \tag{40}$$

$$\mathbf{b}^{(i,c)}_{\psi}(\beta) = \sigma^{(i)}\left(\mathbf{w}^{(i,c)}_{\text{hyper}} \log \beta + \mathbf{b}^{(i,c)}_{\text{hyper}}\right) \odot \mathbf{b}^{(i,c)}_{\text{base}}, \tag{41}$$

where $\mathbf{w}^{(i,c)}_{\text{hyper}}, \mathbf{b}^{(i,c)}_{\text{hyper}} \in \mathbb{R}$. Observe that the proposed hypernetwork parameterization is similar to that of fully-connected layers and only requires $2C_i$ additional parameters to represent the weight and bias. In the forward pass, the convolutional response hypernetwork requires 2 additional elementwise operations. Hence, MR-VAEs parameterization for convolutional layers also incurs a small computation and memory overhead.

B.2 PyTorch Implementation

We show the PyTorch implementation of the MR-VAE layer in Listing 1. The MR-VAE layer can be viewed as pre-activations gating and can be introduced after linear or convolutional layers.

```
from torch import nn
import torch

class MRVAELayer(nn.Module):

 def __init__(self, features: int, activation_fnc: nn.Module) -> None:
  super().__init__()

  self.features = features
  self.hyper_block_scale = nn.Linear(1, self.features, bias=True)
  self.activation_fnc = activation_fnc

 def forward(self, inputs: torch.Tensor, betas: torch.Tensor) ->
     torch.Tensor:
  scale = self.hyper_block_scale(betas)
  scale = torch.activation_fnc(scale)

  if len(inputs.shape) == 4:
   # Unsqueeze for convolutional layers.
   scale = scale.unsqueeze(-1).unsqueeze(-1)

  return scale * inputs
```

Listing 1: MR-VAE Layer implemented in PyTorch.

Dataset	Architecture	Number of Parameters		Increase Percentage
		β-VAE	MR-VAE	
MNIST & Omniglot	CNN	17.53×10^5	17.88×10^5	2.04%
	ResNet	18.81×10^4	18.84×10^4	0.12%
	NVAE	88.68×10^5	89.50×10^5	0.92%
CIFAR & SVHN	CNN	40.46×10^5	40.81×10^5	0.88%
	ResNet	78.96×10^4	78.98×10^4	0.02%
CelebA64	CNN	34.59×10^5	34.96×10^5	1.08%
	ResNet	27.70×10^4	27.74×10^4	0.13%
Yahoo	LSTM	88.21×10^4	93.51×10^4	6.01%

Table 1: An overview of the parameters required by β-VAE and MR-VAE.

B.3 Memory Overhead for MR-VAE

In Table 1, we show the number of additional parameters MR-VAEs used in our experiment. The memory required for training MR-VAEs is similar to that of training a single β-VAE model.

C Experiment Details

This section describes the details of the experiments presented in Section 5. All experiments were implemented using PyTorch (Paszke et al., 2019) and Jax (Bradbury et al., 2018) libraries and conducted with NVIDIA P100 GPUs, except for NVAE models, which were conducted with NVIDIA RTX A5000 GPUs.

C.1 Linear VAEs

We used the MNIST (Deng, 2012) dataset for training linear VAEs. We used the Adam optimizer (Kingma & Ba, 2014) and trained the network for 200 epochs with 10 epochs of learning rate warmup and a cosine learning rate decay. The latent dimension was set to 32. We conducted hyperparameter searches over learning rates $\{0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001\}$ with βs uniformly sampled between 0.01 and 10 on a log scale. We selected the learning rate that achieved the lowest average training loss. The experiments were repeated 3 times with different random seeds, and the mean is presented.

MR-VAEs were trained using the same training and architectural configurations as the baseline. We performed a grid search on learning rates $\{0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001\}$ and selected the learning rate that achieved the best training loss at $\beta = 1$. We also repeated the experiments 3 times with different random seeds.

C.2 Image Reconstruction

We used binary static MNIST (Larochelle & Murray, 2011), Omniglot (Lake et al., 2015), CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and CelebA64 (Liu et al., 2015) datasets.

We used the Adam optimizer for CNN and ResNet (He et al., 2016) architectures for 200 epochs with a batch size of 128 and a cosine learning rate decay. For NVAE architecture with binary images, we used the Adamax optimizer for 400 epochs with a batch size of 256 and a cosine learning rate decay. In the case of CelebA64 dataset, we use a batch size of 16. For all baseline models, we conducted hyperparameter searches over learning rates $\{0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001\}$ with $\beta = 1$, making choices based on the final validation loss. With the chosen set of hyperparameters, we repeated the experiments over 10 βs uniformly sampled between 0.01 and 10 on

Dataset	Architecture	ELBO	Rate	Distortion
MNIST	ResNet	89.63	29.17	60.46
	CNN	96.35	18.33	78.03
	MR-ResNet	**87.89**	27.07	60.82
	MR-CNN	93.51	28.50	65.00
Omniglot	ResNet	111.07	34.98	76.09
	CNN	114.73	34.11	80.62
	MR-ResNet	**107.78**	34.36	73.41
	MR-CNN	110.98	34.21	76.77

Table 2: We present the lowest test ELBO scores for β-VAE and MR-VAE on MNIST and Omniglot datasets.

a log scale with and without the KL warm-up. The experiments were repeated 3 times with different random seeds, and the mean was presented.

We followed the same architectural configurations from Chadebec et al. (2022) for CNN and ResNet architectures[2]. We also adopted the NVAE architecture and configurations from Vahdat & Kautz (2020)[3].

MR-VAEs were trained using the same training and architectural configurations as the baseline. We performed a grid search on learning rates $\{0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001\}$ and selected the learning rate that achieved the best validation loss at $\beta = 1$. With different random seeds, the experiment was repeated 3 times, and the mean was shown.

Note that, for NVAE architecture on the CelebA64 dataset, we observed instability in the initial stage of training. To stabilize the early training stage, we used a warm-up stage for the initial 30% of the training, where the KL weight remained fixed to a small constant.

In addition, we used the Yahoo (Answer) (Zhang et al., 2015) dataset and adopted the training procedure from Hu et al. (2019). We trained the network for 200 epochs with a batch size of 32 using the Adam optimizer. The learning rate was decayed by a factor of 2 when the validation loss did not improve for 2 epochs. For all baseline methods, we performed hyperparameter searches over the learning rates $\{0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001\}$ with the KL warm-up and $\beta = 1$ and selected configuration that achieved the lowest validation loss. With the chosen set of hyperparameters, we repeated the experiments over 10 βs uniformly sampled between 0.01 and 10 on a log scale with and without the KL warm-up.

We also adopted the architectural configurations from Hu et al. (2019)[4]. LSTM models used the embedding dimension of 256 and the hidden dimension of 256. In training MR-VAEs, we followed the same configurations as in the baseline models. We performed a grid search on learning rates in the range $\{0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001\}$ and selected the learning rate that achieved the best validation loss at $\beta = 1$. We report the mean across 3 different random seeds.

C.2.1 β-TCVAE

We used the dSprites (Matthey et al., 2017) dataset and followed the same training and architectural configurations from Chen et al. (2018) [5]. We trained the network for 50 epochs with the Adam optimizer and a batch size of 2048. For the baseline method, we performed grid search over the learning rates $\{0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001\}$ with $\beta = 1$ and selected configuration that achieved the lowest training loss. With the chosen training configuration, we repeated the experiments over 10 βs uniformly sampled between 0.1 and 10 on a log scale. We report the mean across 3 different random seeds.

[2] https://github.com/clementchadebec/benchmark_VAE

[3] https://github.com/NVlabs/NVAE

[4] https://github.com/asyml/texar-pytorch

[5] https://github.com/rtqichen/beta-tcvae

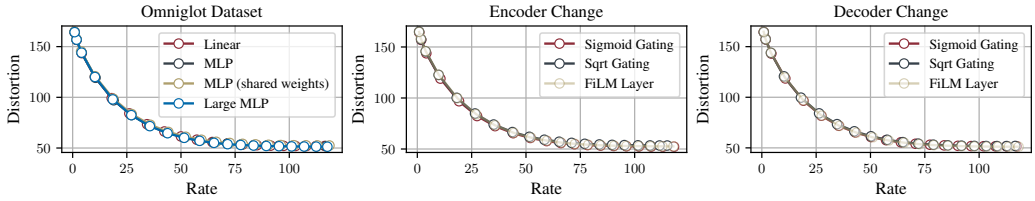

Figure 9: (left) An ablation studying the effect of different architectural designs for the hypernetwork. **(middle and right)** An ablation studying the effect of using different gating mechanisms on the hypernetwork activation. The MR-VAEs were trained on the Omniglot dataset.

The encoder is composed of 2 fully-connected layers with a hidden dimension of 1200 and ReLU activation functions. Similarly, the decoder consists of two fully-connected layers, but the Tanh activation function is used. A latent dimension of 10 is used. We followed the same grid search for training MR-VAEs and selected the learning rate that achieved the best training loss at $\beta = 1$. We also report the mean across 3 different random seeds.

D Additional Experiments

We further investigate the effect of modifying the architecture and activations of MR-VAE architectures. We used the ResNet encoder and decoder trained on the Omniglot dataset, following the same setup from the image reconstruction experiment.

Ablations over Architecture Design. In MR-VAEs, we apply an affine transformation to the $\log \beta$ to scale each layer's pre-activations. Here, we investigate if more expressive architectural designs can help in learning better rate-distortion curves. We repeated the experiment by changing the architecture for the scaling function, where we used linear transformation (default), a 2-layer MLP (with and without weight sharing on the first layer), and a 5-layer MLP. As shown in Figure 9, we observed that increasing the capacity of the model does not improve the final performance on the rate-distortion curve.

Ablations over Activation Function. Next, we investigated the effect of different design choices for gating the pre-activations. We repeated the experiment with sigmoid gating (default for encoders), sqrt gating (default for decoders), and FiLM (Perez et al., 2018) layer which scales and shifts the pre-activations. As shown in Figure 9, while MR-VAE's default choice performs slightly better compared to other combinations, we observed that the choice of the gating mechanism does not significantly impact the final performance on rate-distortion curves.