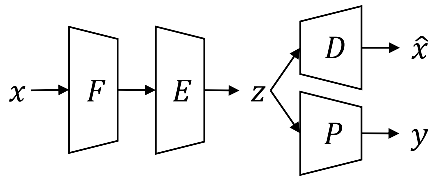


Figure 9: In experiments, we used a common feature-extractor (F), decoder (D), and predictor (P) network backbone while replacing different encoder heads (E).

7 Implementation details

Here, we include implementation details omitted from the main paper for brevity. Code for recreating the experiments in the paper is available at https://anonymous.4open.science/r/complexity_concepts-16A2. Upon acceptance, a deanonymized repository will be released.

7.1 Pretraining

In pre-training (before the finetuning with a small number of examples on a cruder task), we used the following setup. In general, the overall network architecture comprised a feature extractor, an encoder head, a decoder, and a predictor, as depicted in Figure 9.

In all experiments, the predictor was parametrized as a two-layer feedforward neural network with hidden dimension 128 and a ReLU activation. The last layer's dimension depended upon the exact prediction task (e.g., 10 neurons for FashionMNIST, 100 for CIFAR100, and 1010 for iNat) and used a softmax activation.

The feature extractors and decoders varied by domain. For FashionMNIST, the feature extractor used 3 2D convolution layers, followed by one fully connected layer. The decoder used two linear layers, followed by 3 inverse convolution layers. We again emphasize that the code for these models is available in our codebase, linked to at the beginning of this section.

For CIFAR100 and iNat, we pre-processed the images to extract the 512-dimensional activations from the penultimate layer of a ResNet18 pretrained on ImageNet [9]. These features were used as inputs to the feature extractor (x in Figure 9). For both CIFAR100 and iNat, the feature extractor used two linear layers, with a ReLU activation after the first layer, which had hidden dimension 128. The decoder was used to reconstruct the 512-dimension outputs of the ResNet18, using 3 fully-connected layers of dimension 128, 256, and 512, with ReLU activations between layers.

The different encoder heads were β-VAE, VQ-VIB$_{\mathcal{C}}$, and VQ-VIB$_{\mathcal{N}}$ models. β-VAE models used two linear layers, branching off the output of the feature extractor, to generate μ and σ from which to sample a continuous latent variable. VQ-VIB$_{\mathcal{C}}$ directly passed the output of the feature extractor into the vector quantization layer, from which the discrete latent representations were sampled, as described in the main paper. In VQ-VIB$_{\mathcal{N}}$, the output of the feature extractor was passed through two linear layers to generate a μ and a σ (exactly as in the β-VAE case) before the sampled continuous representation was discretized via vector quantization. Across experiments, the only differences among encoder heads that could arise were due to different latent dimensions (although we fixed it to 32 for all experiments) or, for VQ-VIB$_{\mathcal{C}}$ and VQ-VIB$_{\mathcal{N}}$, the number of elements in the learnable codebook or n, the number of quantized vectors to combine into a latent representation.

In the main paper, we discussed the VQ-VIB$_{\mathcal{C}}$ training loss (Equation 1), maximizing utility, minimizing reconstruction loss, and minimizing the entropy of the categorical distribution over codebook elements. A strict generalization of Equation 1, in which a variational bound on the complexity of representations is also penalized, is included in Equation 2:

$$
\begin{aligned}
\max \quad & \lambda_U \mathbb{E}[U(x,y)] - \lambda_I \mathbb{E}[\|x-\hat{x}\|^2] \\
& - \lambda_H \mathbb{E}\left[\sum_{i \in [1,n]} H(\mathbb{P}(\zeta | h_i(x))\right] \\
& - \lambda_C \mathbb{E}\left[D_{\mathrm{KL}}[\mathbb{P}(\zeta|h(x)) \| \mathcal{U}(C)]\right] \\
& - \|\mathtt{sg}[h_i(x)] - \zeta_i(x)\|^2 - \alpha \|h_i(x) - \mathtt{sg}[\zeta_i(x)]\|^2
\end{aligned}
\tag{2}
$$

Equation 2 differs from Equation 1 via the third line, penalizing the KL divergence between the conditional categorical distribution over codebook elements and a uniform distribution over the C elements. This provides a variational bound on $I(X, Z)$, dubbed the complexity of representations in prior literature [35, 27]. In our main experiments, we set $\lambda_C = 0$ and vary λ_H; ablation studies in which we varied λ_C instead of λ_H are included in Appendix 10 and confirm that controlling the entropy of representations supported better finetuning accuracy.

In training β-VAEs, we trained to maximize the function described in Equation 3, where $\mu(x)$ and $\sigma(x)$ represent the μ and σ parameters output by the encoder.

$$
\begin{aligned}
\max \quad & \lambda_U \mathbb{E}[U(x,y)] - \lambda_I \mathbb{E}[\|x-\hat{x}\|^2] \\
& - \lambda_C \mathbb{E}\left[D_{\mathrm{KL}}[\mathcal{N}(\mu(x), \sigma(x)) \| \mathcal{N}(0,1)]\right]
\end{aligned}
\tag{3}
$$

Equation 3 trains agents to maximize classification accuracy, minimize MSE, and minimize the complexity of representations. The scalar weight λ_C can be viewed as a Lagrange multiplier, constraining how much information can be encoded in representations. This equation is closely related to Equation 2 but, given the continuous nature of encodings in β-VAE, we could not penalize the entropy of a categorical distribution.

In training VQ-VIB$_{\mathcal{N}}$, we used the training objective proposed by Tucker et al. [27], which closely resembles the training loss for VQ-VIB$_{\mathcal{C}}$ and is shown in Equation 4 (and closely matches Equation 2). We use the same notation as for the β-VAE and VQ-VIB$_{\mathcal{C}}$ models.

$$
\begin{aligned}
\max \quad & \lambda_U \mathbb{E}[U(x,y)] - \lambda_I \mathbb{E}[\|x-\hat{x}\|^2] \\
& - \lambda_H \mathbb{E}\left[\hat{H}(\mathbb{P}(\zeta|\mu(x))\right] \\
& - \lambda_C \mathbb{E}\left[D_{\mathrm{KL}}[\mathcal{N}(\mu(x), \sigma(x)) \| \mathcal{N}(0,1)]\right] \\
& - \|\mathtt{sg}[h_i(x)] - \zeta_i(x)\|^2 - \alpha \|h_i(x) - \mathtt{sg}[\zeta_i(x)]\|^2
\end{aligned}
\tag{4}
$$

The two key differences between Equation 4 and Equation 2 (used for training VQ-VIB$_{\mathcal{C}}$) are bounds on entropy and complexity (on the second and third lines of Equation 4). Just as for β-VAEs, VQ-VIB$_{\mathcal{N}}$ models uses a KL divergence loss to regulate the complexity of representations. Increasing λ_C, as we did while annealing complexity in experiments, decreases the amount of encoded information. However, as shown in our results, simply increasing λ_C does not ensure that VQ-VIB$_{\mathcal{N}}$ models will use fewer discrete representations. (For visualizations of this effect, see Appendix 9.) Tucker et al. [27] advocate for using a small positive λ_H to penalize the estimated entropy over codebook elements. We explore varying λ_H in Appendix 10 and find some benefits relative to our main experiments, in which we set $\lambda_H = 0$. However, given that VQ-VIB$_{\mathcal{N}}$ only supports an *approximation* of the entropy term, we find that controlling the entropy for VQ-VIB$_{\mathcal{N}}$ is not as effective as controlling the entropy for VQ-VIB$_{\mathcal{C}}$.

7.2 Finetuning

In finetuning, we loaded pretrained frozen encoders and trained new predictor models to map from encodings to downstream predictions.

For a finetuning task with M distinct classes, and a "duplication factor," k, for how many examples of each class to train with, we randomly selected $k * M$ datapoints to train with. For example, when finetuning on the binary CIFAR100 task of living vs. non-living things, $M = 2$, so we loaded 2 total datapoints for $k = 1$, 4 datapoints for $k = 2$, etc.. For each input in the finetuning dataset, we generated an encoding by passing through the encoder once. This generated a new dataset of encodings and labels, which we used the train the predictor. (Note that this approach is distinct from passing the input through the encoder many times; given stochastic encoders, which we used, the same input could result in many different encodings. Here, we assumed a limited budget of *encodings*.)

Predictor neural networks were instantiated as feedforward networks with 4 fully connected layers, with hidden dimension 256 and ReLU activations, and trained to map from shifted encodings (see next paragraph) to classifications for 100 epochs using an Adam optimizer with default parameters, with the learning rate decreasing by a factor of 10 based on plateauing training loss, with a patience of 5 epochs, and early stopping if the learning rate fell below 10^{-8}.

One particular design choice that we made in finetuning predictors merits elaboration: shifting encodings. Rather than directly training predictors to map from encodings to predictions, we applied a simple linear transformation to the encodings before feeding them to the predictor. Specifically, we multiplied all tensor elements by 5 and increased them by 1. This simple linear transformation does not affect any relations between encodings except scale, and indeed we found that predictors could be successfully trained with this rescaling. Nevertheless, this linear transformation was important to provide a check against merely relying upon initialization conditions for good finetuning performance. In particular, we found that if we did not apply this linear transformation (i.e., pass the raw encodings to the predictor), predictors sometimes performed better than they should given the training data. For example, as a sanity check, we trained a predictor on a binary classification task, but only provided two positive datapoints and no negative data. In general, given this data, one would expect a trained predictor to only predict positive labels. However, we observed that in several cases, the predictor would achieve nearly perfect accuracy, including predicting negative labels for negative inputs. This surprising result disappeared when we simply shifted encodings, indicating that the particular initialization conditions of the predictor seemed to align well with pre-trained encoders. We wanted to measure the effect of data on finetuning performance, rather than just initialization conditions, but we note that this odd phenomenon of well-aligned initializations merits further investigation.

We ran 10 finetuning trials per model, which was important given the small amount of randomly-sampled finetuning data.

7.3 Hyperparameters

In the following subsections, we present the hyperparameters used for training different encoders in the different domains. In general, we used the following principles when choosing hyperparameters:

- For VQ-based methods, use a large enough codebook to have at least one element per class. Larger C are also acceptable, as tuning weights should decrease the effective codebook size.
- When annealing, use a small enough weight increment to generate smooth changes during training. Larger increments, however, speed up training.
- When annealing for larger n one can increase the annealing rate. Models with greater n tended to use more complex representations, so annealing could be extremely slow for small increments.

7.3.1 FashionMNIST

For FashionMNIST, we trained all models with batch size 64 for 200 epochs, using the hyperparameters specified in Table 1. The only differences across methods were which hyperparameters we annealed to penalize complexity. Other differences simply reflected differences in architecture (e.g., using a codebook for vector-quantization methods). Pre-training a single model for 200 epochs took approximately 5 minutes on a desktop computer with one NVIDIA 2080 GeForce RTX.

7.3.2 CIFAR100

For CIFAR100, we trained all models with batch size 256 for 400 epochs, using the hyperparameters specified in Table 2. As explained for FashionMNIST, the only substantial differences across

Table 1: Hyperparameters for FashionMNIST training.

ENCODER	LATENT DIM	C	n	λ_U	λ_I	λ_{C_0}	λ_C INCR	λ_{H_0}	λ_H INCR
β-VAE	32	NA	NA	10	10	0.01	0.5	0.0	0.0
VQ-VIB$_{\mathcal{N}}$	32	1000	1	10	10	0.01	0.5	0.0	0.0
VQ-VIB$_{\mathcal{N}}$	32	1000	2	10	10	0.01	0.5	0.0	0.0
VQ-VIB$_{\mathcal{N}}$	32	1000	4	10	10	0.01	0.5	0.0	0.0
VQ-VIB$_{\mathcal{C}}$	32	1000	1	10	10	0.0	0.0	0.001	0.2
VQ-VIB$_{\mathcal{C}}$	32	1000	2	10	10	0.0	0.0	0.001	0.4
VQ-VIB$_{\mathcal{C}}$	32	1000	4	10	10	0.0	0.0	0.001	0.8

Table 2: Hyperparameters for CIFAR100 training.

ENCODER	LATENT DIM	C	n	λ_U	λ_I	λ_{C_0}	λ_C INCR	λ_{H_0}	λ_H INCR
β-VAE	32	NA	NA	10	10	0.01	0.1	0.0	0.0
VQ-VIB$_{\mathcal{N}}$	32	1000	1	10	10	0.01	0.5	0.0	0.0
VQ-VIB$_{\mathcal{N}}$	32	1000	2	10	10	0.01	0.5	0.0	0.0
VQ-VIB$_{\mathcal{N}}$	32	1000	4	10	10	0.01	0.5	0.0	0.0
VQ-VIB$_{\mathcal{C}}$	32	1000	1	10	10	0.0	0.0	0.001	0.04
VQ-VIB$_{\mathcal{C}}$	32	1000	2	10	10	0.0	0.0	0.001	0.08
VQ-VIB$_{\mathcal{C}}$	32	1000	4	10	10	0.0	0.0	0.001	0.12

architectures were architecture-specific terms that needed to be specified, and which terms were annealed to penalize complexity. Pre-training a single model for 400 epochs took approximately 10 minutes on a desktop computer with one NVIDIA 3080.

7.3.3 iNaturalist

For iNat, we trained all models with batch size 256, using the hyperparameters specified in Table 3. We trained β-VAE and VQ-VIB$_{\mathcal{C}}$ models for 300 epochs, while we trained VQ-VIB$_{\mathcal{N}}$ models for 600. We used more annealing epochs for VQ-VIB$_{\mathcal{N}}$ simply because it seemed to need more epochs to eventually anneal to random chance. Likely, a larger annealing rate could also accomplish the desired effect, but initial experiments with faster annealing tended to induce rapid codebook collapse that did not generate the smooth spectrum of MSE values we desired. Pre-training a single model for 300 epochs took approximately 10 minutes on a desktop computer with one NVIDIA 3080 (and twice as long for 600 epochs).

8 Further Finetuning Results

In the main paper, we included only a small number of graphs highlighting our key results. Here, we include further results that corroborate the main trends stated in the paper. Primarily, these plots include further experiments for varying the amount of finetuning data, as well as varying n, the number of codebook elements to combine into a latent representation. Results are divided by domain: FashionMNIST, CIFAR100, and iNat.

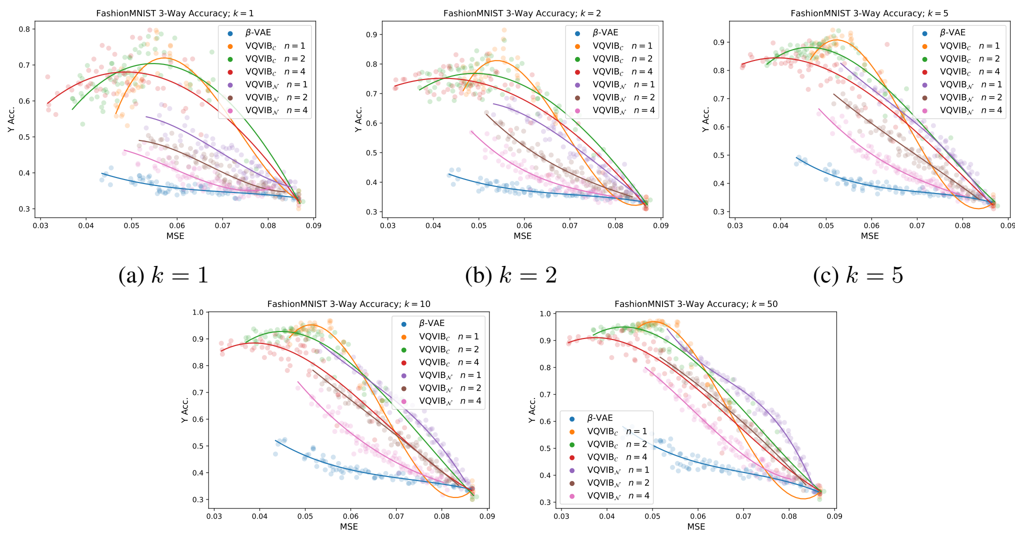
8.1 FashionMNIST

Here, we include the finetuning results for the FashionMNIST domain for varying amounts of finetuning data, ranging over $k \in [1, 2, 5, 10, 50]$, and n, the number of quantized vectors to combine into a single representation. Results for each k are included in Figure 10.

As expected, increasing the amount of finetuning data improved performance for all models, and the gap between all model types (VQ-VIB$_{\mathcal{C}}$, VQ-VIB$_{\mathcal{N}}$, and β-VAE) shrank. It is noteworthy, however, that a VQ-VIB$_{\mathcal{C}}$ model, tuned to the right complexity level and trained with just one example per ternary class (Figure 10 a), achieved better accuracy than a β-VAE model trained with 50 examples per class (Figure 10 e). Further, for any fixed k, VQ-VIB$_{\mathcal{C}}$ consistently outperformed VQ-VIB$_{\mathcal{N}}$, suggesting that many recent works that use VQ-VIB $_{\mathcal{N}}$ could be improved by replacing the model

Table 3: Hyperparameters for iNaturalist training.

ENCODER	LATENT DIM	C	n	λ_U	λ_I	λ_{C_0}	λ_C INCR	λ_{H_0}	λ_H INCR
β-VAE	32	NA	NA	10	10	0.0001	0.03	0.0	0.0
VQ-VIB$_\mathcal{N}$	32	2000	1	10	10	0.001	0.1	0.0	0.0
VQ-VIB$_\mathcal{N}$	32	2000	2	10	10	0.001	0.2	0.0	0.0
VQ-VIB$_\mathcal{N}$	32	2000	4	10	10	0.001	0.4	0.0	0.0
VQ-VIB$_\mathcal{C}$	32	2000	1	10	10	0.0	0.0	0.001	0.05
VQ-VIB$_\mathcal{C}$	32	2000	2	10	10	0.0	0.0	0.001	0.10
VQ-VIB$_\mathcal{C}$	32	2000	4	10	10	0.0	0.0	0.001	0.20



(a) $k=1$ (b) $k=2$ (c) $k=5$

(d) $k=10$ (e) $k=50$

Figure 10: FashionMNIST finetuning results for varying k. As k increased, all models benefited. The data-efficiency of advantage of VQ-VIB$_\mathcal{C}$ was most pronounced when using the least amount of data.

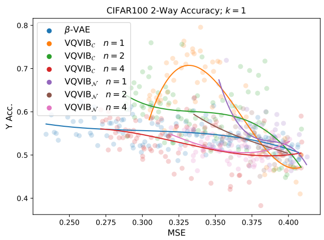
type [27, 28, 13, 8]. Lastly, for both VQ-VIB$_\mathcal{N}$ and VQ-VIB$_\mathcal{C}$, increasing n tended to support lower MSE but worse finetuning accuracy. This supports an intuition that combining more discrete representations starts to more densely fill the representation space, trending towards continuous representations.

We note briefly that VQ-VIB$_\mathcal{N}$, both in this domain and others (explored in the next sections), typically failed to learn as complex representations as either VQ-VIB$_\mathcal{C}$ or β-VAEs. This is apparent given the limited range of MSE values for the VQ-VIB$_\mathcal{N}$ curves. We consistently struggled to make VQ-VIB$_\mathcal{N}$ learn as rich representations as for the other model types, which led to worse reconstructions and higher MSE values.

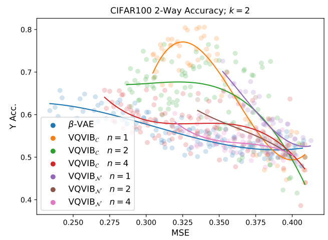
8.2 CIFAR100

We found similar trends in the CIFAR100 to those in the FashionMNIST domain and plotted results in Figures 11 and 12 (for 2-way and 20-way finetuning tasks, respectively). In all experiments, VQ-VIB$_\mathcal{C}$ outperformed both β-VAE and VQ-VIB$_\mathcal{N}$. In the 2-way finetuning example, we again found a peaked curve for VQ-VIB$_\mathcal{C}$ finetuning accuracy as a function of MSE, indicating that tuning to the right complexity level induced the best accuracy. In the more complex 20-way classification task, however, we did not observe this peak.

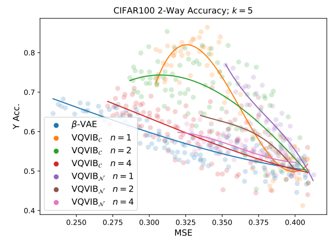
This last result is unsurprising: the 20-way hierarchy in CIFAR100 is less semantically meaningful and likely less obvious in photos than the 2-way task of distinguishing living and non-living things. For example, two of the 20 categories are simply different sorts of vehicles. It would be extremely surprising for VQ-VIB$_\mathcal{C}$ to learn such arbitrary groups automatically while compressing representations. Without learning the right groupings, VQ-VIB$_{\mathcal{C}}$ cannot benefit from learning less complex representations.



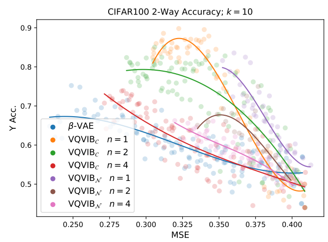
(a) $k = 1$



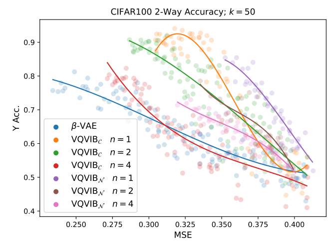
(b) $k = 2$



(c) $k = 5$

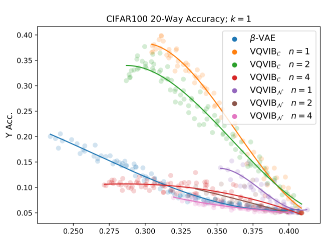


(d) $k = 10$

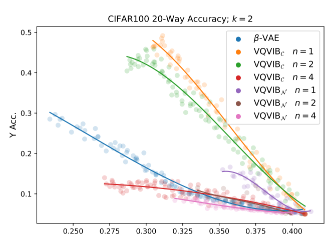


(e) $k = 50$

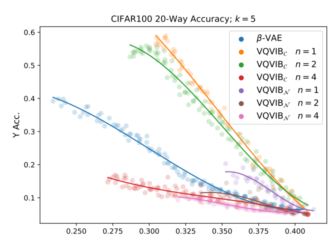
Figure 11: CIFAR100 2-way finetuning results for varying k.



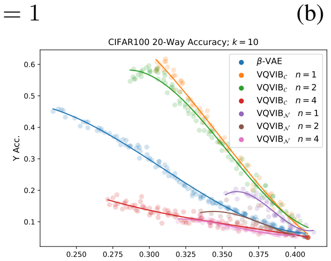
(a) $k = 1$



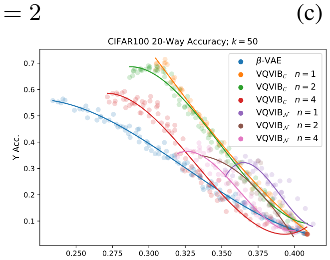
(b) $k = 2$



(c) $k = 5$



(d) $k = 10$



(e) $k = 50$

Figure 12: CIFAR100 20-way finetuning results for varying k.

8.3 iNaturalist

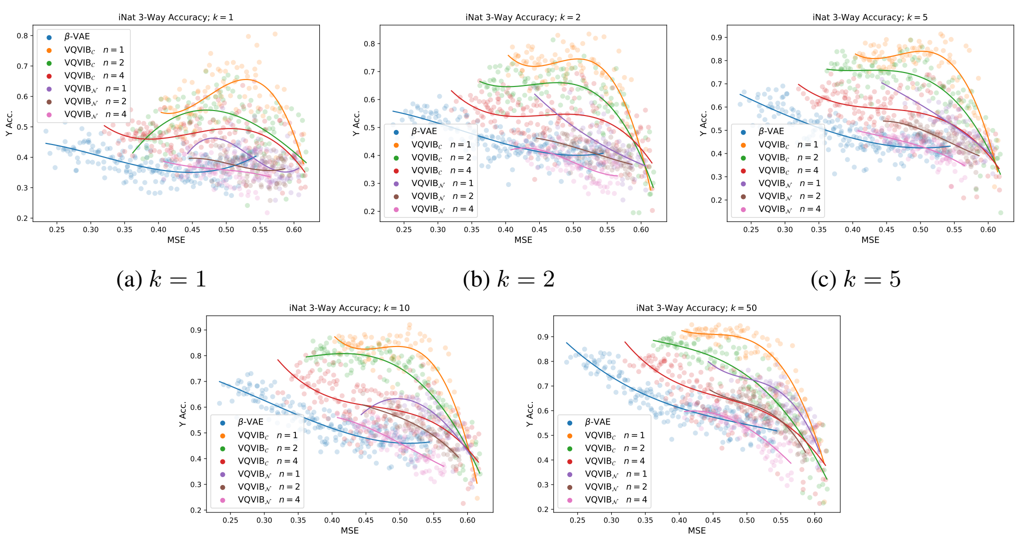
Lastly, we found similar trends in finetuning in the iNat domain, finetuned on a 3-way (Figure 13), 34-way (Figure 14), and 1010-way (Figure 15) finetuning task.

On the 3-way finetuning task (between animals, plants, and fungi), we observed similar peaking behavior as in earlier experiments, indicating yet again the importance of tuning to the right complexity. In addition, as in prior results, we found a similar trend that greater n tended to allow greater complexity (lower MSE) but induced worse finetuning performance. For example, in Figure 13 b, the orange line, corresponding to $n = 1$ stays above and to the right of the green ($n = 2$) and red ($n = 4$) lines. Intuitively, this seems to indicate that the more combinatorial representations, with greater n, were somewhat of a midpoint between the continuous β-VAE representations and the discrete representations used by VQ-VIB$_{\mathcal{C}}$ for $n = 1$.

Results from the 34-way finetuning followed similar patterns as before as well. Just as in CIFAR100 wherein we tested both a 2-way and 20-way finetuning task, this 34-way finetuning task for iNat showed that VQ-VIB$_{\mathcal{C}}$ continued to outperform VQ-VIB$_{\mathcal{N}}$ and β-VAE for more complex finetuning tasks, although the performance gap shrank as k increased.

Most interestingly, perhaps, we conducted yet another iNat finetuning experiment, this time using the 1010 low-level labels that had originally been used during pre-training. As before, we used very small amounts of data in finetuning (e.g., for $k = 1$, only 1 example from each class, so 1010 labeled examples total). Results from those experiments are shown in Figure 15.

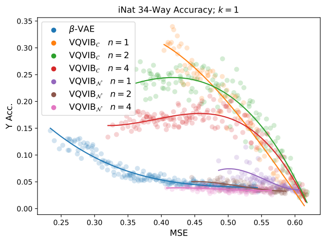
For small k, we again see that VQ-VIB$_{\mathcal{C}}$ outperforms other model types. For larger k, however, we see one of the limitations of VQ-VIB$_{\mathcal{C}}$. Because the discrete encoders learned less complex representations than β-VAEs (as shown by the fact that they never reach lower MSE values), with enough finetuning data, β-VAEs are able to capture distinctions between classes that VQ-VIB$_{\mathcal{C}}$ models cannot. Thus, in the particular case of large amounts of finetuning data and complex finetuning tasks, more complex, continuous encoders continue to outperform our method.



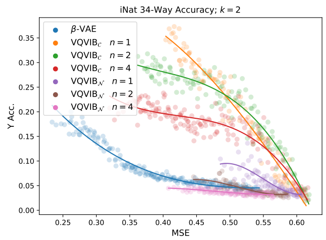
(a) $k = 1$ (b) $k = 2$ (c) $k = 5$

(d) $k = 10$ (e) $k = 50$

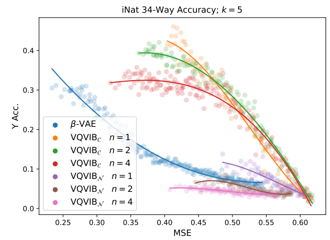
Figure 13: iNat 3-way finetuning results for varying k.



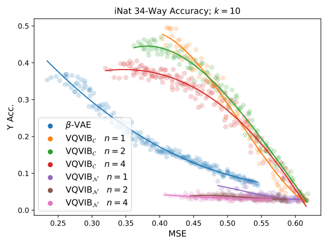
(a) $k = 1$



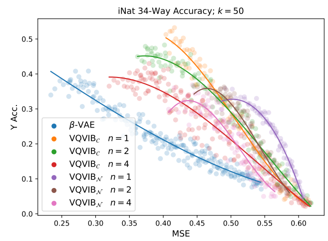
(b) $k = 2$



(c) $k = 5$

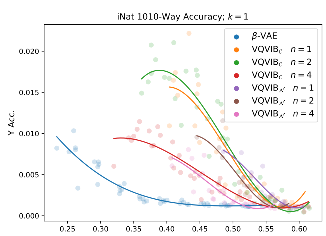


(d) $k = 10$

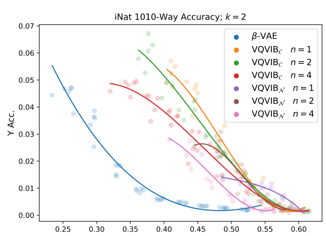


(e) $k = 50$

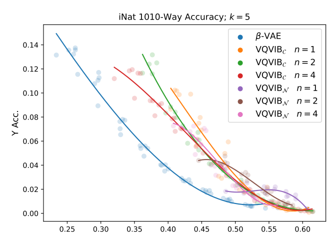
Figure 14: iNat 34-way finetuning results for varying k.



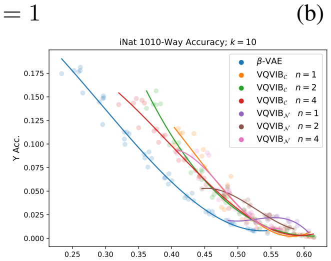
(a) $k = 1$



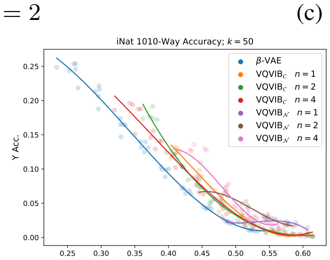
(b) $k = 2$



(c) $k = 5$

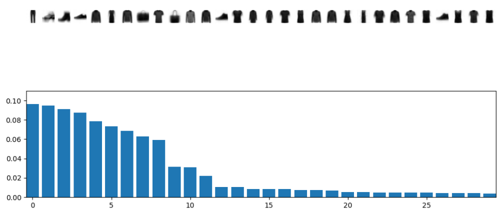


(d) $k = 10$

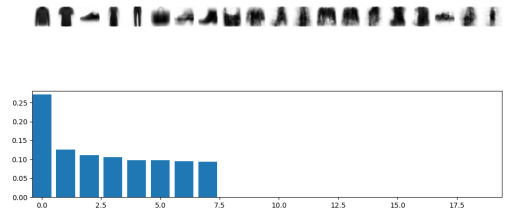


(e) $k = 50$

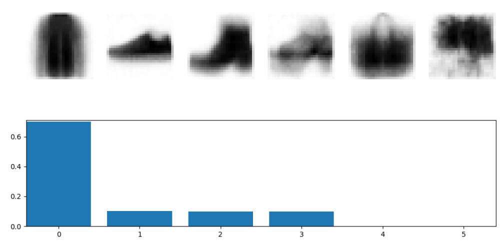
Figure 15: iNat 1010-way finetuning results for varying k.



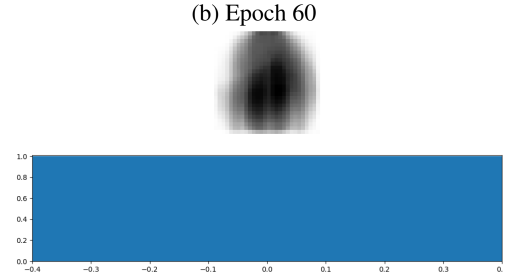
(a) Epoch 40



(b) Epoch 60



(c) Epoch 70



(d) Epoch 199

Figure 16: The evolution of the distribution over prototypes during annealing for VQ-VIB$_{\mathcal{C}}$ in the FashionMNIST domain. In early epochs, VQ-VIB$_{\mathcal{C}}$ uses many prototypes, with a long-tailed distribution. Over the course of annealing the entropy, the probability distribution becomes more concentrated (b and c) before collapsing to a single prototype (d).

9 Prototype Utilization: Further Visualizations

Here, we include some further visualizations that we omitted from the main paper due to space constraints. These visualization primarily illustrate the importance of entropy-regulated representation learning (for VQ-VIB$_{\mathcal{C}}$) vs. complexity-regulated (for VQ-VIB$_{\mathcal{N}}$).

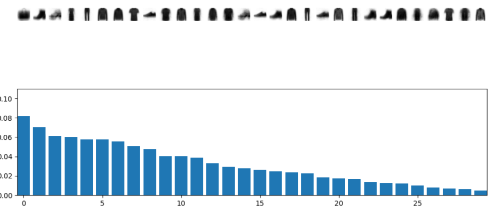
Figures 16 and 17 shows the prototypes for VQ-VIB$_{\mathcal{C}}$ and VQ-VIB$_{\mathcal{N}}$, respectively, in the FashionMNIST domain over the course of training. Each subfigure consists of a top row of decoded prototypes, with associated probabilities (frequency of use measured when passing through images from the test set) below. The 30 most frequent prototypes are visualized, or fewer prototypes if fewer were used.

There is an important trend in Figures 16 and 17: the entropy-based annealing for VQ-VIB$_{\mathcal{C}}$ caused models to use fewer prototypes, while the complexity-based annealing for VQ-VIB$_{\mathcal{N}}$ did not. At epoch 40, just as both methods begin annealing, VQ-VIB$_{\mathcal{C}}$ and VQ-VIB$_{\mathcal{N}}$ use a large number of prototypes, as seen by the long-tailed distributions. Over the course of annealing, however, VQ-VIB$_{\mathcal{C}}$ uses fewer prototypes, and merges images of different classes into the same prototype. Thus, the degenerate encoder at the end of annealing (epoch 199) uses just a single prototype to represent all possible inputs (Figure 16). At the same time, VQ-VIB$_{\mathcal{N}}$, during annealing, does not use fewer prototypes. Rather, the complexity-penalization term seems to induce the model to make the mapping from input to prototype more stochastic (Figure 17). Thus, the degenerate VQ-VIB$_{\mathcal{N}}$ encoder uses many prototypes, each of which is blurry because it could correspond to any input.

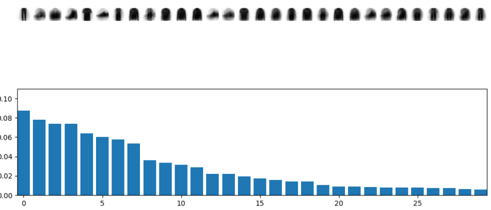
Visualizations of decoded prototypes for the CIFAR100 domain is more challenging. In richer image domains, prototype-based methods often use training examples as prototypes [3, 20, 4], which can make it more difficult to understand when a single prototype represents more than one concept. Nevertheless, by visualizing the distribution over prototypes (without decoding them), we see the same pattern that VQ-VIB$_{\mathcal{C}}$ tends to learn to use fewer prototypes over the course of annealing than VQ-VIB$_{\mathcal{N}}$. Snapshots of the categorical distributions for CIFAR100 are included in Figure 18.

10 Ablation Study: Entropy vs. Complexity

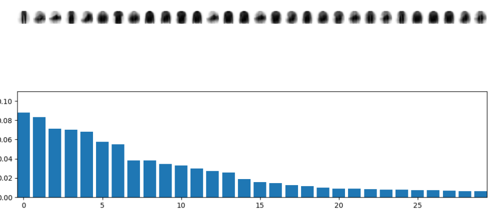
Here, we present results motivating penalizing entropy, as opposed to complexity, in VQ-VIB$_{\mathcal{C}}$. Appendix 9 showed how annealing entropy in VQ-VIB$_{\mathcal{C}}$ caused models to use fewer prototypes, whereas penalizing complexity in VQ-VIB$_{\mathcal{N}}$ did not induce similar reductions in effective codebook



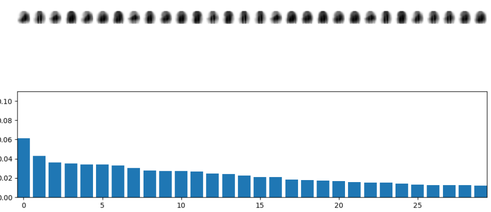
(a) Epoch 40



(b) Epoch 60

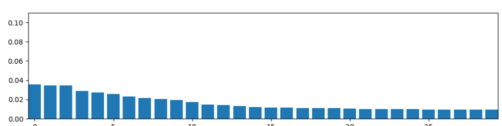


(c) Epoch 70

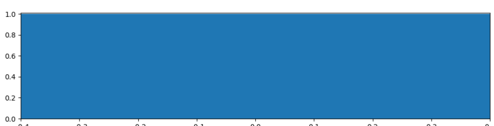


(d) Epoch 199

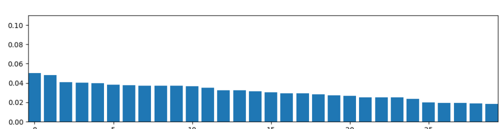
Figure 17: The evolution of the distribution over prototypes during annealing for VQ-VIB$_{\mathcal{N}}$ in the FashionMNIST domain. Unlike annealing entropy for VQ-VIB$_{\mathcal{C}}$, annealing the complexity in VQ-VIB did not result in fewer prototypes being used. Instead, each prototype became blurrier, indicating that each prototype became more likely regardless of class.



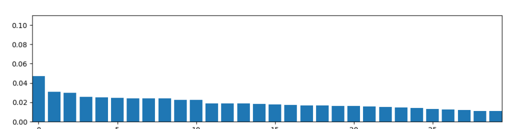
(a) VQ-VIB$_{\mathcal{C}}$ Epoch 50



(b) VQ-VIB$_{\mathcal{C}}$ Epoch 399

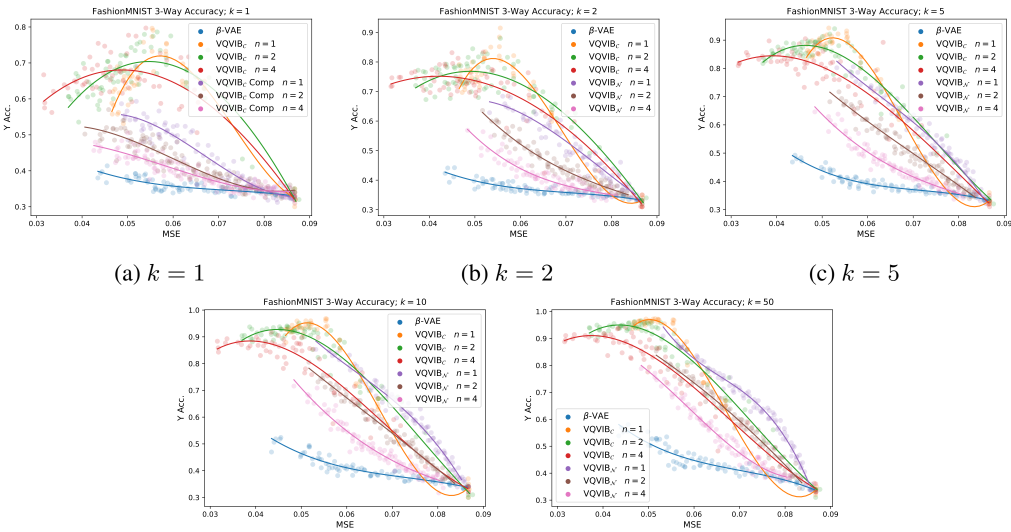


(c) VQ-VIB$_{\mathcal{N}}$ Epoch 50



(d) VQ-VIB$_{\mathcal{N}}$ Epoch 399

Figure 18: Categorical distribution over prototypes while annealing in the CIFAR100 domain at the start of annealing (Epoch 50) and at the end (Epoch 399) for VQ-VIB$_{\mathcal{C}}$ (top row) and VQ-VIB $_{\mathcal{N}}$ (bottom row). Annealing the entropy in VQ-VIB$_{\mathcal{C}}$ caused the model to use fewer prototypes (note the degenerate categorical distribution over only one prototype at Epoch 399), whereas annealing complexity for VQ-VIB$_{\mathcal{N}}$ did not cause similar concentration.



(a) $k = 1$ (b) $k = 2$ (c) $k = 5$

(d) $k = 10$ (e) $k = 50$

Figure 19: FashionMNIST finetuning results for varying k, comparing annealing by entropy (VQ-VIB$_\mathcal{C}$) and annealing by complexity (VQ-VIB$_\mathcal{C}$ Comp.). Annealing by complexity resulted in worse finetuning performance.

size. Further experiments corroborate our findings that penalizing entropy was the key to this difference in behavior.

We trained VQ-VIB$_\mathcal{C}$ agents on the FashionMNIST task, using the same pre-training and finetuning procedures as in the main paper, with the only difference being that we annealed the complexity of representations instead of the entropy. Results from finetuning such models are included in Figure 19.

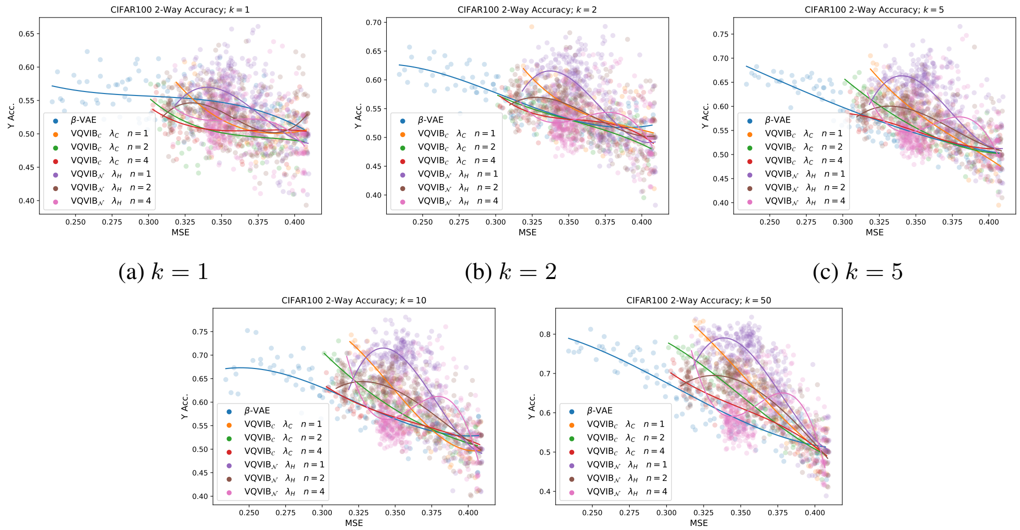
Figure 19 shows that annealing by entropy, as opposed to complexity, was the key factor in improving VQ-VIB$_\mathcal{C}$ finetuning performance. The difference in performance when penalizing entropy vs. complexity closely matches the difference in performance between VQ-VIB$_\mathcal{C}$ and VQ-VIB$_\mathcal{N}$ examined in the main paper. Thus, the entropy-regularization term seems to explain much of the difference between VQ-VIB$_\mathcal{C}$ and VQ-VIB$_\mathcal{N}$.

In subsequent experiments in the CIFAR100 and iNat domains, therefore, we tested whether penalizing the estimated entropy of VQ-VIB$_\mathcal{N}$ models matched VQ-VIB$_\mathcal{C}$ results from the main paper. We note that Tucker et al. [27] advocate for a small positive λ_H to penalize entropy, but the authors also acknowledge that exactly computing this entropy is impossible given the VQ-VIB$_\mathcal{N}$ architecture.

Finetuning results for CIFAR100 (on the 2-way and 20-way finetuning tasks) and iNat (on the 3-way and 34-way finetuning tasks), for VQ-VIB$_\mathcal{C}$ trained by varying λ_C and VQ-VIB$_\mathcal{N}$ trained by varying λ_H, are included in Figures 20, 21, 22, and 23. Several important trends emerge from viewing these plots, especially compared to results from our main paper for VQ-VIB$_\mathcal{C}$ controlled via λ_H.

First, finetuning performance is noisier using these models compared to results from the main text. This likely arises, for VQ-VIB$_\mathcal{N}$ models, because increasing λ_H failed to consistently reduce the number of discrete representations used. Thus, for a given MSE value, different models used different numbers of representations, and therefore exhibited different finetuning performance.

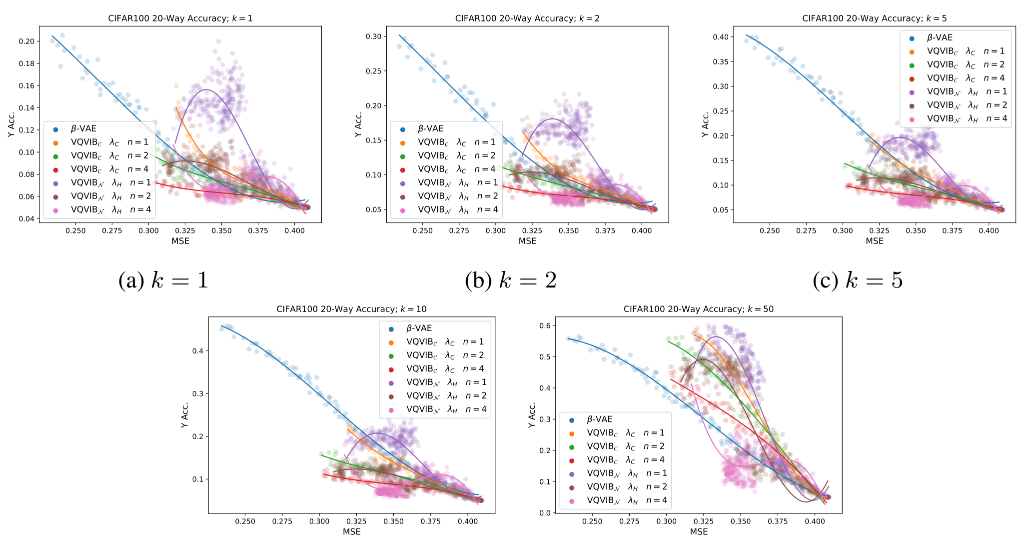
Second, varying λ_H, instead of λ_C, seemed to somewhat improve VQ-VIB$_\mathcal{N}$ performance, but not as much as when varying λ_H for VQ-VIB$_\mathcal{C}$, as presented in our main paper. For example, consider Figure 21 a. The best-performing model, VQ-VIB$_\mathcal{N}, n = 1$, peaks at finetuning accuracy of approximately 0.16, outperforming VQ-VIB$_\mathcal{C}$ models when varying λ_C. However, in Figure 12 a, we found that VQ-VIB$_\mathcal{C}$ models in the exact same setting achieved a mean accuracy of approximately 0.38: more than double the VQ-VIB$_\mathcal{N}$ performance. Thus, varying λ_H seemed to improve VQ-VIB$_\mathcal{N}$ performance somewhat, but VQ-VIB$_\mathcal{C}$ better supports penalizing entropy, and therefore achieves higher performance.



(a) $k = 1$ (b) $k = 2$ (c) $k = 5$

(d) $k = 10$ (e) $k = 50$

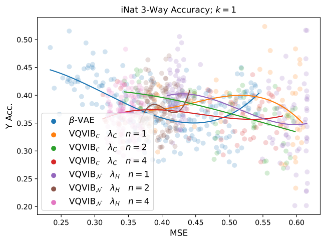
Figure 20: Ablation study results for the CIFAR100 2-way finetuning task. Tuning λ_C for VQ-VIB$_\mathcal{C}$ or λ_H for VQ-VIB$_\mathcal{N}$ led to worse results than in our main paper.



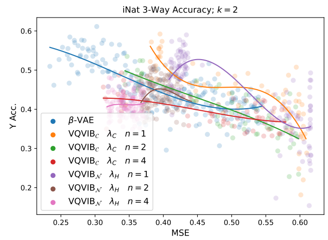
(a) $k = 1$ (b) $k = 2$ (c) $k = 5$

(d) $k = 10$ (e) $k = 50$

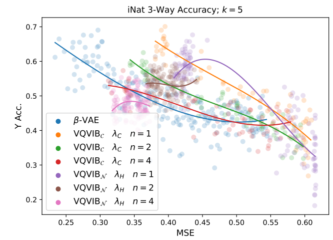
Figure 21: Ablation study results for the CIFAR100 20-way finetuning task. Tuning λ_C for VQ-VIB$_\mathcal{C}$ or λ_H for VQ-VIB$_\mathcal{N}$ led to worse results than in our main paper.



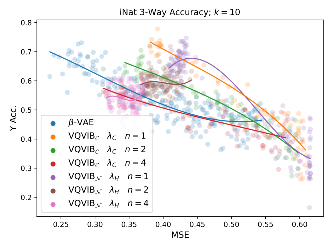
(a) $k = 1$



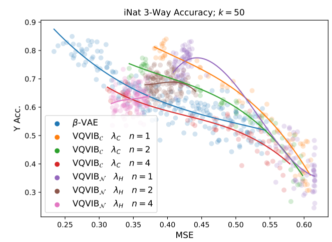
(b) $k = 2$



(c) $k = 5$

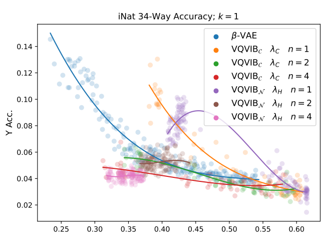


(d) $k = 10$

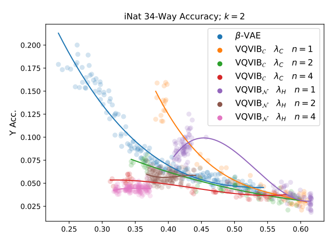


(e) $k = 50$

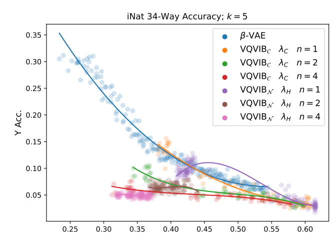
Figure 22: Ablation study results for the iNat 3-way finetuning task. Tuning λ_C for VQ-VIB$_\mathcal{C}$ or λ_H for VQ-VIB$_\mathcal{N}$ led to worse results than in our main paper.



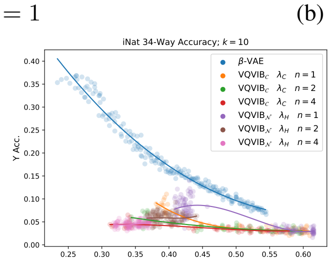
(a) $k = 1$



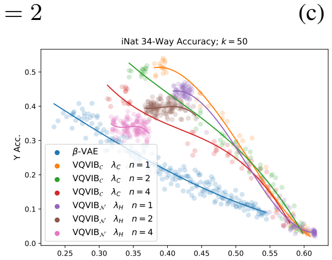
(b) $k = 2$



(c) $k = 5$



(d) $k = 10$



(e) $k = 50$

Figure 23: Ablation study results for the iNat 34-way finetuning task. Tuning λ_C for VQ-VIB$_\mathcal{C}$ or λ_H for VQ-VIB$_\mathcal{N}$ led to worse results than in our main paper.

Third, varying λ_C, instead of λ_H, for VQ-VIB$_\mathcal{C}$ worsened finetuning performance. Once again, by comparing finetuning performance for VQ-VIB$_\mathcal{C}$ models in Figure 21 a (achieving a maximum accuracy around 0.14), to results from our main paper, we note the importance of penalizing the entropy of representations.

Thus, in general these ablation studies support many of the design decisions made in the main paper.

1. Varying λ_H, instead of λ_C for VQ-VIB$_\mathcal{C}$ improves finetuning performance by decreasing the number of discrete representations used.
2. VQ-VIB$_\mathcal{N}$ benefits somewhat from penalizing entropy, but because it is architecturally unable to support exact calculations of entropy, we were unable to match VQ-VIB$_\mathcal{N}$ performance.

It is certainly possible that some optimal combination of λ_H and λ_C might further improve VQ-VIB$_\mathcal{N}$ or VQ-VIB$_\mathcal{C}$ performance; initial explorations of such combinations with fixed λ_H values while annealing λ_C did not yield obvious results. Most importantly, our current findings are enough to indicate that controlling the entropy of discrete representations appears important for data-efficient finetuning.

11 User Study

In the subsequent pages, we have included the exact pdf document shared with participants of the user study, edited only to preserve anonymity during peer-review. Following standard user study procedure, we initial briefed users by telling them how long the study was and that they were free to leave at anytime. Demographic information was collected in person. After the study, users were debriefed and given the email of the study designer to contact if they had any questions.

Block 1

Brief

You are free to leave this study at any time.

It will take less than 5mins.

Thank you for your participation!

Block 2

Introduction

A factory has created a robot that is good at sorting clothes into these 10 categories

- T-shirt
- Trouser
- Pullover
- Dress
- Coat
- Sandal
- Shirt
- Sneaker
- Bag
- Ankleboot

See below for examples of these 10 categories

Block 3

Introduction

However, in your home you don’t care about these 10 categories.

You only sort your clothes into these 3 categories:

- Shirts
- Shoes
- Trousers/Bags

So, we want to teach the robot to be more general and only care about these three high level categories.

Block 10

Introduction

In doing so, we are going to re-train ***two different robots*** over a period of time.

At a certain point during this re-training, they will be able to sort these clothes into these three new categories best.

Your task is to pick the point in which they are going to categorize these three high-level categories best.

Block 4

Introduction

You will have to answer one question about each robot.

First, you will be shown one robot which only communicates with its error score. *The lower the score, the better the robot is at classifying the 10 categories*.

Secondly, you will be shown another robot. This robot has learned to communicate what it has learned though visualizing the most important images it uses for categorizing the 3 high-level concepts. You have to decide which visualization corresponds to the robot being able to categorize the three new high-level categories best.

Generally, the more images the second robot is using, the less general it will be, and the worse it will do at categorizing the 3 new high-level categories.

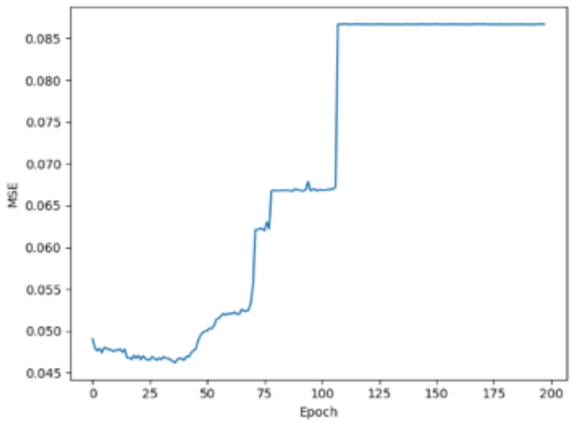
Block 5

Robot 1

Here is an example of the first robot communicating its error scores over training.

You will have to pick a score from 3 sampled options you think will perform best on your desired task of learning the 3 clothing concepts. ***A lower score is typically seen as better (which corresponds to a lower point on the blue curve below).***

Your task is to pick the point which you think will perform best at representing these three categories.



Block 6

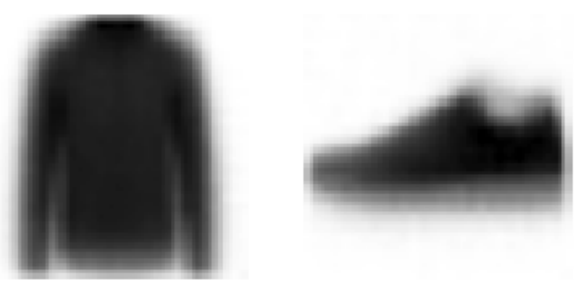
Robot 2

Then you will be shown visualizations like this, which is second robot's method of communicating its learned concepts.

If the visualization shows that (1) the robot is not using many images, and (2) they roughly represent the three high-level categories (shirts, shoes, trousers/bags), then the robot should perform well in the task of sorting clothes into these three categories.

Your task is to pick the visualization which you think will perform best at sorting the clothes into these 3 high-level categories.

For example, here is one robot's visualisation which is perhaps too general, as there is not even three categories being used.



And here is one which could be too specific, because there is a lot of detail.

Important: We want the robot to learn the three high-level concepts, but no more or no less!

Block 11

Click next to begin the study

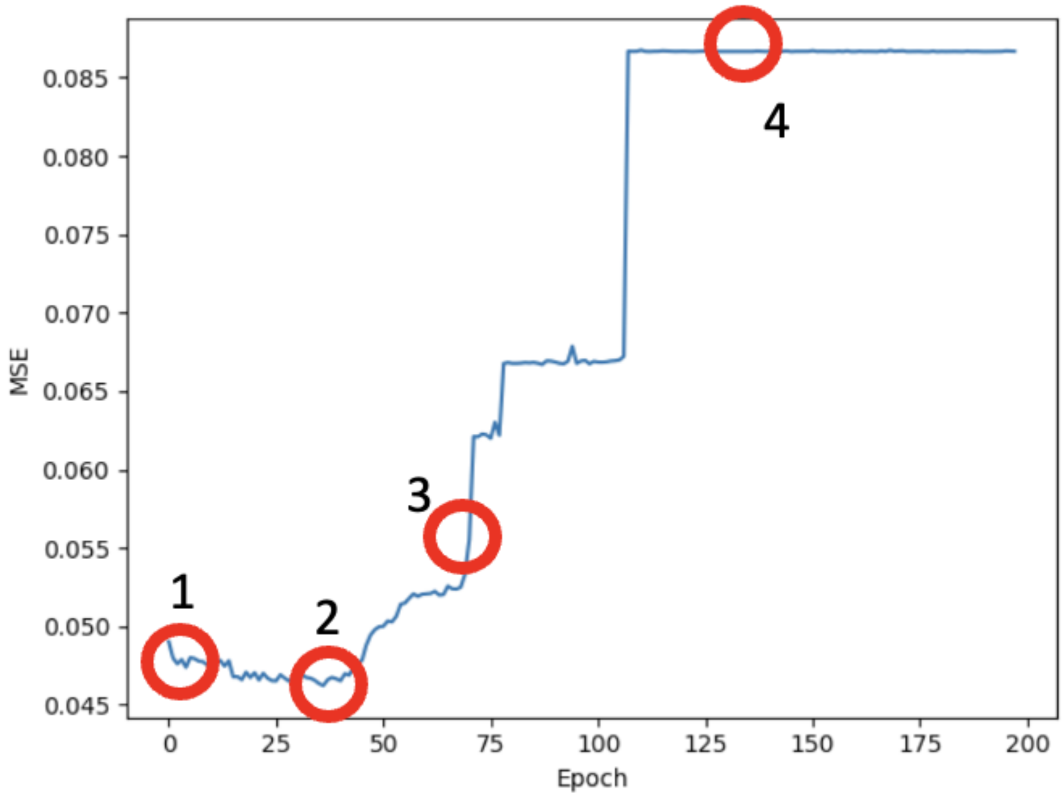
Block 7

Question 1:

Each circled region represents the robot trained with a different error score. Remember this error is on the 10-way clothing classification task.

Choose the point that you think the robot is going to perform the best at classifying the high-level categories.

- Shirts
- Shoes
- Trousers/Bags



Choose the robot now:

- 1
- 2
- 3
- 4

Block 8

Question 2:

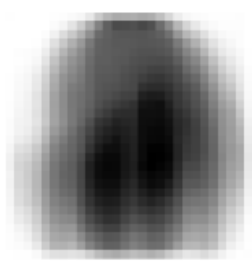
Each visualization represents the robot trained with different high-level categories.

Choose the visualization that you think will help the robot perform best at sorting your clothes into these three high-level categories:

- Shirts
- Shoes
- Trousers/Bags

The first option is

The second option is



The Third Option is

The Fourth option is

719

Click to write the question text

- First option
- Second option
- Third option
- Fourth option

Block 9

Debrief Page

Thank you for your participation, this study was designed to evaluate people's ability to use an explainable artificial intelligence system to help fine-tuning towards down-stream tasks.

if you have any questions please contact [redacted]

Click next to finish

[redacted]