Few-shot Learning for Free by Modelling Global Class Structure

Xuechen Li,* Will Grathwohl,* Eleni Triantafillou,* David Duvenaud, Richard Zemel
University of Toronto & Vector Institute

Abstract

Most state-of-the-art approaches to few-shot classification involve some form of meta-learning. We present a simpler approach: learning a generative model that captures the global structure of classes that is capable of few-shot learning 'for free'. It matches the state-of-the-art on few-shot classification on Omniglot and can retaining knowledge of old classes while quickly learning that of new ones, unlike previous meta-learners trained end-to-end solely to few-shot classify. Moreover, our model is capable of few-shot generation: generating examples of new classes given only few examples of known classes. We present compelling results on these three tasks on the Omniglot dataset.

1 Introduction

Few-shot classification is the problem of learning to distinguish between a set of new classes of which only a few examples are available. In this setting, a training set is available but it contains examples of a completely disjoint set of classes than those that the model is required to 'quickly learn' at test time. This problem has received significant attention and specialized meta-learning models have emerged to tackle it [1, 2, 3, 4]. These models leverage the set of training classes by constructing from them a set of *tasks* that resemble in structure the few-shot classification tasks they will ultimately encounter. After learning to few-shot learn via a sequence of these tasks, the learned algorithm is directly applied to solve each test task too. This procedure is referred to as *episodic training*, and it has been believed to be responsible for most recent success in this problem [2].

We pose the question: Can we instead obtain a model that is amenable to few-shot learning 'for free' by having it learn the overall structure of the training classes and examples? We hypothesize that a model that understands this structure is also able to reason about where new classes can be placed in this global picture. This would put forward a very different approach to the problem than the episodic procedure outlined above. Indeed, we present a new method for this based on a variational autoencoder [5, 6] (VAE) whose latent space is defined by a mixture of Gaussians, with one component per training class. Members of a class have latent variables z drawn from this Gaussian cluster and the data x is drawn from a standard neural-network decoder given z.

In this model, new classes can be incorporated by inferring the cluster parameters of the distribution in latent space which generated the data. Classification is performed by inferring which distribution a given example's latents were most likely to be drawn from. In this way, new classes can be incorporated into our model without degrading predictive performance on previously seen classes, an ability which competing methods for few-shot classification lack. Learning on new tasks while retaining the knowledge on old tasks can be viewed as an instance of continual learning [7, 8]. Additionally, the generative nature of our model enables us to synthesize novel examples of potentially new classes with high fidelity, given only a few examples. This is all achieved without any modification to the model or training procedure.

*Equal contribution, order chosen randomly.

2nd Workshop on Meta-Learning at NeurIPS 2018, Montréal, Canada.

2 Our Generative Model

Our model was inspired by powerful semi-supervised generative models that also operate in a scenario with little labeled data [9], but we modify them to tackle few-shot learning. In particular, existing models which use a classifier $q(y|x)$ that assumes all classes are known a-priori [9, 10] are not well suited for generalization to unseen classes. Moreover, the continuous latent code z is usually designed specifically to encode information unrelated to class. Instead, we augment our generative model with the capability of reasoning about different classes. Our continuous latent code z is drawn conditionally based on the class label y. The data x is drawn conditionally from z. This allows us to reason about the distribution over classes and the relationship between them at the same time as we reason about how the data is generated. This leads to a generative model which factorizes as:

$$p(x, y, z) = p(x|z)p(z|y)p(y). \tag{1}$$

We choose to model $p(z, y)$ with a Gaussian mixture where each class defines a unique component in the mixture. This requires us to augment our generative model with the locations of the mixture components η. Thus, we obtain the complete generative model:

$$p(x, y, z, \eta) = \prod_{i=1}^{D} p(x_i|z_i)p(z_i|\eta, y_i)p(y_i) \prod_{c=1}^{|\mathcal{C}|} p(\eta_c) \tag{2}$$

$$p(x_i|z_i) = \text{neural\_net\_likelihood}(z_i; \theta) \tag{3}$$

$$p(z_i|\eta, y_i) = \mathcal{N}(z_i; \eta_{y_i}, \sigma^2) \tag{4}$$

$$p(y) = \text{Categorical}\left(\frac{1}{|\mathcal{C}|}\right) \tag{5}$$

$$p(\eta_c) = \mathcal{N}(\eta_c|0, I) \tag{6}$$

where σ^2 and θ are the parameters of the generative model, D is the number of data points, and $\mathcal{C}$ is the set of training classes. The conditional data generative distribution (Equation 3) is some distribution parameterized by a neural network with parameters θ. A plate diagram of this generative model can be found in Appendix C.

We propose to train a VAE using this generative model. We define our inference model as:

$$q(z, \eta, y|x) = \prod_{i=1}^{D} q(z_i|x_i)q(y_i|z_i, \eta) \prod_{c=1}^{|\mathcal{C}|} q(\eta_c) \tag{7}$$

$$q(z_i|x_i) = \mathcal{N}(\mu(x_i; \phi), \sigma^2(x_i; \phi)) \tag{8}$$

$$q(\eta_c) = \mathcal{N}(\eta_c|\mu_c, \beta_c^2) \tag{9}$$

$$q(y_i|z_i, \eta) = p(y_i|z_i, \eta) = \frac{p(z_i|y_i, \eta)}{\sum_{j=1}^{|\mathcal{C}|} p(z_i|y_j, \eta)} \tag{10}$$

where β_c, μ_c and ϕ are the parameters of the inference model and $\mu(\cdot; \theta)$ and $\sigma^2(\cdot; \theta)$ are neural networks. Equation 10 comes from the fact that since $p(z, y, \eta)$ is a Gaussian mixture, the posterior $p(y|z, \eta)$ can be computed analytically.

We train our model by optimizing

$$\begin{aligned}\mathcal{L}_\eta(x, y) = \mathbb{E}_{q(\eta)q(z|x)}[&\log p(\eta) + \log p(y) + \log p(z|\eta_y) + \log p_\theta(x|z) - \\ &\log q_\phi(\eta) - \log q_\phi(z|x)].\end{aligned} \tag{11}$$

We also add a classification loss to the supervised objective, in a fashion similar to [9]. Given x and y, we sample $z \sim q_\phi(z|x)$ and $\eta \sim q(\eta)$ and optimize the objective :

$$\mathcal{F}_\eta(x, y) = \mathbb{E}_{q(\eta)q(z|x)}[\log q_\phi(y|z, \eta)]. \tag{12}$$

The overall objective that our model maximizes is then:

$$\mathcal{L}_\eta(x, y) + \alpha \cdot \mathcal{F}_\eta(x, y; \eta) \tag{13}$$

where α is a hyper-parameter, that controls the weight of the discriminative loss.

This training procedure differs considerably from most state-of-the-art few-shot classification approaches which train on episodes sampled from the training set [3, 11, 12, 1, 4, 2]. This difference allows our model to infer class relationships in a global fashion which is key to our success in learning about new classes without hurting performance on previously seen classes. We now explain how this model can tackle three types of few-shot learning tasks.

3 Few-shot Learning with our Generative Model

Few-shot Classification Few-shot classification performance is assessed through a series of test episodes, each presenting an instance of a classification task for N classes sampled from the test set. Specifically, each k-shot N-way episode contains a *support set* $\mathcal{S}$ which is comprised of k examples of each of the N new classes, and a *query set* $\mathcal{Q}$ of additional examples of these new classes. The task is to correctly label each query example (into one of the N classes), after using only the support set to learn these new classes.

For solving the task presented in each test episode, we again perform variational inference. We initialize N new components to our mixture model, since we now know that our data belongs to N different classes. We optimize their parameters using the support set $\mathcal{S}$ while keeping the rest of the model fixed. This has a number of benefits. First, we do not train the neural network in this step, so we greatly reduce the risk of over-fitting to the small support set. Further, we do not retrain the cluster centers η for the training classes, allowing our model to maintain its ability to recognize examples from these classes. We are in fact free to utilize these previously inferred cluster centers to inform our new inference where in the latent space *not* to place the new centers.

In particular, let η^* denote the N new cluster centers that are initialized for a given episode. Denote by η^+ the augmented set of η's obtained by 'concatenating' η^* and η, totalling $N + |\mathcal{C}|$ clusters.

We optimize with respect to the parameters of η^* using the support set with the following objective:

$$\max_{\eta^*} \sum_{(x,y)\in\mathcal{S}} \mathcal{L}_{\eta^*}(x,y) + \alpha \cdot \mathcal{F}_{\eta^+}(x,y) \tag{14}$$

where $\mathcal{S}$ denotes the support set, and $\mathcal{L}$ and $\mathcal{F}$ denote the ELBO and classification objective for the labeled data, as defined in the previous section.

Then, once η^* is learned on the support set, our model infers a class label for each query out of the N possible labels for this episode. Specifically, let x_q denote a query example. The model labels it as:

$$y_q = \arg\max_y \mathbb{E}_{q(\eta^*)q(z_q|x_q)}[\log p(y|z_q, \eta^*)] \tag{15}$$

Few-shot Integration This is the task of *expanding* a classifier's repertoire of classes by adding into it a set of new classes for which only a few examples are available, also studied in [13].

Evaluation is performed in a series of integration episodes. In a k-shot N-way few-shot integration test episode, the support set contains k examples of each of the N *new* classes sampled from the test set (and no examples of the training classes). The query set, however, which the model is tasked to label, is split in half between examples of the N new classes as well as examples of another set of N randomly sampled *training* classes. The query examples that belong to the training classes are drawn from a held-out pool, ruling out the possibility of the model seeing a query example that it had originally been trained on. Importantly, we provide no information to the model about which query examples come from the old or new classes, nor which old classes the query examples are drawn from. The model is therefore faced with a $(|\mathcal{C}| + N)$-way classification problem for each (old or new) query point. In the literature the training classes and test classes are often referred to as *base* and *novel* classes, respectively (Omniglot has 2880 base classes!).

The support set contains examples only from new classes so the model cannot refresh its memory of old classes by retraining on examples of them. In this case, we perform the exact same procedure for training as for the standard few-shot classification task, using the objective in Equation 14.

The method for labeling each query only differs from that of standard few-shot classification in Equation 15 in that all new and old clusters η^+ are now considered as potential targets, since query points can belong to either of these:

$$y_q = \arg\max_y \mathbb{E}_{q(\eta^+)q(z_q|x_q)}[\log p(y|z_q, \eta^+)] \tag{16}$$

Few-Shot Generation Our model naturally possess the ability to generate novel examples of known classes, or novel classes altogether. Performing these generation tasks allows us to explore the distribution that our model has learned over the data as well as the distribution over the global class structure in the dataset. We can inspect our model's ability to "imagine" new classes by first sampling $\eta \sim p(\eta)$ and then sample $z \sim p(z|\eta)$. Then we sample $x \sim p(x|z)$ to obtain our data. Samples of imagined classes can be seen in Figure 1. Our model can also perform few-shot generation. In this setting we are given n examples of a previously unseen class $\hat{y}$. We then infer $\eta_{\hat{y}}$ with the same procedure we use for few-shot classification defined in Section 3. We then sample $z \sim p(z|\eta_{\hat{y}})$ and sample $x \sim p(x|z)$. Few-shot samples can be seen in Figures 2 and 3.

4 Experiments

We present experiments on the three tasks outlined above, on the Omniglot dataset [14]. We refer the reader to the Appendix for full details regarding the datasets, optimization, architectures, and hyper-parameters used. In Table 1 we show that our model matches the state-of-the-art on the standard few-shot classification task, despite its simpler training procedure. To obtain a baseline to compare against for few-shot integration, we modify Prototypical Networks [1] to maintain a global prototype for each class instead of temporary episodic prototypes. As can be seen from Table 2, our method outperforms this baseline by a large margin. Details of this baseline can be seen in Appendix D.2. Few-shot generation results can be seen in Appendix B.

Model	1-shot 5-way	5-shot 5-way	1-shot 20-way	5-shot 20-way
Matching Networks [2]	97.9	98.9	93.8	98.5
Neural Statistician [15]	98.1	99.5	93.2	98.1
MAML [3]	98.7	**99.9**	95.8	98.9
Prototypical Networks [1]	98.8	99.7	96.0	98.9
GNN [16]	**99.2**	99.7	**97.4**	99.0
Prototypical Networks (our encoder)	98.6	99.4	95.5	98.1
Ours	98.9	99.6	96.7	**99.2**

Table 1: Few-shot classification results on Omniglot [14]. Prototypical Networks (our encoder) indicates a baseline of the Prototypical Networks algorithm using an identical encoder to ours.

Model	5-way 1-shot			5-way 5-shot		
	Novel	Base	Both	Novel	Base	Both
PN	$7.4 \pm .6$	$55.0 \pm .1$	$31.2 \pm .7$	$4.2 \pm .4$	$62.3 \pm .1$	$33.3 \pm .6$
Ours	$\mathbf{60.9 \pm .1}$	$\mathbf{87.1 \pm .6}$	$\mathbf{74.0 \pm .6}$	$\mathbf{86.0 \pm .7}$	$\mathbf{87.1 \pm .7}$	$\mathbf{86.5 \pm .5}$
	20-way 1-shot			20-way 5-shot		
	Novel	Base	Both	Novel	Base	Both
PN	$.1 \pm .01$	$57.9 \pm .6$	$29.0 \pm .3$	$1.2 \pm .1$	$51.0 \pm .6$	$26.1 \pm .3$
Ours	$\mathbf{60.1 \pm .5}$	$\mathbf{87.4 \pm .3}$	$\mathbf{74.1 \pm .3}$	$\mathbf{86.0 \pm .3}$	$\mathbf{87.4 \pm .3}$	$\mathbf{86.7 \pm .2}$

Table 2: Few-shot integration results, averaged over 1000 test episodes and reported with 95% confidence intervals. PN refers to the prototypical network baseline defined in Appendix D.2. Novel, base, both is performance on new classes, old classes, and both combined, respectively.

5 Conclusion and Further Work

In this work we have presented a generative model which we demonstrate is a competent few-shot learner without being trained specifically for that purpose. Without any modification, our model can also learn quickly about new classes without degrading predictive performance on previously seen classes. Models which preform comparably to ours on few-shot classification fail completely at this task. As future work, we are confident our results could be improved by incorporating multiple levels of stochastic variables [17, 18], using more advanced inference techniques such as normalizing flows [6], or using more advanced neural network architectures [19].

References

[1] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090, 2017.

[2] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[4] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[6] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[7] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[8] Mark B Ring. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

[9] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[10] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

[11] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pages 2252–2262, 2017.

[12] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[13] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. *arXiv preprint arXiv:1804.09458*, 2018.

[14] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

[15] Harrison Edwards and Amos Storkey. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

[16] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[17] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

[18] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[19] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[20] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Luke Hewitt, Andrea Gane, Tommi Jaakkola, and Joshua B Tenenbaum. The variational homoencoder: Learning to infer high-capacity generative models from few examples. 2018.

[23] Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.

Appendix A Experimental Details

A.1 Model Architectures

Our encoder (equation 8) and decoder (equation 3) are parameterized by neural networks. We use neural networks with residual connections for both encoder and decoder and use an autoregressive decoder. Our encoder, and decoder are exactly the same as the MNIST model used in [20].

We note that the encoder and decoder used in our experiments is much more powerful than those used in competing methods [3, 1, 2, 15]. For this reason we re-implement the strongest method, Prototypical Networks [1], using our PixelVAE encoder and find that performance drops slightly compared to the results with their original network architecture. Thus, we believe our improved results are not simply caused by an enhanced encoder architecture.

A.2 Data Likelihood Model

The Omniglot dataset contains 28×28 grey-scale images whose pixels take on integer values in the range $[0, 255]$. The pixel values are approximately binary, so it is common when training generative models on this dataset to scale to the range $[0, 1]$ and threshold the data to $\{0, 1\}$. Then a Bernoulli distribution is used to model the pixel values.

When running the Prototypical Network baseline with our PixelVAE encoder, we experimented with both binary and continuous inputs and found the baseline to perform slightly better with continuous inputs (the results presented in the second-to-last line of Table 1).

We experimented with using a 1-out-of-256 Categorical distribution to model the pixel values as well but found this to decrease convergence speed and without improving performance.

A.3 Training Procedure

Our models are trained using the Adam [21] optimizer with a fixed learning rate of .001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. For all models trained, we use mini-batches of size 64.

We monitor training with the standard validation set every 1000 training iterations, and pick the model with the best performance after 1500 epochs. The per-cluster variance in the prior distribution was chosen among the values: 1, 0.1, and 0.01. The weight for the classification regularization loss was chosen among the values: $0, 0.1, 1, 10$, and 100. Validation was specifically done on the most difficult 20 way 1 shot task for the regular few-shot classification task.

For the few-shot integration and generation tasks, we reused the best set of hyper-parameters for the few-shot classification task.

A.4 Evaluation Procedure

We follow [1] on evaluating performance for few-shot classification on the Omniglot dataset, i.e. we aggregate performance over 1000 test episodes and report the mean accuracy as well as the 95% confidence interval. For each episode, we optimize the usual loss as in eq. 13 with respect to only the variational distribution (mean and variance of Gaussian in our case) for the new clusters using the Adam optimizer with learning rate 0.01, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ for 200 iterations.

For few-shot integration, we need to create a new split of held-out **examples** of each **training** class. This is required so that the examples of the training classes that appear in the query set of the few-shot integration episodes are not examples that were seen at training (though other examples of the same class were seen at training). This ensures that we are fairly measuring how well the network can retain its knowledge of training classes, without having somehow memorized specific examples of those classes. To implement this split, we hold out 2 drawers' images. In particular, in Omniglot there are 20 drawers each of which drew each character once. By holding out two drawers we are holding out 2 images out of the total 20 images of each class.

Unlike [13] we never train our model specifically for the task of integration, but we adapt the same setup as them for evaluation. There, each test episode's support set only contains examples of the N novel classes as usual. The query set is balanced between examples of the N novel classes and of another set of N training classes. The ones coming from the training classes come from a held-out pool of them as mentioned earlier.

As before, for our final results we again sample 1000 test episodes and report the mean accuracy as well as the 95% confidence interval.

Appendix B Few-shot Generation Results

We present results on zero-shot, one-shot, and five-shot generation. Samples can be found in Figures 1, 2, 3, respectively. We obtain results similar in quality to those of [22, 23] while having a significantly simpler generative model and training procedure.

Appendix C Generative Model Plate Diagram

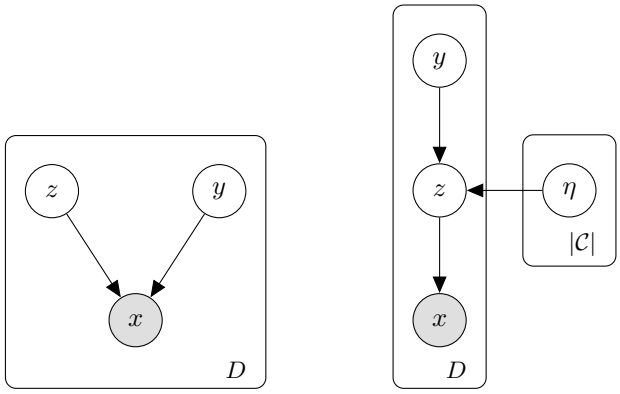

Figure 4: Plate diagrams of semi-supervised models. **Left**: M2 model from [9]. **Right**: Our proposed model. D denotes the number of examples in the dataset, and $\mathcal{C}$ denotes the set of classes.

Appendix D Baselines

D.1 Prototypical Networks with Our Encoder

Our training procedure closely echos that of [1]. We train using the same "shot" as we will use in testing but we use a larger "way," set to 60 as in [1]. We train using episodes with 5 query points per class. The only difference in our training procedure (other than the encoder architecture) is that we decay the learning rate every 10,000 iterations (instead of 2,000) since our larger encoder requires more iterations to train.

D.2 Prototypical Networks for Few-shot Integration

Few shot integration is a new task so To demonstrate the importance of explicitly storing information about previously inferred classes to inform predictions about new classes, we adapt a pre-trained Prototypical Network [1] to this task. To perform few-shot integration with a prototypical network we compute prototypes for each of the classes in the training set of Omniglot by computing the mean embedding of all examples of each training class. Then when given a test episode, we compute N new prototypes from the support set and add them to the prototypes from the training classes. Then we perform the $(N + |\mathcal{C}|)$-way classification via nearest neighbors. Performance of this baseline can be seen in Table 2.

While the accuracy numbers of this baseline appear small, we note that the model is being asked to perform 2880 + $|\mathcal{C}|$ way classification, so even 1% accuracy is considerably better than random chance. We also note that, when the classification task is restricted only to the $|\mathcal{C}|$ new classes, this baseline becomes standard Prototypical networks.

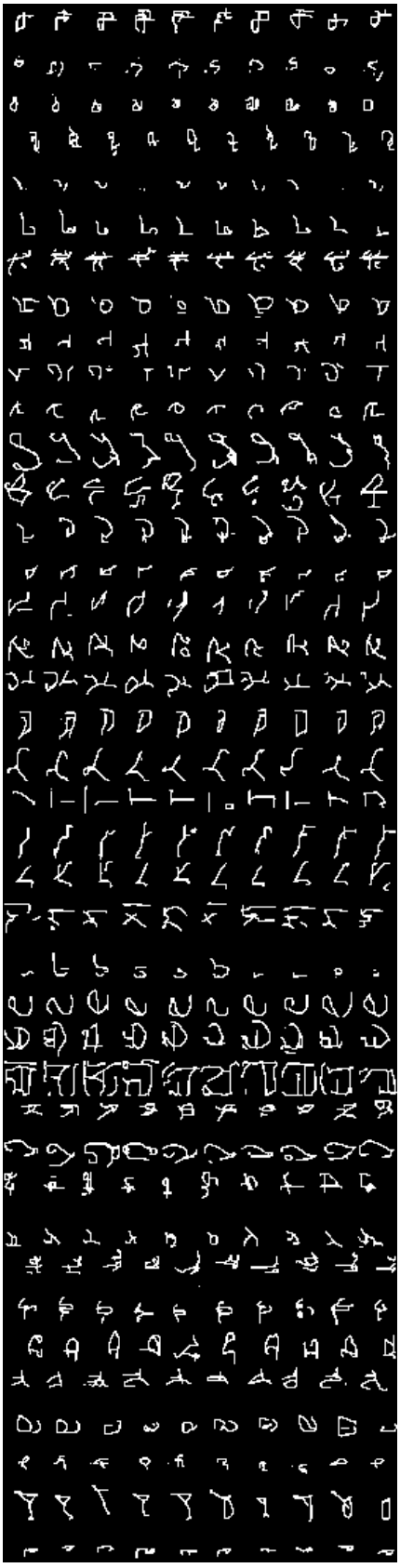

Figure 1: Zero-shot generation from our model. In each row we sample $\eta \sim p(\eta)$, then $z_i \sim p(z|\eta)$ and $x_i \sim p(x|z_i)$. Each row contains examples of a different imagined class.

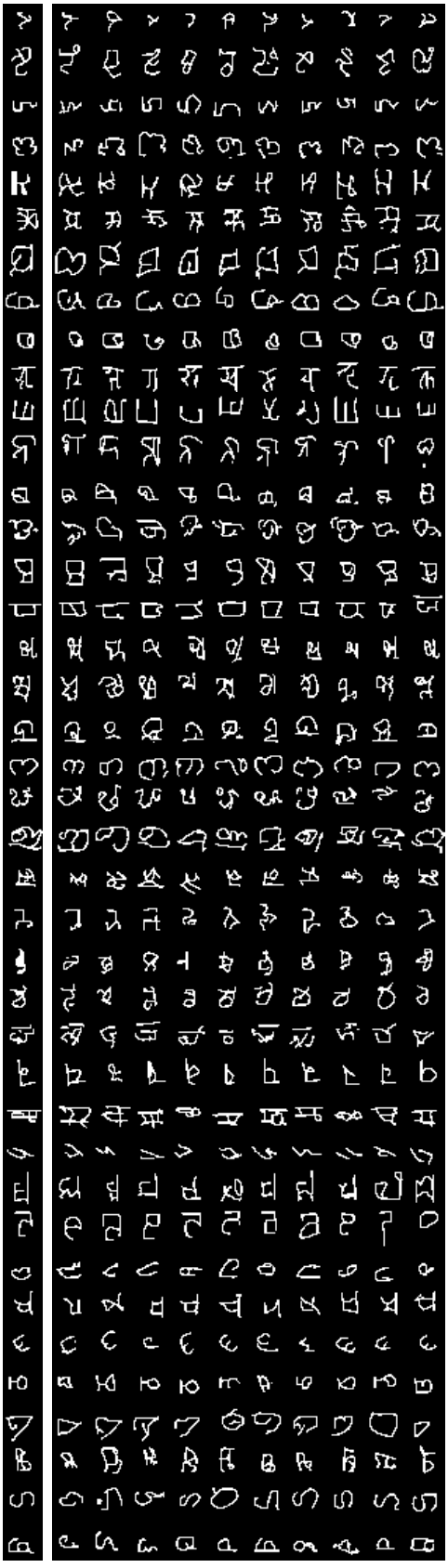

Figure 2: One-shot generation from our model. Left: Data from the Omniglot test set. Right: 10 Samples from our model of these newly seen classes. Each row corresponds to a different character.

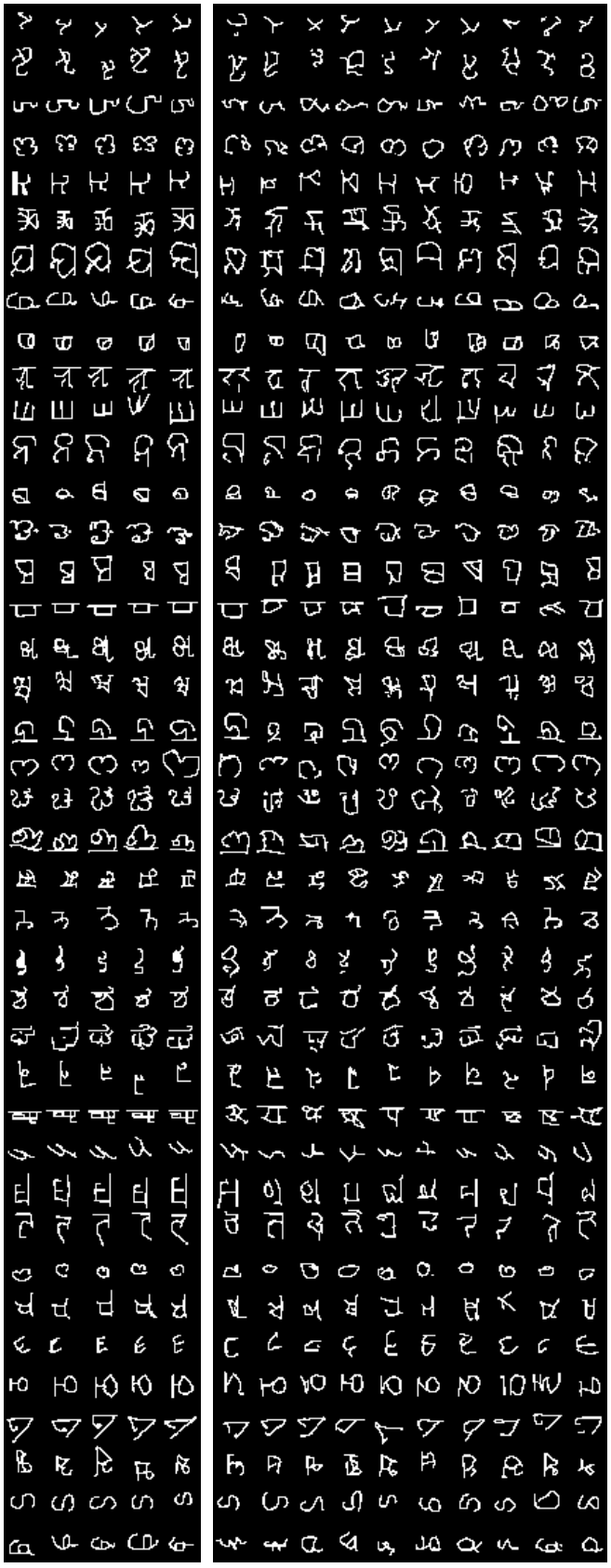

Figure 3: Five-shot generation from our model. Left: Data from the Omniglot test set. Right: 10 Samples from our model of these newly seen classes. Each row corresponds to a different character.