Fast Adaptation in Generative Models with Generative Matching Networks

Sergey Bartunov[†] **& Dmitry P. Vetrov**[‡]
National Research University Higher School of Economics (HSE)[†‡], Yandex[‡]
Moscow, Russia

Abstract

We develop a new class of deep generative model called generative matching networks (GMNs) which is inspired by the recently proposed matching networks for one-shot learning in discriminative tasks. By conditioning on the additional input dataset, generative matching networks may instantly learn new concepts that were not available during the training but conform to a similar generative process, without explicit limitations on the number of additional input objects or the number of concepts they represent. Our experiments on the Omniglot dataset demonstrate that GMNs can significantly improve predictive performance on the fly as more additional data is available and generate examples of previously unseen handwritten characters once only a few images of them are provided.

1 Introduction

Deep generative models are currently one of the most promising directions in generative modelling. In this class of models the generative process is defined by a composition of conditional distributions modelled using deep neural networks which form a hierarchy of latent and observed variables.

Such models are trained by stochastic gradient methods which can handle large datasets and a wide variety of model architectures but also present certain limitations. The training process usually consists of small, incremental updates of networks' parameters and requires many passes over training data. Notably, once a model is trained it cannot be adapted to newly available data without complete re-training to avoid catastrophic interference (McCloskey & Cohen, 1989; Ratcliff, 1990). There is also a risk of overfitting for concepts that are not represented by enough training examples caused by high capacity of the models. Hence, most of deep generative models are not well-suited for rapid learning which is often required in real-world applications where data acquisition is expensive.

In this paper we present Generative Matching Networks (GMNs), a new class of deep generative models suitable for fast learning in few-shot setting. Instead of learning a single generative distribution for a fixed training dataset, GMNs develop an adaptation mechanism that can be used at the test time to learn new concepts similar to the training data, e.g. new classes of handwritten characters with just a single image of each.

Similarly-stated problems were considered before in the literature, for example, Salakhutdinov et al. (2013) used ideas from deep learning and Bayesian nonparametrics to create a model suitable for low-data regime. Later, Lake et al. (2015) proposed Bayesian program learning approach for data-efficient inference of probabilistic generative programs. However, while these approaches demonstrated the ability of few-shot learning of generative models, they relied on the extensive sampling-based inference procedures which made the *fast* learning ability generally hard to achieve.

The next generation of conditional generative models used the amortized variational inference approach (Gershman & Goodman, 2014), i.e. learning a complementary neural network performing regression over parameters of the approximate posterior. Rezende et al. (2016) proposed a conditional variational autoencoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014) that was able to condition on a single object to produce new examples of the concept it represents. Later, Edwards & Storkey (2016) proposed another extension of VAE that maintained a global latent variable capturing statistics about multiple input objects which was used to condition the generative distribution.

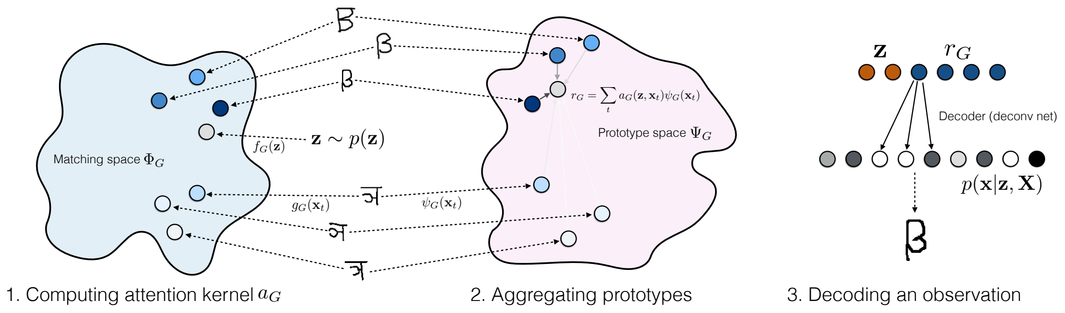

Matching space Φ_G
$f_G(\mathbf{z})$
$\mathbf{z} \sim p(\mathbf{z})$
$g_G(\mathbf{x}_t)$
$\psi_G(\mathbf{x}_t)$
$r_G = \sum_t a_G(\mathbf{z},\mathbf{x}_t)\psi_G(\mathbf{x}_t)$
Prototype space Ψ_G
$\mathbf{z}$
r_G
Decoder (deconv net)
$p(\mathbf{x}|\mathbf{z},\mathbf{X})$
1. Computing attention kernel a_G
2. Aggregating prototypes
3. Decoding an observation

Figure 1: Conditional generation of a new sample.

It allowed to implement the fast learning ability, but due to the particular model architecture used the model was not well-suited to datasets consisting of several different concepts.

2 Generative Matching Networks

Generative matching networks aim to model conditional generative distributions of the form $p(\mathbf{x}|\mathbf{X}, \boldsymbol{\theta})$, where $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$ is a conditioning dataset of size T. Similarly to other deep generative models we introduce a local latent variable $\mathbf{z}$. Thus the full joint distribution of our model can be expressed as:

$$p(\mathbf{x}, \mathbf{z}|\mathbf{X}, \boldsymbol{\theta}) = p(\mathbf{z}|\mathbf{X}, \boldsymbol{\theta})p(\mathbf{x}|\mathbf{z}, \mathbf{X}, \boldsymbol{\theta}). \tag{1}$$

We also maintain a recognition model approximating the posterior over the latent variable $\mathbf{z}$: $q(\mathbf{z}|\mathbf{x}, \mathbf{X}, \boldsymbol{\phi}) \approx p(\mathbf{z}|\mathbf{x}, \mathbf{X}, \boldsymbol{\theta})$.

The main idea behind GMNs is to employ the dependency on the conditioning dataset $\mathbf{X}$ by adopting the matching procedure originally proposed in (Vinyals et al., 2016) for supervised learning problems to unsupervised setting. In the basic version of our model, the prior is simply a standard Normal distribution and the conditional likelihood $p(\mathbf{x}|\mathbf{z}, \mathbf{X}, \boldsymbol{\theta})$ generates a new observation by matching sample from the prior $\mathbf{z}$ with each of the conditioning objects $\mathbf{x}' \in \mathbf{X}$ to extract few representative examples that would be used as prototypes for generation.

Further we describe the generative process more formally. First, we denote the space of latent variables as $\mathcal{Z}$ and the space of observations as $\mathcal{X}$. In addition to the latent space $\mathcal{Z}$ our model also defines the matching space Φ_G and the prototype space Ψ_G (subscript $\cdot_G$ here stands for "generative"). The former is used to match samples of the latent variable with conditioning objects while the latter is used as a space of decoder's inputs. We map both $\mathbf{z}$ and $\mathbf{x}' \in \mathbf{X}$ to the same *matching* space Φ_G by establishing $f_G : \mathcal{Z} \to \Phi_G$ and $g_G : \mathcal{X} \to \Phi_G$ respectively. The results of mapping are compared using a similarity function $\text{sim}(\cdot, \cdot)$ (dot product in our implementation) to form an *attention kernel* $a_G(\mathbf{z}, \mathbf{x})$:

$$a_G(\mathbf{z}, \mathbf{x}_t) = \frac{\exp(\text{sim}(f_G(\mathbf{z}), g_G(\mathbf{x}_t)))}{\sum_{t'=1}^{T} \exp(\text{sim}(f_G(\mathbf{z}), g_G(\mathbf{x}_{t'})))}, \quad r_G = \sum_{t=1}^{T} a_G(\mathbf{z}, \mathbf{x}_t)\psi_G(\mathbf{x}_t) \tag{2}$$

This attention kernel is used for interpolation between the *prototypes* of conditioning objects that are defined in the prototype space Ψ_L. The conditioning objects are converted to prototypes using mapping $\psi_G : \mathcal{X} \to \Psi_G$. Finally, the weighted average r_G of those prototypes is used as an input to decoder which returns a distribution on x, e.g. a deconvolution network. The described generative process is also shown on the figure 1.

In order to make our model applicable in the case when no additional data $\mathbf{X}$ is available, by default we assume presence of a *pseudo-input* $\mathbf{x}^*$ which is implicitly modelled as a supposed output of the functions $f_G^* = f_G(x^*)$, $g_G^* = g_G(x^*)$ and $\psi_G^* = \psi_G(x^*)$. Further we refer to models without pseudo-input as *conditional*.

The recognition model $q(\mathbf{z}|\mathbf{x}, \mathbf{X}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}, \mathbf{X}, \boldsymbol{\phi}), \Sigma(\mathbf{x}, \mathbf{X}, \boldsymbol{\phi}))$ has the similar structure and maintains its own functions f_R, g_R and ψ_R defining spaces Ψ_R and Φ_R. The only difference is that

Table 1: Conditional negative log-likelihoods for the test part of Omniglot.

		Number of conditioning examples							
Model	$C_{\textbf{test}}$	**0**	**1**	**2**	**3**	**4**	**5**	**10**	**19**
GMN, $C_{\text{train}} = 2$	1	89.7	83.3	78.9	75.7	72.9	70.1	59.9	45.8
GMN, $C_{\text{train}} = 2$	2	89.4	86.4	84.9	82.4	81.0	78.8	71.4	61.2
GMN, $C_{\text{train}} = 2$	4	89.3	88.3	87.3	86.7	85.4	84.0	80.2	73.7
GMN, $C_{\text{train}} = 2$, conditional	1		93.5	82.2	78.6	76.8	75.0	69.7	64.3
GMN, $C_{\text{train}} = 2$, conditional	2			86.1	83.7	82.8	81.0	76.5	71.4
GMN, $C_{\text{train}} = 2$, conditional	4					86.8	85.7	82.5	78.0
VAE		89.1							
GMN, $C_{\text{train}} = 1$, avg	1	92.4	84.5	82.3	81.4	81.1	80.4	79.8	79.7
GMN, $C_{\text{train}} = 2$, avg	2	88.2	86.6	86.4	85.7	85.3	84.5	83.7	83.4
GMN, $C_{\text{train}} = 1$, avg, conditional	1		88.0	84.1	82.9	82.4	81.7	80.9	80.7
GMN, $C_{\text{train}} = 2$, avg, conditional	2			85.7	85.0	85.3	84.6	84.5	83.7

function f_R is defined as $f_R : \mathcal{X} \to \Phi_R$ since instead of latent variable, the recognition model is conditioned on observation (and represents a distribution over latent variables). We also found it useful to employ the full contextual embedding described in (Vinyals et al., 2016; 2015) that allows to perform multiple attention steps of the form (2) over conditioning data, thus processing it jointly. It also allowed us to implement data-dependent prior $p(\mathbf{z}|\mathbf{X}, \boldsymbol{\theta})$. Please refer to the appendix for more details about model architecture.

3 Experiments

We evaluate our model on the Omniglot dataset (Lake et al., 2015) which consists of 1623 classes of handwritten characters from 50 different alphabets. Only 20 examples of each class are available which makes this dataset specifically useful for few-shot learning problems. We used the canonical train/test split provided by Lake et al. (2015). In order to speed-up training we downscaled images to 28×28 resolution.

We trained our model by maximizing the following objective: $\mathbb{E}_{p_d(\mathbf{X})}\left[\log p(\mathbf{X}|\boldsymbol{\theta})\right]$, where $p_d(\mathbf{X})$ is a data-generating distribution over datasets of length $T = 20$ that was constrained to generate datasets consisting of examples that represent up to C randomly selected classes so that the model has a clear incentive to re-use conditioning data. Since exact maximization of the marginal likelihood is intractable, we approximated it using variational inference with the recognition model serving as a variational approximation for the posterior.

We compared models by expected conditional log-likelihood $\mathbb{E}_{p_d(\mathbf{X})} p(\mathbf{x}_t|\mathbf{X}_{<t})$ at different number of conditioning examples available, from 0 (the unconditional case) to 19 ($t = 20$). Besides GMNs trained with and without pseudo-inputs our comparison included the standard VAE and GMNs without attention where eq. (2) is replaced by simple, unweighted average of prototypes as a similar dataset summarization approach was used in the neural statistican model (Edwards & Storkey, 2016); this model is denoted as *avg*. The results can be found in table 1. One can see that predictive performance of all variants of GMNs improves as more data is available to the model. Attentional matching is clearly beneficial comparing to the simple prototypes averaging, especially on datasets of higher diversity (C_{test}) and with more conditioning examples.

4 Conclusion

In this paper we presented a new class of conditional deep generative models called Generative Matching Networks. These models are capable of fast adaptation to a conditioning dataset by adjusting both the latent space and the predictive density while making very few assumptions on the data. We believe that these ideas can evolve further and help to implement more data-efficient models in other domains such as reinforcement learning where data acquisition is especially hard.

Acknowledgments

We would like to thank Michael Figurnov and Timothy Lillicrap for useful discussions. Dmitry P. Vetrov is supported by RFBR project No.15-31-20596 (mol-a-ved) and by Microsoft: MSU joint research center (RPD 1053945).

References

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

Samuel J Gershman and Noah D Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

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

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.

Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1278–1286, 2014.

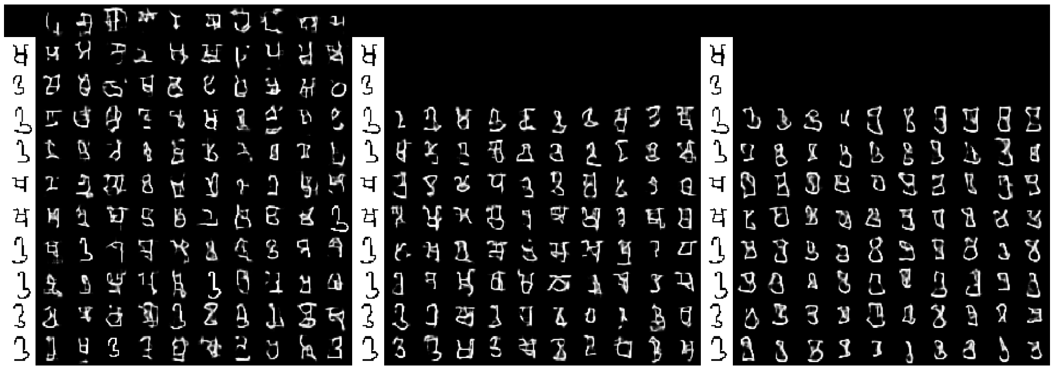
Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.

Ruslan Salakhutdinov, Joshua B Tenenbaum, and Antonio Torralba. Learning with hierarchical-deep models. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1958–1971, 2013.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.



(a) Full matching (b) Full matching, conditional (c) Average matching, conditional

Figure 2: Conditionally generated samples. First column contains conditioning data in the order it is revealed to the model. Row number t (counting from zero) consists of samples conditioned on first t input examples.

Appendix A. Samples from the model

Figure 2 contains samples generated using different number of conditioning examples. One can see that the conditional models (without pseudo-inputs) provide better-quality samples which is somewhat contradicting with the log-likelihood evaluation yet studied before in the literature Theis et al. (2015).

Appendix B. Full matching procedure

Although the basic model is capable of instant adaptation to the conditioning dataset $\mathbf{X}$, it admits a number of extensions that can seriously improve it's performance.

The disadvantage of the basic matching procedure (2) is that conditioning observations $\mathbf{X}$ are embedded to the space Φ independently from each other, via pairwise comparisons. Similarly to discriminative matching networks we address this problem by computing *full contextual embeddings* (FCE) (Vinyals et al., 2015). In order to obtain a joint embedding of conditioning data we allow K attentional passes over $\mathbf{X}$ of the form (2), guided by a recurrent controller R which accumulates global knowledge about the conditioning data in its hidden state h. The hidden state is thus passed to feature extractors f and g to obtain context-dependent embeddings.

We refer to this process as the *full matching* procedure which modifies equation (2) as:

$$\begin{aligned} r_G^k &= \sum_{t=1}^{T} a_G(\mathbf{z}, \mathbf{x}_t)\psi_G(\mathbf{x}_t), \quad a_G(\mathbf{z}, \mathbf{x}_t) = \frac{\exp(\mathrm{sim}(f_G(\mathbf{z}, h_k), g_G(\mathbf{x}_t, h_k)))}{\sum_{t'=1}^{T}\exp(\mathrm{sim}(f_G(\mathbf{z}, h_k), g_G(\mathbf{x}_{t'}, h_k)))}, \\ h_{k+1} &= R(h_k, r_G^k). \end{aligned} \tag{3}$$

The output of the full matching procedure is thus the interpolated prototype vector from the last iteration r_L^K and the last hidden state of h_{K+1}. The analogous procedure is used for obtaining r_R^K in recognition model.

Variables h_{K+1} aggregate information about the whole set of conditioning objects $\mathbf{X}$. Thus we can use similar process for making our model more flexible and establish *data-dependent* prior $p(\mathbf{z}|\mathbf{X})$. We establish recurrent controller as follows

$$\begin{aligned} r_P^k &= \sum_{t=1}^{T} a_P(\mathbf{x}_t)\psi_P(\mathbf{x}_t), \quad a(\mathbf{x}_t) = \frac{\exp(\mathrm{sim}(f_P(h_k), g_P(\mathbf{x}_t, h_k)))}{\sum_{t'=1}^{T}\exp(\mathrm{sim}(f_P(h_k), g_P(\mathbf{x}_{t'}, h_k)))}, \\ h_{k+1} &= R(h_k, r_P^k). \end{aligned} \tag{4}$$

Output of the procedure is then used to compute parameters of the prior, i.e. means and diagonal covariance matrix in our case: $p(\mathbf{z}|\mathbf{X}) = \mathcal{N}(\mathbf{z}|\mu(r_0^K), \Sigma(r_0^K))$.

Similarly to the basic matching procedure, we also implicitly add a pseudo-input $\mathbf{x}^*$ for models that are not denoted as conditional.

In our implementation, we shared the recurrent controller between generative and recognition models and had a separate controller for the prior. During preliminary experiments we empirically found that $K = 4$ attentional passes in the shared controller and a single pass for the prior's controller provides a reasonable trade-off between computational costs and predictive performance, hence we used these values in all reported experiments.

APPENDIX C. MODEL ARCHITECTURE

CONDITIONAL GENERATOR

The conditional generator network producing parameters for $p(\mathbf{x}|\mathbf{z}, \mathbf{X}, \boldsymbol{\theta})$ has concatenation of $\mathbf{z}$ and the output of the matching operation $[r, h]$ as input which is transformed to $3 \times 3 \times 32$ tensor and then passed through 3 residual blocks of transposed convolutions. Each block has the following form:

$$\begin{aligned} h &= \text{conv}_1(x), \\ y &= f(\text{conv}_2(h) + h) + \text{pool}(\text{scale}(x)), \end{aligned}$$

where f is a non-linearity which in our architecture is always parametric rectified linear function (He et al., 2015).

The block is parametrized by size of filters used in convolutions conv_1 and conv_2, shared number of filters F and stride S.

- scale is another convolution with 1×1 filters and the shared stride S.
- In all other convolutions number of filters is the same and equals F.
- conv_1 and pool have also stride S.
- conv_2 preserve size of the input by padding and has stride 1.

Blocks used in our paper have the following parameters $(W_1 \times H_1, W_2 \times H_2, F, S)$:

1. $(2 \times 2, 2 \times 2, 32, 2)$
2. $(3 \times 3, 3 \times 3, 16, 2)$
3. $(4 \times 4, 3 \times 3, 16, 2)$

Then log-probabilities for binary pixels were obtained by summing the result of these convolutions along the channel dimension.

FEATURE ENCODER ψ

Function ψ has an architecture which is symmetric from the generator network. The only difference is that the scale scale operation is replaced by bilinear upscaling.

The residual blocks for feature encoder has following parameters:

1. $(4 \times 4, 3 \times 3, 16, 2)$
2. $(3 \times 3, 3 \times 3, 16, 2)$
3. $(2 \times 2, 2 \times 2, 32, 2)$

The result is a tensor of $3 \times 3 \times 32 = 288$ dimensions.

FUNCTIONS f AND g

Each function f or g used in our model is simply an affine transformation of feature encoder's output (interpreted as a vector) which is shared across different parts of the model to a 200-dimensional space followed by parametric rectified non-linearity.