Few-shot Generative Modelling with Generative Matching Networks

Sergey Bartunov and **Dmitry P. Vetrov**
National Research University Higher School of Economics[1]

Abstract

Despite recent advances, the remaining bottlenecks in deep generative models are necessity of extensive training and difficulties with generalization from small number of training examples. We develop a new generative model called Generative Matching Network which is inspired by the recently proposed matching networks for one-shot learning in discriminative tasks. By conditioning on the additional input dataset, our model can instantly learn new concepts that were not available in the training data but conform to a similar generative process. The proposed framework does not explicitly restrict diversity of the conditioning data and also does not require an extensive inference procedure for training or adaptation. Our experiments on the Omniglot dataset demonstrate that Generative Matching Networks significantly improve predictive performance on the fly as more additional data is available and outperform existing state of the art conditional generative models.

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. This approach allows to build models with complex, non-linear dependencies between variables and efficiently learn the variability across training examples.

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 retraining 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 which is caused by high capacity of the models. Hence, most of deep generative models are not well-suited for rapid learning in real-world applications where data acquisition is expensive or fast adaptation to new data is required.

We present Generative Matching Network (GMN), a deep generative model suitable for fast learning in the few-shot setting, that makes progress in both of these directions. After the model is trained on a particular domain, it can instantly adapt it's generative distribution by conditioning on the additional dataset from a similar domain, without invoking a computationally extensive inference procedure. Generative matching networks are inspired the attentional mechanism implemented in Matching Networks, originally proposed for supervised discriminative tasks (Vinyals et al., 2016). The attentional mechanism, extended to unsupervised learning tasks, allows to smoothly interpolate between similar examples in the conditioning data and implicitly account for the underlying class structure, hence being robust to interference from diverse data, which was not previously demonstrated by analogous models.

This paper is organized as follows. First, in section 2 we revisit the necessary background in variational approach to training generative models and mention the related work in conditional generative models. Then, in section 3 we describe the proposed generative model, its recognition counterpart and a number of extensions. Section 4 describes the training procedure for generative matching networks. Finally, section 5 contains experimental evaluation of the proposed model as both generative model and unsupervised feature extractor in small-shot learning settings. We conclude with discussion of the results in section 6. Additional experiments as well as the detailed specification of model architecture can be found in the supplementary material.

[1]This work was done at HSE, the first author is now at DeepMind.

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

2 Background

We consider the problem of learning a probabilistic generative model which can be expressed as a probability distribution $p(\mathbf{x}|\boldsymbol{\theta})$ over objects of interests $\mathbf{x}$ parameterized by $\boldsymbol{\theta}$. The major class of generative models introduce also *latent* variables $\mathbf{z}$ that are used to explain or generate an object $\mathbf{x}$ such that $p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{z}|\boldsymbol{\theta})p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})d\mathbf{z}$ and assumed to be non-observable.

Currently, the common practice is to restrict the conditional distributions $p(\mathbf{z}|\boldsymbol{\theta})$ and $p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})$ to tractable distribution families and use deep neural networks for regressing their parameters. The expressive power of deep non-linear generative models comes at a price since neither marginal distribution $p(\mathbf{x}|\boldsymbol{\theta})$ can be computed analytically nor it can be directly optimized in a statistically efficient way. Fortunately, intractable maximum likelihood training can be avoided in practice by resorting to adversarial training (Gutmann & Hyvärinen, 2012; Goodfellow et al., 2014) or variational inference framework (Kingma & Welling, 2013; Rezende et al., 2014) which we consider further.

2.1 Training with variational inference

Recent developments in variational inference alleviate problems with maximizing the intractable marginal likelihood $\log p(\mathbf{x}|\boldsymbol{\theta})$ by approximating it with a lower bound (Jordan et al., 1999):

$$\log p(\mathbf{x}|\boldsymbol{\theta}) \geq \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}) - \log q(\mathbf{z}|\mathbf{x},\boldsymbol{\phi})\right] \tag{1}$$

Tightness of the bound is controlled by the recognition model $q(\mathbf{z}|\mathbf{x},\boldsymbol{\phi})$ which aims to minimize Kullback-Leibler divergence to the true posterior $p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$.

Similarly to the generative model, recognition model may also be implemented with the use of deep neural networks or other parameter regression which is known as *amortized inference* (Gershman & Goodman, 2014). Amortized inference allows to use a single recognition model for many training examples. Thus, it is convenient to perform training of the generative model $p(\mathbf{x}|\boldsymbol{\theta})$ by stochastic gradient optimization of variational lower bounds (1) corresponding to independent observations $\{\mathbf{x}_i\}_{i=1}^N$.

The clear advantage of this approach is its scalability. Every stochastic update to the parameters computed from only a small portion of training examples has an immediate effect on the whole dataset. However, while a single parameter update may be relatively fast, a large number of them is required to significantly improve generative or inferential performance of the model. Hence, gradient training of generative models usually results into an extensive computational process which prevents from rapid incremental learning. In the next section we discuss potential solutions to this problem that allow to implement fast learning ability in generative models.

2.2 Fast learning in generative models

In probabilistic modelling framework the natural way of incorporating knowledge about newly available data is conditioning. One may design a model that being conditioned on the additional input data $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ represents a new generative distribution $p(\mathbf{x}|\mathbf{X},\boldsymbol{\theta})$.

An implementation of this idea can be found in the model by Rezende et al. (2016) which was able to produce new examples of a concept that was missing at the training time but had similarities in the underlying generative process with the other training examples. The model supported an explicit conditioning on a single observation $\mathbf{x}'$ representing the new concept to construct a new generative distribution of the form $p(\mathbf{x}|\mathbf{x}',\boldsymbol{\theta})$.

The explicit conditioning when adaptation is performed *by the model* itself and has to be learned is not the only way to propagate knowledge about new data. Another solution which is often encountered in Bayesian models is to maintain a *global* latent variable encoding information about the whole available dataset such that the individual observations are conditionally independent given it's value. Denoting the global variable as $\boldsymbol{\alpha}$, a typical model from this class would have the following form:

$$p(\mathbf{X}|\boldsymbol{\theta}) = \int p(\boldsymbol{\alpha}|\boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathbf{x}_t|\boldsymbol{\alpha},\boldsymbol{\theta}) d\boldsymbol{\alpha}.$$

The principal existence of such a global variable may be justified by the de Finetti's theorem (Diaconis & Freedman, 1980) under the exchangeability assumption. In global latent variable models, the conditional generative distribution $p(\mathbf{x}|\mathbf{X},\boldsymbol{\theta})$ is then defined implicitly via posterior over the global variable: $p(\mathbf{x}|\mathbf{X},\boldsymbol{\theta}) = \int p(\mathbf{x}|\boldsymbol{\alpha},\boldsymbol{\theta})p(\boldsymbol{\alpha}|\mathbf{X},\boldsymbol{\theta})d\boldsymbol{\alpha}$. Once there is an efficient inference procedure for the global variable $\boldsymbol{\alpha}$, learning or fast adaptation can be implemented straightforwardly.

There are several relevant examples of generative models with global latent variables used for model adaptation and few-shot learning. Salakhutdinov et al. (2013) combined deep Boltzmann machine (DBM) with nested Dirichlet process (nDP) in a Hierarchical-Deep (HD) model. While DBM was used to learn low-level features, the nonparametric distribution over high-level features defined via nDP allowed to infer a latent global hierarchy of concepts from the training data. Later, Lake et al. (2015) proposed Bayesian program learning (BPL) approach for building a generative model of handwritten characters. The model was defined as a probabilistic program contained fine-grained specification of prior knowledge of the task such as generation of strokes and their composition into characters mimicking

human drawing behaviour.

While being suitable for learning from small data, both HD and BPL models required extensive sampling as a necessary part of either training or generation procedures. Hence, although Bayesian inference over the global latent variable may prevent overfitting, *fast* learning still remains a challenge for sampling-based inference.

The recently proposed neural statistician model (Edwards & Storkey, 2016) is another deep generative model with a global latent variable. The model was trained by optimizing a variational lower bound following the approach described in section 2.1, but with an additional recognition model approximating posterior distribution over the global latent variable. Authors designed the recognition model to be computationally efficient and require only a single pass over data which consisted of extracting special features from the examples, applying to them a pooling operation (e.g. averaging) and passing the result to another network providing parameters of the variational approximation.

This simple architecture allowed for the fast learning and guaranteed invariance to both data permutations and size of the conditioning dataset. However, authors evaluated the fast learning ability in the model only in the setting where all of the training examples represented the same single concept. Indeed, as we show later in section 5.2, this approach is less efficient for adaptation to more complex data, perhaps because a fixed parametric description is too restrictive for an accurate representation of datasets of varying complexity.

3 Generative Matching Networks

Generative matching networks aim to model conditional generative distributions of the form

$$p(\mathbf{x}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{z}|\mathbf{X}, \boldsymbol{\theta}) p(\mathbf{x}|\mathbf{z}, \mathbf{X}, \boldsymbol{\theta}) d\mathbf{z}, \qquad (2)$$

where $\mathbf{z}$ is a latent variable generated by a (potentially data-dependent) prior $p(\mathbf{z}|\mathbf{X}, \boldsymbol{\theta})$ and $p(\mathbf{x}|\mathbf{z}, \mathbf{X}, \boldsymbol{\theta})$ is a conditional likelihood.

We assume that the model is allowed to train on a large number of examples from a certain domain, accumulating knowledge about the *domain* in parameters $\boldsymbol{\theta}$. At the test time, the model can be conditioned on an additional dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ and has to adapt it's generative distribution to the conditioning data.

In order to design a fast adaptation mechanism, we have to make certain assumptions about relationships between training data and the new data $\mathbf{X}$ used to condition the model. Thus we assume the homogeneity of generative processes for training and conditioning data up to some parametrization. The generative process is assumed to have an approximately linear dependence on such parameters, i.e. the interpolation between parameters corresponding to different examples of the same concept can serve as good parameters for generating other examples. Below we describe the proposed model as well as a number of extensions that can significantly improve it's performance.

3.1 Basic model

In the basic version of our model, the prior is simply a standard Normal distribution $p(\mathbf{z}) = \mathcal{N}(z|0, I)$ which does not depend on the conditioning data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$. A new observation is generated by first sampling the latent variable from the prior $\mathbf{z} \sim p(\mathbf{z})$ and then matching it with each of the conditioning objects $\mathbf{x}' \in \mathbf{X}$ to extract few relevant examples that would be used as prototypes for generation.

The relevance of the conditioning objects $\mathbf{x}' \in \mathbf{X}$ is determined by a similarity function $\mathrm{sim}(\cdot, \cdot)$. Since latent variables and observations typically have very different representations, we first project them to the same *matching space* Φ_G, where similarity can be computed naturally. A particularly convenient choice of such Φ_G would be a high-dimensional real vector space where the similarity function can be defined simply as cosine between corresponding vectors or, even simpler, dot-product which we used in our implementation.

The obtained similarity scores are then transformed through a softmax function defining the *attention kernel* $a_G(\mathbf{z}, \mathbf{x})$:

$$\begin{aligned} a_G(\mathbf{z}, \mathbf{x}_t) &= \frac{\exp(\mathrm{sim}(f_G(\mathbf{z}), g_G(\mathbf{x}_t)))}{\sum_{t'=1}^{T} \exp(\mathrm{sim}(f_G(\mathbf{z}), g_G(\mathbf{x}_{t'})))}, \\ r_G &= \sum_{t=1}^{T} a_G(\mathbf{z}, \mathbf{x}_t) \psi_G(\mathbf{x}_t), \end{aligned} \qquad (3)$$

where functions $f_G(\cdot)$ and $g_G(\cdot)$ implemented by neural networks map observations and latent variables to the matching space Φ_G.

The attention kernel provides a normalized weight assigned to a conditioning object which is used to extract different subsets of $\mathbf{X}$ conditioned on the sample $\mathbf{z}$ in a soft manner. These subsets are aggregated in r_G by interpolating between their *prototype* descriptions computed with another network $\psi_G(\cdot)$. It is important to see, that the generated output does not depend on the order in which conditioning objects $\mathbf{X}$ are provided to the model. Finally, the decoder network is provided with the weighted average r_G together with the latent variable $\mathbf{z}$ and outputs a distribution over observations. Figure 1 contains a cartoon explanation of the described generative process.

Functions $g_G(\cdot)$ and $\psi_G(\cdot)$ are applied to observations and hence can be considered as feature extractors. Since fea-

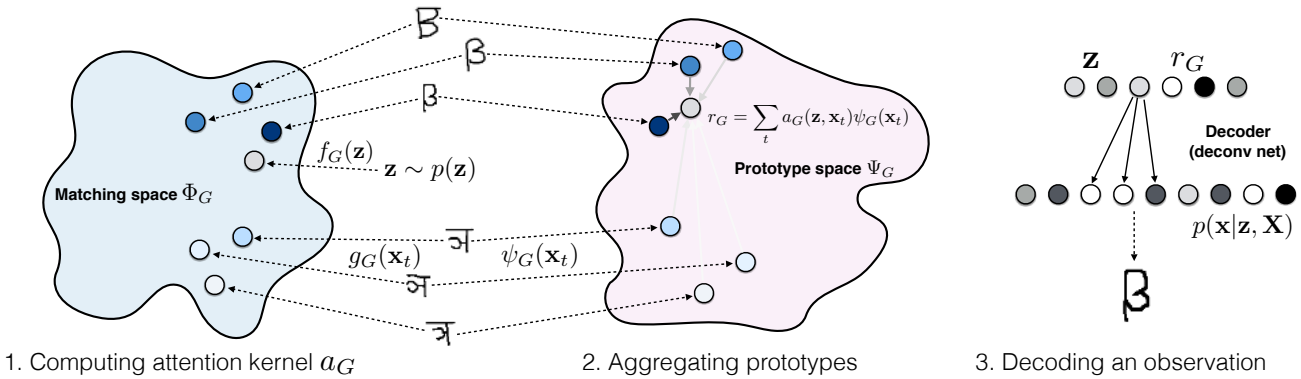


Figure 1: Generation of a new sample in a basic generative matching network, see section 3.1 for the description of functions f, g and ψ.

tures useful to specify the generative process are not necessarily good for discrimination and vice versa, it makes sense to represent these functions differently, taking into account nature of a considered domain. Thus, in our experiments these functions were implemented as convolutional networks with partial parameter sharing between them.

A major difference between the GMNs and the originally proposed discriminative matching networks (Vinyals et al., 2016) is that since no label information is available to the model, the interpolation in equations (3) is performed not in the label space but rather in the prototype space Ψ which itself is defined by the model and is learned during the training.

3.2 Recognition model

The recognition model $q(\mathbf{z}|\mathbf{X}, \mathbf{x})$ is used to approximate the posterior distribution of latent variable given object $\mathbf{x}$ in the context of previously observed conditioning objects $\mathbf{X}$. Similarly to the conditional likelihood, a similarity function operating in a (potentially different) matching space Ψ_R is used to form an attention kernel, with the difference that function f_R is now applied to the observation $\mathbf{x}$.

The attention kernel and the interpolated prototype are given by:

$$\begin{aligned} a_R(\mathbf{x}, \mathbf{x}_t) &= \frac{\exp(\text{sim}(f_R(\mathbf{x}), g_R(\mathbf{x}_t)))}{\sum_{t'=1}^{T} \exp(\text{sim}(f_R(\mathbf{x}), g_R(\mathbf{x}_{t'})))}, \\ r_R &= \sum_{t=1}^{T} a_R(\mathbf{x}, \mathbf{x}_t)\psi_R(\mathbf{x}_t) \end{aligned} \tag{4}$$

After the matching, interpolated prototype vector r_R is used to compute parameters of the approximate posterior which in our case was a normal distribution with diagonal covariance matrix, i.e. $q(\mathbf{z}|\mathbf{X}, \mathbf{x}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{z}|\mu(r_R), \Sigma(r_R))$.

3.3 Pseudo-inputs

One can note that the described model is not applicable in a situation where no conditioning objects are available, i.e. $\mathbf{X} = \emptyset$. A possible solution to this problem involves implicit addition of a *pseudo-input* to the set of conditioning objects, i.e. $\mathbf{X}$. There is no need to model pseudo-input as an actual observation, so we just represent it as corresponding outputs of functions $g^* = g(\mathbf{x}^*)$ and $\psi^* = \psi(\mathbf{x}^*)$ which are assumed to be another trainable parameters.

3.4 Full context matching

The potential limitation of the basic matching procedure (3) is that conditioning observations $\mathbf{X}$ are embedded independently from each other. Similarly to discriminative matching networks we address this problem by computing *full context embeddings* (Vinyals et al., 2015), i.e. embeddings that are computed jointly in the context of other conditioning examples.

We make K attentional passes over $\mathbf{X}$ of the form (3), 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 in order to obtain context-dependent embeddings.

We refer to this process as the *full context matching* procedure which modifies equation (3) as:

$$a_G(\mathbf{z}, h_k, \mathbf{x}_t) = \frac{\exp(\text{sim}(f_G(\mathbf{z}, h_k), g_G(\mathbf{x}_t, h_k)))}{\sum_{t'=1}^{T} \exp(\text{sim}(f_G(\mathbf{z}, h_k), g_G(\mathbf{x}_{t'}, h_k)))}, \tag{5}$$

where the interpolated prototype vector r_K and the hidden state h_k are given by:

$$\begin{aligned} r_G^{k+1} &= \sum_{t=1}^{T} a_G(\mathbf{z}, h_k, \mathbf{x}_t)\psi_G(\mathbf{x}_t, h_k), \\ h_{k+1} &= R(h_k, r_G^k). \end{aligned} \tag{6}$$

The output of the full matching procedure is thus the interpolated prototype vector from the last iteration r_G^K and the last hidden state of h_{K+1} passed to a decoder. The analogous procedure is used in the recognition model. In our implementation we shared the recurrent controller for generative and recognition models, thus we further refer to it as *shared controller*.

Since the full context matching consists of a sequence of basic matching operations, it is also invariant to the formal order of conditioning objects $\mathbf{X}$.

3.5 Data-dependent prior

Full context matching described in the previous section also allows us to implement a context-dependent prior $p(\mathbf{z}|\mathbf{X}, \boldsymbol{\theta})$ which adjusts our prior assumptions based on the conditioning data $\mathbf{X}$. Although in theory it should be possible to use a data-independent prior and translate the dependency on the conditioning data to the likelihood with the same effect, as we show below in our experiments, data-dependent prior has a positive effect on model's performance.

As before, we use the hidden state h of a recurrent controller to match the conditioning data:

$$a_P(h_k, \mathbf{x}_t) = \frac{\exp(\text{sim}(f_P(h_k), g_P(\mathbf{x}_t, h_k)))}{\sum_{t'=1}^{T} \exp(\text{sim}(f_P(h_k), g_P(\mathbf{x}_{t'}, h_k)))}, \tag{7}$$

and then aggregate the result:

$$\begin{aligned} r_P^{k+1} &= \sum_{t=1}^{T} a_P(h_k, \mathbf{x}_t)\psi_P(\mathbf{x}_t, h_k), \\ h_{k+1} &= R(h_k, r_P^k), \end{aligned} \tag{8}$$

where subscript $\cdot_P$ denotes "prior".

As opposed to the conditional likelihood and the recognition model, only the hidden state is passed to function f_P as there is no other information to use for matching. The prior is then parametrized as a normal distribution with diagonal covariance: $p(\mathbf{z}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}|\mu(r_P^k, h_{K+1}), \Sigma(r_P^k, h_{K+1}))$.

4 Training

Training of our model consists of maximizing marginal likelihood of a dataset $\mathbf{X}$ which can be expressed as:

$$p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{X}_{<t}, \boldsymbol{\theta}), \quad \mathbf{X}_{<t} = \{\mathbf{x}_s\}_{s=1}^{t-1}. \tag{9}$$

One should note that since each of the conditional distributions does not depend on the order of conditioning data $\mathbf{X}_{<t}$ provided, the joint distribution is also order-invariant.

We use the available training data to dynamically construct a large number of randomized few-shot learning problems and train GMNs to adapt on each of these problems simultaneously. Such a training strategy is rooted in curriculum learning (Bengio et al., 2009) and meta-learning (Thrun, 1998; Vilalta & Drissi, 2002; Hochreiter et al., 2001). It recently was successfully applied for one-shot discriminative learning (Santoro et al., 2016) and below we adapt it to our setting.

We define a *task-generating* distribution $p_d(\mathbf{X})$ which samples datasets $\mathbf{X}$ of size T from training data. Then we train our model to maximize the marginal likelihood of each dataset sampled on average:

$$\mathbb{E}_{p_d(\mathbf{X})} \left[\log p(\mathbf{X}|\boldsymbol{\theta})\right] \to \max_{\boldsymbol{\theta}}.$$

The standard practice is constrain p_d to generate datasets that consist only of objects of a single class, so that the model has a clear incentive to re-use conditioning data (Rezende et al., 2016; Edwards & Storkey, 2016). As we show further, Generative Matching Networks impose much less requirements for the data generating distribution and can be trained on datasets representing $C > 1$ different concepts or classes.

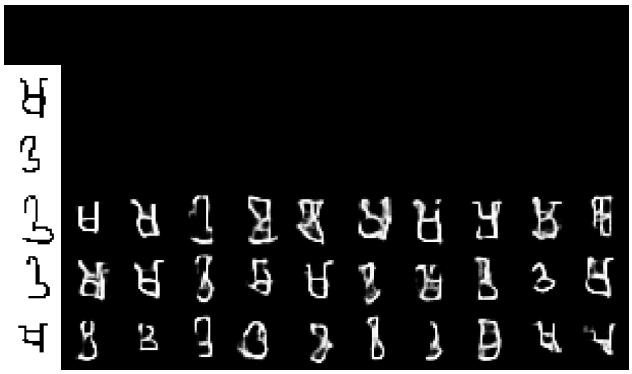
Since the marginal likelihood (9) as well as the conditional marginal likelihoods are intractable we instead use variational lower bound (see section 2.1) as a proxy to $\log p(\mathbf{X}|\boldsymbol{\theta})$:

$$\mathcal{L}(\mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_t, \mathbf{X}_{<t}, \boldsymbol{\phi})} \big[\log \frac{p(\mathbf{x}_t, \mathbf{z}_t|\mathbf{X}_{<t}, \boldsymbol{\theta})}{q(\mathbf{z}_t|\mathbf{x}_t, \mathbf{X}_{<t}, \boldsymbol{\phi})}\big].$$

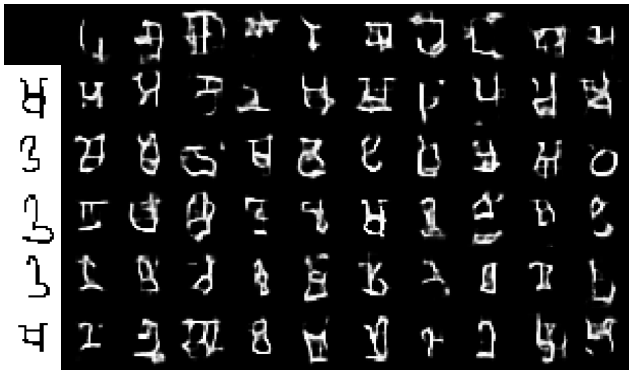
5 Experiments

For our experiments we use the Omniglot dataset (Lake et al., 2015) which consists of 1623 classes of handwritten characters from 50 different alphabets. The first 30 alphabets are devoted for training and the remaining 20 alphabets are left for testing. Importantly, only 20 examples of each class are available which makes this dataset specifically useful for few-shot learning problems. Unfortunately, the literature is inconsistent in usage of the dataset and multiple versions of Omniglot were used for evaluation which differ by train/test split, resolution, binarization and augmentation, see e.g. (Burda et al., 2015; Rezende et al., 2016; Santoro et al., 2016).

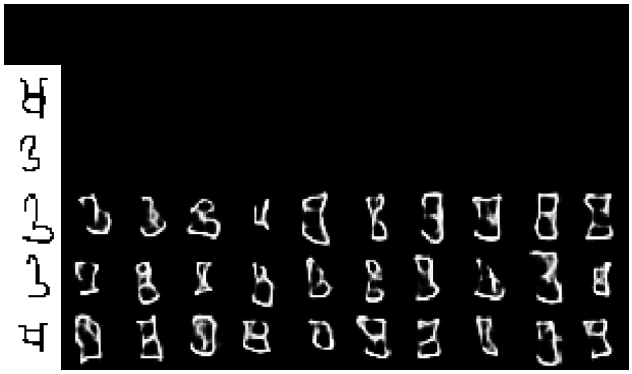
We use the canonical split provided by Lake et al. (2015). In order to speed-up training we downscaled images to 28×28 resolution and since the result was fully binary we did not apply any further pre-processing. We also did not augment our data in contrast to (Santoro et al., 2016; Edwards & Storkey, 2016) to make future comparisons with our results easier. Unless otherwise stated, we train models



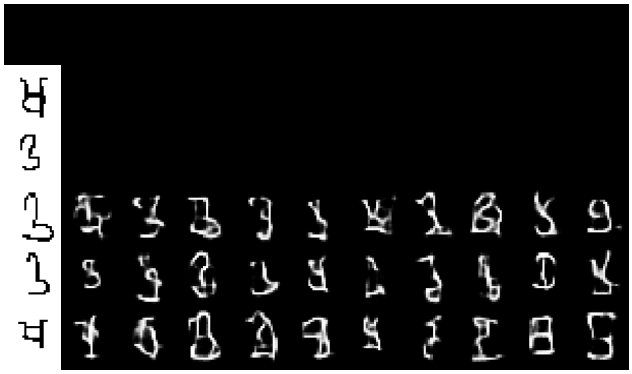
(a) GMN (no pseudo-input)



(b) GMN (one pseudo-input)



(c) GMN (no attention, no pseudo-input)



(d) Neural statistician

Figure 2: Conditionally generated samples. For each image: 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.

on datasets of length $T = 20$ and of up to $C_{\text{train}} = 2$ different classes as we did not observe any improvement from training on more diverse datasets.

5.1 Effect of number of attention steps

Since the full context matching procedure consists of multiple attention steps, it is interesting to see the effect of these numbers on model's performance. We trained several models with smaller architecture than the one used in further experiments and $T = 10$, varying number of attention steps allowed for the likelihood and recognition shared controller and the prior controller respectively. The models were compared using exponential moving averages of lower bounds corresponding to different numbers of conditioning examples $\mathbf{X}_{<t}$ obtained during the training. Results of the comparison can be found on figure 3.

Interestingly, larger numbers of steps lead to better results, however lower bounds are almost not improving after the shared controller is allowed for 4 steps. This behaviour was not observed with discriminative matching networks perhaps confirming the difficulty of unsupervised learning. Another important result is that the standard Gaussian prior makes adaptation significantly harder for the model yet still possible which justifies importance of the data-dependent prior (see section 3.5).

One may also see that all models preferred to set higher variances for a prior resulting to higher entropy comparing to standard normal prior. Clearly as more examples are available, generative matching networks become more certain about the data and output less dispersed Gaussians.

Based on this study, for our experiments we have chosen a model with 4 attention steps in the shared controller and single step in the prior controller, which is a reasonable compromise between computational cost and performance.

5.2 Fast learning and few-shot generation

In this section we compare generative matching networks with a set of baselines by expected conditional likelihoods $\mathbb{E}_{p_d(\mathbf{X})} p(\mathbf{x}_t|\mathbf{X}_{<t})$. The conditional likelihoods were estimated using importance sampling with 1000 samples from the recognition model used as a proposal. When training and evaluating conditional models we ensure that the first C_{train} objects in a dataset belong to different classes so that they in principle contain enough information to explain rest of the dataset.

5.2.1 Models

We compare different variants of generative matching networks with a set of baselines. To make the evaluation consistent, all the models use the same architecture for the encoder and the decoder parts, which we describe in detail in

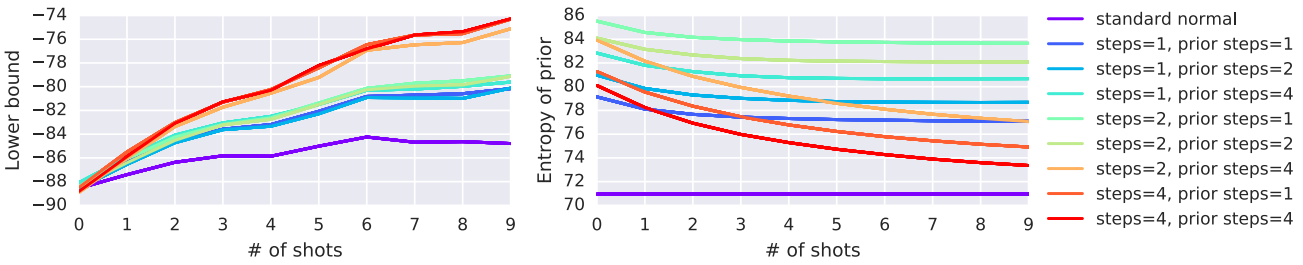


Figure 3: Lower bound estimates (left) and entropy of prior (right) for various numbers of attention steps and numbers of conditioning examples. Numbers are reported for the training part of Omniglot.

Table 1: Conditional negative log-likelihoods for the test part of Omniglot. C_{train} and C_{test} denote the maximum number of classes in datasets used for training and evaluating respectively.

		Number of conditioning examples							
Model	C_{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, $C_{\text{train}} = 2$	3	**89.6**	**88.1**	**86.0**	**85.0**	**84.1**	**82.0**	**76.3**	**69.4**
GMN, $C_{\text{train}} = 2$	4	**89.3**	**88.3**	**87.3**	**86.7**	**85.4**	**84.0**	**80.2**	**73.7**
GMN, $C_{\text{train}} = 2$, no pseudo-input	1		93.5	82.2	78.6	76.8	75.0	69.7	64.3
GMN, $C_{\text{train}} = 2$, no pseudo-input	2			86.1	83.7	82.8	81.0	76.5	71.4
GMN, $C_{\text{train}} = 2$, no pseudo-input	3				86.1	84.7	83.8	79.7	75.3
GMN, $C_{\text{train}} = 2$, no pseudo-input	4					86.8	85.7	82.5	78.0
VAE		89.1							
One-shot VAE	1		83.9						
Neural statistician, $C_{\text{train}} = 1$	1		102	83.4	77.8	75.2	74.6	71.7	71.5
Neural statistician, $C_{\text{train}} = 2$	2			86.4	**82.2**	82.3	80.6	79.7	79.0
GMN, $C_{\text{train}} = 1$, no attention	1	92.4	84.5	82.3	81.4	81.1	80.4	79.8	79.7
GMN, $C_{\text{train}} = 2$, no attention	2	88.2	86.6	86.4	85.7	85.3	84.5	83.7	83.4
GMN, $C_{\text{train}} = 1$, no attention, no pseudo-input	1		88.0	84.1	82.9	82.4	81.7	80.9	80.7
GMN, $C_{\text{train}} = 2$, no attention, no pseudo-input	2			85.7	85.0	85.3	84.6	84.5	83.7

the supplementary material.

VAE. In order to get a sense of quantitative improvement over non-conditional generative models, we implemented a variational auto-encoder (VAE) with a similar architecture to our model, but lacking any adaptation mechanisms.

One-shot VAE. Following Rezende et al. (2016), we implemented a simple conditional VAE of the form $p(\mathbf{x}|\mathbf{x}') = \int p(\mathbf{z}|\mathbf{x}')p(\mathbf{x}|\mathbf{z},\mathbf{x}')d\mathbf{z}$. This model aims at generating a new example of a character represented by $\mathbf{x}'$ and hence can be trained only with $C = 1$.

Neural Statistician. Another non-trivial baseline is the Neural Statistician model (Edwards & Storkey, 2016) (see section 2.2) which can be considered as state of the art in few-shot generative modelling.

Generative Matching Network (GMN). For evaluation, we used a model with full matching procedure (see section 3.4) using 4 steps for the shared controller and a single step for the prior controller (see supplementary materials for discussion of these parameters). Further, if not stated otherwise, we consider GMNs with a single pseudo-input (described in section 3.3), as we found the benefit of adding more pseudo-inputs negligible.

Generative Matching Network without attention (GMN, no attention). We also consider a restricted version of our model with the attentional matching procedure (3) replaced by a uniform kernel that effectively leads to simple averaging prototypes of all conditioning examples. The same architecture was used in the neural statistician model to aggregate conditioning data, except that the latter models uncertainty about this aggregate.

Table 1 contains results of the evaluation on test alphabets from Omniglot. One can see, that Generative Matching Networks demonstrate significant improvement in predictive performance as more data is available to the model. As one would expect, larger values of C_{test} made adaptation harder since on average less examples of the same class are

Table 2: Few-shot classification accuracy (%) on the test part of Omniglot. All models were trained with $C_{\text{train}} = 1$ and no pseudo-input (if applicable).

		5 CLASSES		20 CLASSES	
MODEL	METHOD	1-SHOT	5-SHOT	1-SHOT	5-SHOT
GMN	likelihood	82.7	**97.4**	64.3	**90.8**
GMN, no attention	likelihood	**90.8**	96.7	**77.0**	**91.0**
GMN	cosine	62.7	80.8	45.1	67.2
GMN, no attention	cosine	72.0	86.0	50.1	72.6
One-shot VAE	likelihood	**90.2**		76.3	
One-shot VAE	cosine	72.1		50.1	
Neural statistician	likelihood	82.0	94.8	63.1	87.6
Neural statistician	cosine	66.4	85.5	47.3	71.7
Matching networks, no fine-tuning	cosine	98.1	98.9	93.8	98.5

available to the model. Still GMNs are capable of working in low-data regime even when $C_{\text{test}} > C_{\text{train}}$.

Unsurprisingly, models with prototype averaging (GMN, no attention and Neural statistician) performed well for simple datasets constructed of a single class, although significantly worse than the proposed matching procedure. On more difficult datasets with mixed examples of two different classes ($C_{\text{test}} = 2$) GMNs clearly outperformed all concurrent models, thus justifying importance of nonparametric representations for complex data.

In order to visually assess the fast learning ability of GMNs we also provide conditionally generated samples in figure 2. Interestingly, the GMN without pseudo-inputs generated samples more similar to the conditioning data while sacrificing the predictive performance. Such counter-intuitive mismatch between visual quality of samples and predictive performance has been studied before (Theis et al., 2015) and may suggest that without a pseudo-input, GMNs tend to learn less of the common knowledge about the domain and slightly "overfit" to conditioning data. Therefore, presence or absence of the pseudo-input should depend on the target application of interest, i.e. density estimation or producing new examples. We provide more samples generated by GMNs in supplementary materials.

5.3 Classification

Even if an adaptive generative model did not take into account the label information during the training, as in the case of GMN, it can still be used for few-shot classification. Given a small number of labeled examples $\mathbf{X}_c = \{\mathbf{x}_{c,1}, \mathbf{x}_{c,2}, \dots \mathbf{x}_{c,N}\}$ for each class $c \in \{1, \dots, C\}$, it possible to use $p(\mathbf{x}|\mathbf{X}_c)$ as a score for assigning label c to $\mathbf{x}$. While one would not expect this method to provide state of the art results, few-shot classification using class-conditional densities is still a good test for adaptation capabilities.

Alternatively, one may use the recognition model $q(\mathbf{z}|\mathbf{x}, \mathbf{X}_1, \dots, \mathbf{X}_C)$ to extract features describing the new object $\mathbf{x}$ and then use a classifier of choice, e.g. nearest-neighbour classifier with cosine similarity of mean parameters as in our experiments. This method can be used as a general adaptive feature extraction technique which itself is an interesting application of GMNs.

The results under different number of training examples available are provided in table 2. Surprisingly, the simpler GMN with uniform attention outperformed all other models, including the full model. As shown in the previous section, GMNs without pseudo-inputs are very smooth density estimators, so maybe even conditioned on a number of same-class examples, they still assign enough probability mass to discrepant observations.

6 Conclusion

We presented Generative Matching Network, a new conditional generative model that is capable of fast adaptation to conditioning dataset by adjusting both the latent space and the predictive density. The nonparametric matching procedure enabling these features can be seen as a generalization of the original matching network architecture since it allows the model to define the label space itself, thus extending applicability of matching networks to unsupervised and perhaps semi-supervised settings. One of the principal innovations over existing conditional and few-shot generative models is that GMNs can naturally work with diverse conditioning data and put less assumptions on its homogeneity. We believe that these ideas can evolve further and help to implement more data-efficient models in other domains where data acquisition is especially hard.

Acknowledgements We would like to thank Michael Figurnov and Timothy Lillicrap for useful discussions. Dmitry Vetrov was partly supported by Samsung Research, Samsung Electronics and Russian Science Foundation grant 17-11-01027.

References

Bengio, Yoshua, Louradour, Jérôme, Collobert, Ronan, and Weston, Jason. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Burda, Yuri, Grosse, Roger, and Salakhutdinov, Ruslan. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Diaconis, Persi and Freedman, David. Finite exchangeable sequences. *The Annals of Probability*, pp. 745–764, 1980.

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

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

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Gutmann, Michael U and Hyvärinen, Aapo. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

Hochreiter, Sepp, Younger, A Steven, and Conwell, Peter R. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Jordan, Michael I, Ghahramani, Zoubin, Jaakkola, Tommi S, and Saul, Lawrence K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

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

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

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

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

Rezende, Danilo J, Mohamed, Shakir, and Wierstra, Daan. 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.

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

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

Santoro, Adam, Bartunov, Sergey, Botvinick, Matthew, Wierstra, Daan, and Lillicrap, Timothy. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

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

Thrun, Sebastian. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

Vilalta, Ricardo and Drissi, Youssef. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

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

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