A Review of Deep Gaussian Process: Learn Meaningful Embedding with Extreme Small Dataset

Yijin Zeng
Imperial College London

Abstract

This report reviews deep Gaussian processes [2] for learning meaningful embedding with extreme small dataset. We begin by revisiting Gaussian process models and single-layer Gaussian process latent variable models. Then, we focus on the deep Gaussian processes architecture, and its kernel design. The performance of deep Gaussian process models are evaluated on the MNIST dataset for figures generating. We analyze the effects of sample size and model depth on the model performances. We find out that even for extreme small sample size, e.g. 10 instances, the model learns meaningful latent representation.

1 Introduction

Deep learning is recognized as a class of machine learning methods based on artificial neural networks. It has demonstrated impressive performance across a wide range of tasks, including computer vision [7], autonomous driving [6], and drug discovery [1]. However, most deep learning algorithms rely heavily on large datasets to achieve good performance. This raises the question of how well deep learning can perform in data-scarce scenarios. This report reviews deep Gaussian processes (DGP) [2] for learning meaningful embedding with extreme small dataset.

DGP are deeply architectures based on Gaussian Process [5], which defines priors over "'sufficiently smooth" functions. It can be applied for both supervised and unsupervised learning. In this report, we focus on the unsupervised learning for the DGP. In other words, we are given high dimensional observations Y, and are interested in learning meaningful representation X in lower dimensional embedded latent space.

We first review the standard Gaussian Process and single layer Gaussian Process latent variable models. Then, we discuss the structures of the DGP, including the design of the kernel and model hierarchy. We evaluate the DGP on the MNIST dataset for figures generating and study the impact of sample size and model depth on the quality of generated figures.

2 Gaussian process

Gaussian Processes are defined as a collection of random variables, any finite number of which have a joint multivariate Gaussian distribution [5]. More formally, we assume the output y is generated from a signal term $f(x)$ by adding Gaussian noise ϵ

$$y = f(x) + \epsilon,$$

where $\epsilon \sim \mathbb{N}(0, \sigma_\epsilon^2)$ and the function $f(x)$ is distributed as a Gaussian process:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')).$$

Gaussian process is decided completely by the mean function and covariance function. The mean function $m(x)$ defines the expected value of y at x; the covariance function $k(x, x')$ describes the

dependence between the function values at different input x and x'. In practice, the mean function $m(x)$ is often set to 0 for computation simplicity and the covariance function is called $kernel$, which can specified using a set of parameters Θ. A popular choice of the kernel is the radial basis function kernel, defined as

$$k(t, t') = \sigma_f \exp\left(-\frac{||t - t'||^2}{2\lambda^2}\right).$$

The variance of likelihood function σ_ϵ^2 and the parameters of the kernel $\Theta = (\sigma_f, \lambda)$ are called hyper-parameters of the Gaussian process. These hyper-parameters are set to maximize the model's generalization ability through maximizing the marginal likelihood function [5]:

$$L = \log p(Y|\sigma_\epsilon, \Theta) = -\frac{1}{2}\log|K(T,T) + \sigma_\epsilon^2\mathbf{I}| - \frac{1}{2}Y^T\left[K(T,T) + \sigma_\epsilon^2\mathbf{I}\right]^{-1}Y - \frac{n}{2}\log 2\pi.$$

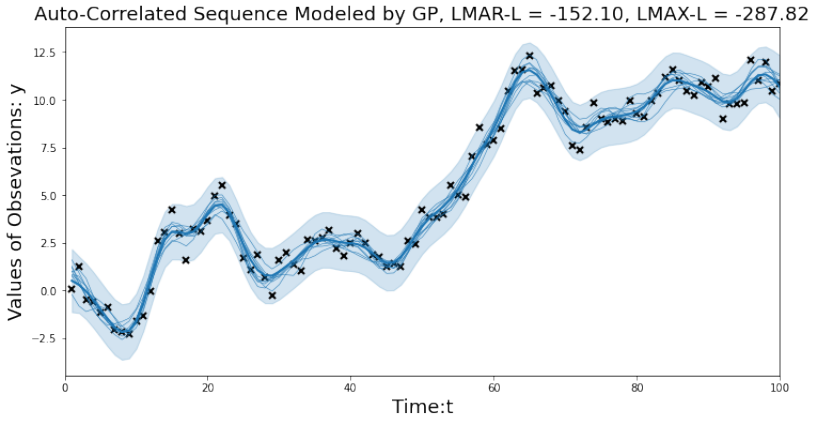

Figure 1: An example of Gaussian process: an auto-correlated sequence modeled by the Gaussian process. The hyperparameters are tuned with L-BFGS [4]. Log marginal likelihood (LMAR-L): = -152.10, log maximum likelihood under normal assumption(LMAX-L): = -287.82. The large log marginal likelihood indicates the Gaussian process generalize the observations well.

3 Gaussian process latent variable model

The Gaussian Process can also be used for unsupervised learning. In the Gaussian process latent variable model [3], the unobserved inputs X are treated as latent variables:

$$y_d = f_d(x) + \epsilon_d,$$

where y_d denotes the d column(dimension) of Y and $\epsilon_d \sim \mathbb{N}(0, \sigma_{d,\epsilon})$. In other words, each dimension of the observations Y is taken as generated by a Gaussian process. By further assuming the independence of each dimension of Y, we have the Gaussian process prior on X as

$$p(F|X) = \prod_{d=1}^{D} \mathbb{N}(f_d|0, k(x,x)),$$

with $F = \{f_d\}_{d=1}^D$. Integrating the prior, we have the marginal likelihood function:

$$-\frac{D}{2}\log|K(N,N) + \sigma_\epsilon^2\mathbf{I}| - \frac{D}{2}Y^T\left[K(N,N) + \sigma_\epsilon^2\mathbf{I}\right]^{-1}Y - \frac{DN}{2}\log 2\pi.$$

Our goal is to find the best representation is to find the best representation X in latent space. The learning process is subsequently to optimise this marginal likelihood with respect to the latent variables X as well as the hyper-parameters.

4 Deep Gaussian process

The Gaussian process latent variable model is flexible for non-linear embedding [3]. In more complex real life settings, however, single layer Gaussian process latent variable model is limited. It is thus natural to extend it to deeper and more complex models.

The architecture of the deep Gaussian process is as follows [2]: let leaf nodes $Y \in \mathbb{R}^{N\times D}$ denote observations; $X_h \in \mathbb{R}^{N\times Q_h}$ denote the intermediate latent layers, where $h = 1, \cdots, H-1$ denotes number of hidden layers and Q_h denotes the number of latent dimensions in layer h; $Z = X_H \in \mathbb{R}^{N\times Q_H}$ denote the parent nodes. As an illustration, a two-layer deep Gaussian process is:

$$
\begin{aligned}
y_{nd} &= f_d^Y(x_n) + \epsilon_{nd}^Y, d = 1, \cdots, D \\
x_{nd} &= f_q^Y(z_n) + \epsilon_{nq}^X, q = 1, \cdots, Q
\end{aligned}
$$

Automatic Relevance Determination Kernel. For each hidden layer h, the author in [2] designs the so-called automatic relevance determination kernel:

$$
k(x_i, x_j) = \sigma_{ard}^2 e^{-\frac{1}{2}\sum_{q=1}^{Q_h} w_q (x_{i,q} - x_{j,q})^2}.
$$

This covariance function automatically assigns different weights w_q to each latent dimension by maximizing the marginal likelihood.

More Hidden Layers. In two layer deep Gaussian process, the output of the parent nodes are treated as the input of the intermediate latent layers, which subsequently produce inputs to leaf nodes. Similarly, by taking the output of the higher layer as the input of lower layer, we have following similar structure:

$$
\begin{aligned}
y_{nd} &= f_d^Y(x_n) + \epsilon_{nd}^Y, d = 1, \cdots, D \\
x_{nd} &= f_q^Y(x_n) + \epsilon_{nq}^X, q = 1, \cdots, Q_1 \\
&\vdots \\
x_{nd} &= f_q^Y(x_n) + \epsilon_{nq}^X, q = 1, \cdots, Q_{H-1} \\
x_{nd} &= f_q^Y(z_n) + \epsilon_{nq}^X, q = 1, \cdots, Q_H.
\end{aligned}
$$

5 Experiments

In this section, we empirically investigate the effects of sample size and model depth on the performance of the deep Gaussian process (DGP). In Section 5.1, we evaluate a two-layer DGP on the MNIST dataset with varying sample sizes ranging from 10 to 50. Section 5.2 considers the same problem with different number of hidden layers from 1 to 3.

5.1 The effect of sample size

In this section, we train a DGP on the MNIST dataset with varying sample sizes for investigating the minimum number of observations required for a DGP to learn a meaningful latent representation Z. We use a two-layer DGP architecture, where the output layer has 2 latent dimensions and the intermediate layer has 5 latent dimensions. The training set is constructed by randomly sampling n examples for each of the digits 0, 1, 6 from MNIST. To study the effect of sample size, we experiment with $n \in \{10, 30, 50\}$ for each digit. Additional experiments and details are provided in the Appendix.

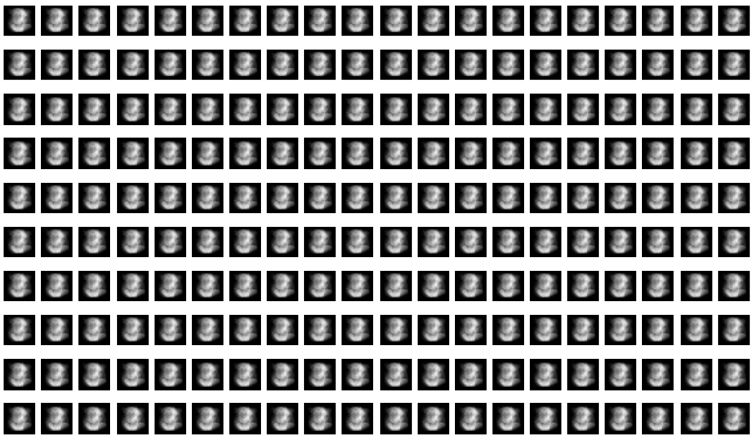

Figure 2: Sample Size: 10 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

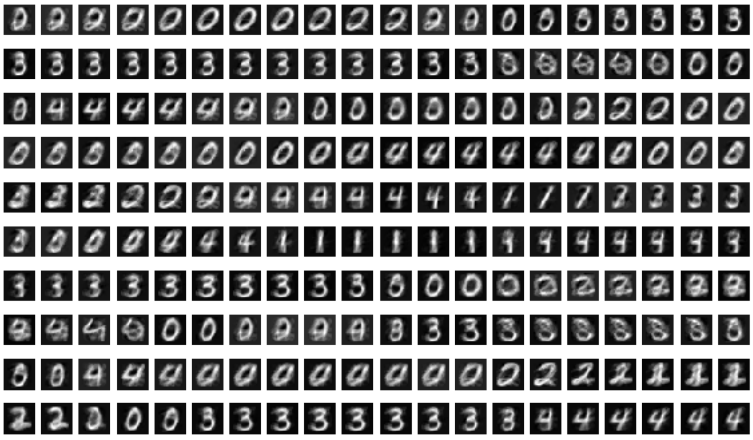

Figure 3: Sample Size: 30 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

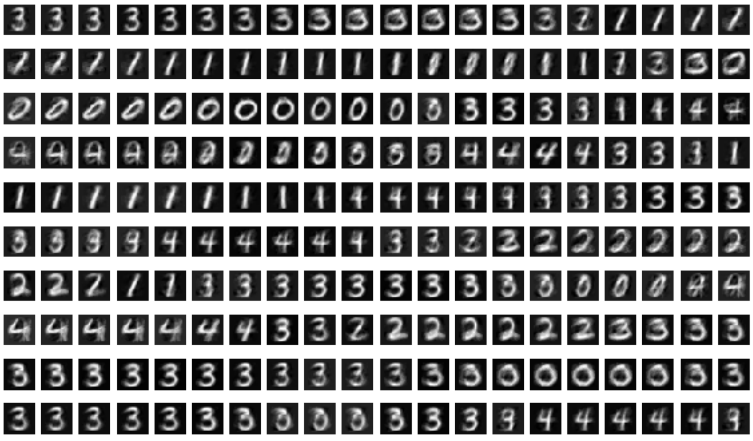

Figure 4: Sample Size: 50 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

Figures 2, 3, and 4 are generated by first sampling from the two-dimensional latent space and then mapping the samples back to the data space. By visualizing these mappings as digit images, we

observe that when the sample size is 10, the DGP fails to learn a meaningful mapping from the latent space to the data space, but when the sample size is 50 the DGP produces better results.

5.2 The effect of model depth

As previously mentioned, when the sample size is extremely small (e.g., 10), the DGP fails to learn informative mappings from the data space to latent space.

In this section, we investigate whether adjusting the depth of the model help recover useful latent representations under such data-scarce conditions. To investigate this, we vary the depth of the DGP from 1 to 3 layers, using the same training set for each model. The training set is randomly sampled from the MNIST dataset, consisting of 10 examples for each of the digits 0, 1, 6.

Figures 5, 6, and 7 are generated by sampling from the two-dimensional latent space (defined by the output layer of the DGP) and mapping the samples back to the data space. Interestingly, the results show that with such a limited sample size, simpler models (i.e., shallower DGPs) tend to produce more informative and structured mappings. This observation is further supported by marginal likelihood comparisons.

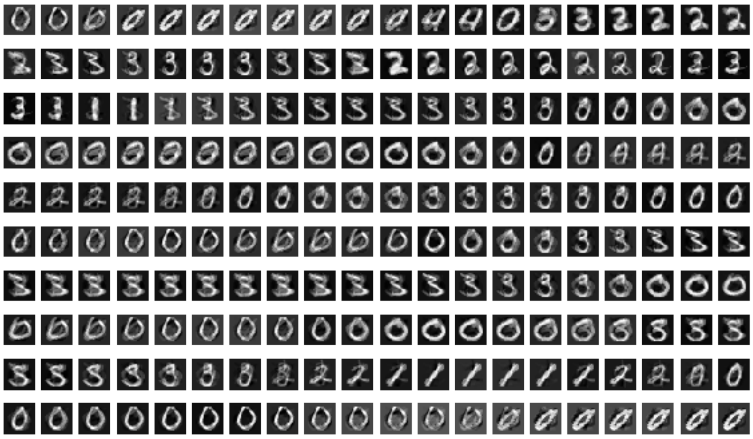

Figure 5: Sample Size: 10 examples for each {0,1,2,3,4}. 1 layers deep Gaussian process, log marginal likelihood: -740.88.

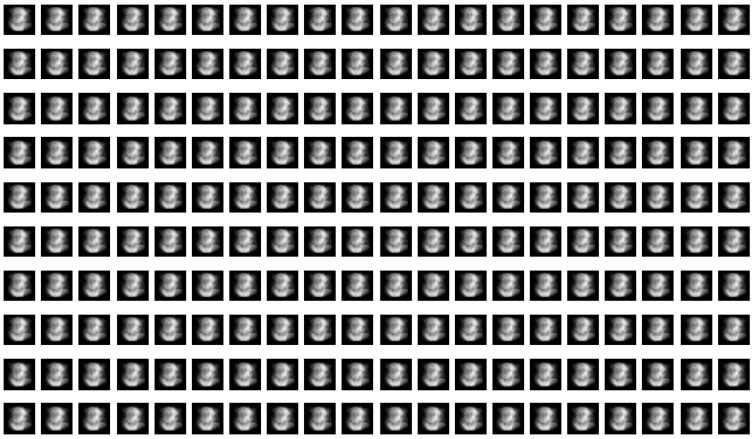

Figure 6: Sample Size: 10 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process, log marginal likelihood: -4205.70.

Figure 7: Sample Size: 10 examples for each {0,1,2,3,4}. 3 layers deep Gaussian process, log marginal likelihood: -4273.58.

Overall, we find that DGPs are capable of learning meaningful latent representations even with relatively small sample sizes (e.g., 30 training samples). When the sample size is extremely limited (e.g., 10 training samples), simpler models—such as a single-layer Gaussian process latent variable model—tend to perform better in practice.

6 Conclusion

This report reviews deep Gaussian process (DGP) models, a hierarchical extension of Gaussian processes. By evaluating DGPs on the MNIST dataset, we find that they are empirically well-suited for unsupervised learning tasks with limited observations. Notably, the performance of DGPs tends to depend on the model depth. When the sample size is small, shallower architectures—such as a single-layer DGP—are generally more effective than deeper ones.

References

[1] Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., and Blaschke, T. (2018). The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250.

[2] Damianou, A. and Lawrence, N. D. (2013). Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR.

[3] Lawrence, N. and Hyvärinen, A. (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research*, 6(11).

[4] Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

[5] Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer.

[6] Rastgoo, M. N., Nakisa, B., Maire, F., Rakotonirainy, A., and Chandran, V. (2019). Automatic driver stress level classification using multimodal deep learning. *Expert Systems with Applications*, 138:112793.

[7] Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018.

A Additional simulation results for the effect of sample size

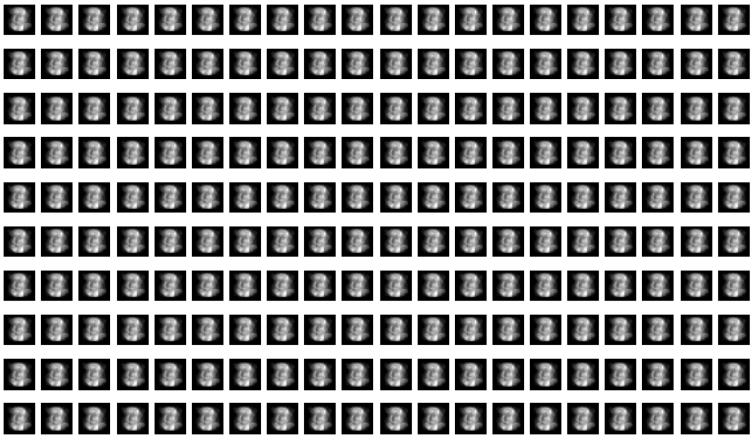

Figure 8: Sample Size: 5 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

Figure 9: Sample Size: 10 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

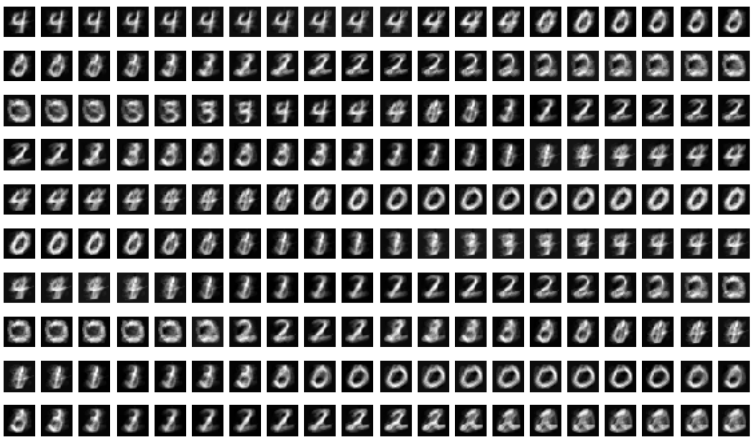

Figure 10: Sample Size: 15 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

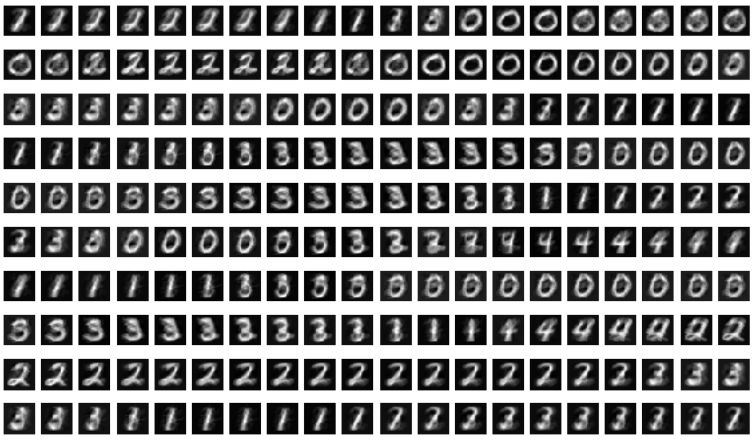

Figure 11: Sample Size: 20 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

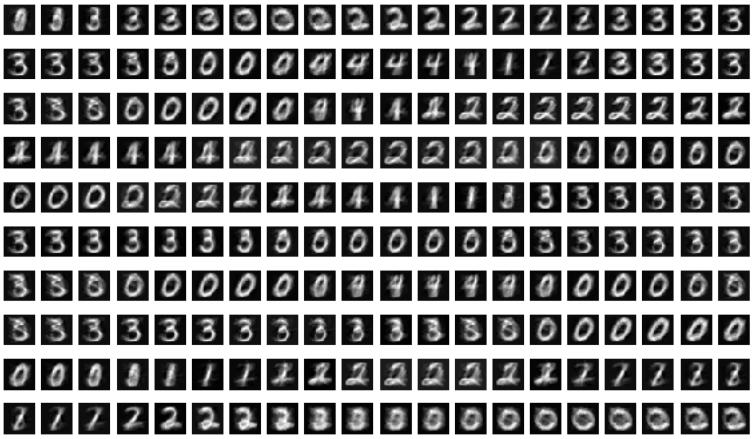

Figure 12: Sample Size: 25 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

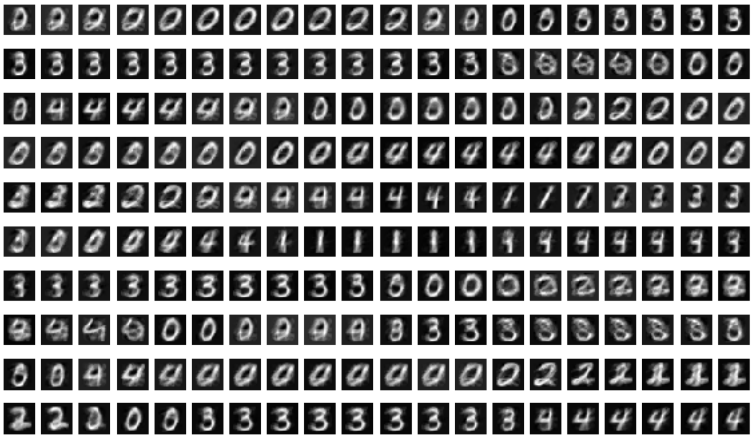

Figure 13: Sample Size: 30 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

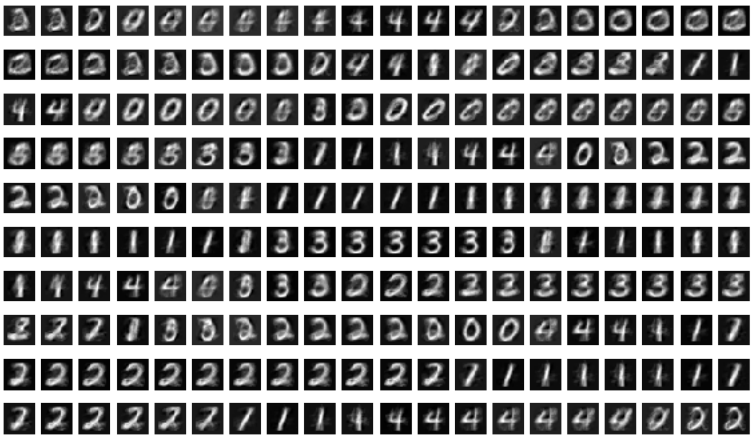

Figure 14: Sample Size: 35 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

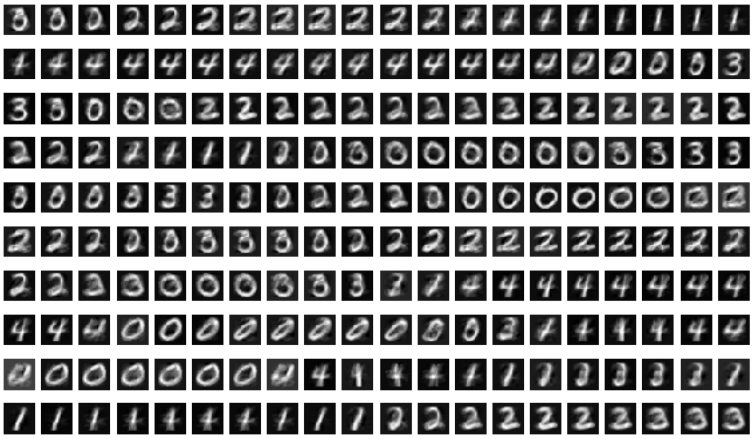

Figure 15: Sample Size: 40 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

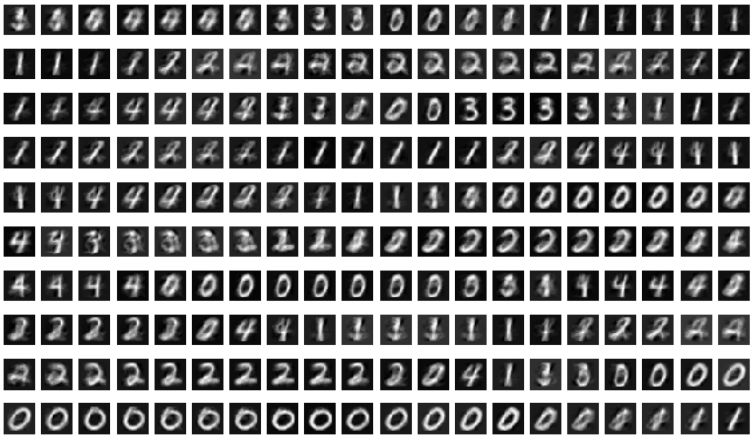

Figure 16: Sample Size: 45 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.

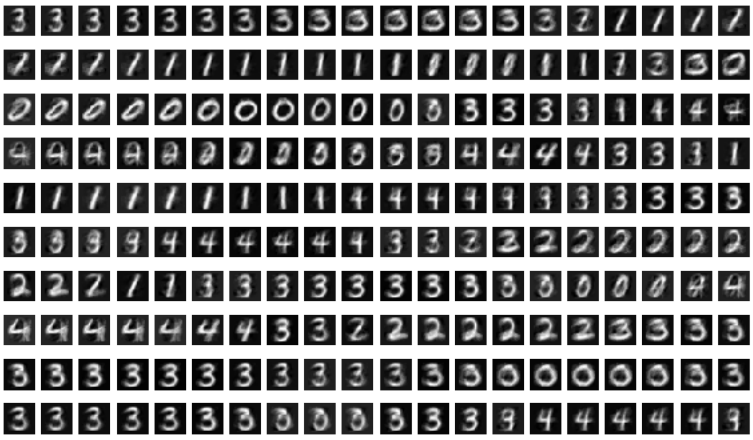

Figure 17: Sample Size: 50 examples for each {0,1,2,3,4}. 2 layers deep Gaussian process.