Variational continual learning for diffusion models

Charles London
Department of Computer Science
University of Oxford
Oxford, OX1 3QD
charles.london@cs.ox.ac.uk

Abstract

Diffusion models have recently emerged as the state-of-the-art for high-quality image generation, and the backbone of the majority of consumer-facing Imagen-style models. However, these models are time- and compute-intensive to train and usually have to be trained from scratch when adding new data or tasks, as they suffer from catastrophic forgetting. Continual learning would allow these models to add new tasks incrementally, without retraining on the entire dataset. By modelling diffusion models as likelihood-based generative models, we can use variational continual learning to learn new tasks without destroying previous generation capabilities. We find that this achieves significantly better results than naive online learning on the MNIST dataset, without requiring any data replay.

1 Introduction

Diffusion models have emerged as a powerful tool for generating high-quality images [18], forming the foundation of many state-of-the-art generative models. However, these models face a significant challenge when it comes to continual learning, as they are prone to catastrophic forgetting - the phenomenon where a model forgets previously learned information when trained on new data [12]. This limitation often necessitates retraining the entire model from scratch to incorporate new data, which is computationally expensive and time-consuming. In this paper, we propose a novel approach to address this issue by applying variational continual learning (VCL) [15] to diffusion models, enabling them to learn new tasks incrementally without sacrificing performance on previously learned tasks.

Our proposed method leverages the likelihood-based formulation of diffusion models [1, 9, 21] to incorporate VCL, allowing the model to update its posterior parameters for each task while minimizing the Kullback-Leibler divergence with the parameter distribution on previous tasks, minimizing the impact on its ability to generate samples from previously learned distributions. We evaluate our approach on the MNIST dataset [6] and compare its performance to experience replay, the current state-of-the-art methods for continual learning in diffusion models [3, 28]. The results demonstrate the potential of VCL as a promising direction for enabling continual learning in diffusion models, paving the way for more adaptable generative models.

2 Related work

Continual learning of diffusion models has so far largely focused on data replay methods, in which data from previous tasks is shown to the model when learning a new task to avoid catastrophic forgetting. These methods include experience replay [3], where some data from previous tasks is saved and replayed, and generative replay [14, 20], where a sequence of generators and discriminators are trained together, with the generator learning to generate data from both the new task and previous

Preprint. Under review.

tasks (using samples generated from the previous generator in the sequence). Zając et al. [28] found that experience replay achieves the best performance for diffusion models. This validates our approach, as VCL is shown to improve performance over using just experience replay (referred to as "coresets" in Nguyen et al. [15]).

Model merging is another way of enabling models to learn multiple tasks, by taking models of the same architecture trained on different tasks and merging the weights [11, 25]. This method has been so successful for diffusion models that a community of mergers has sprung up around Stable Diffusion [18]. However, for the best results, the models to be merged should have been fine-tuned from the same pre-trained base model [25]. Pre-training a base model is expensive, requiring enormous compute and data resources, somewhat limiting the applicability of this technique. An interesting question, which is beyond the scope of this paper, is whether a Bayesian perspective can be used to explain the efficacy of model merging, i.e. as a form of Bayesian model averaging [10].

3 Proposed approach

The goal of this study is to investigate whether VCL can be used to learn new tasks with diffusion models without catastrophic forgetting. We will compare the performance of VCL with and without experience replay to that of experience replay alone. Our experiments will be conducted on the MNIST dataset, using classifier uncertainty as an evaluation metric,[1] following the experimental setup of Nguyen et al. [15]. Our investigation will also cover the effect of the number of tasks on the performance of VCL, as well as the effect of the number of samples used for experience replay.

Our hypothesis is that VCL will mitigate catastrophic forgetting compared to naive online learning, and that VCL with experience replay will outperform experience replay alone. We also expect VCL's performance to degrade with more tasks and improve with larger experience replay samples.

Due to the time and compute-intensive nature of training diffusion models, our intention was to write all of our code in JAX [2], which consistently benchmarks as the fastest deep learning library [27]. However, due to time constraints, we only completed a replication of the VCL discriminative model experiments in JAX (see Appendix A). The diffusion model experiments were written in PyTorch [16], using code from Pearce [17] as a starting point.

4 VCL for diffusion models

This introduction to diffusion models is necessarily brief to comply with page limits; for an excellent pedagogical overview, see Bishop and Bishop [1]. There are different perspectives on diffusion models, including the score-based perspective [24], but in this paper we will adopt the likelihood-based perspective [1, 9, 21] as it allows us to easily demonstrate the applicability of VCL to these models.

Diffusion models are generative models made up of a forward process and a reverse process. The forward process is a series of T steps in which a sample $\mathbf{x}_0$ from the distribution to be modeled has noise gradually added to it by sampling $\mathbf{x}_t$ from a tractable Gaussian distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ for $1 < t < T$. This results in a sample $\mathbf{x}_T$ whose marginal distribution is Gaussian noise ($p(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_T|\mathbf{0}, \mathbf{I})$). The reverse process gradually denoises a Gaussian noise sample $\mathbf{x}_T$ to a sample $\mathbf{x}_0$ from the data distribution. The reverse distributions $p(\mathbf{x}_t|\mathbf{x}_{t+1})$ are intractable, so we approximate them with a learned model $q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w})$. Mathematically, these processes are expressed as

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t|\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \;\; q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{w}) = \mathcal{N}(\mathbf{x}_t|\mu(\mathbf{x}_t, \mathbf{w}, t), \beta_t\mathbf{I}), \tag{1}$$

where $\mu(\mathbf{x}_t, \mathbf{w}, t)$ is a neural network with parameters $\mathbf{w}$ that takes as input the sample $\mathbf{x}_t$ and the time-step t and outputs the mean of the distribution. Importantly, the same neural network is used for all time-steps, with the weights shared across time. The diffusion model can be trained by maximising

[1] Frechet Inception Distance (FID) is the current standard for evaluating generative models, but computing this metric accurately requires generating a minimum of 10,000 samples [7], which proved to be too computationally expensive for this project. We do not use the log-likelihood of the generated data for the same reason.

the evidence lower bound (ELBO) of the log-likelihood of the data (approximate maximum likelihood estimation [MLE] of $\mathbf{w}$), see equation 5 in Appendix B. In practice, Ho et al. [9] found that generated data quality can be improved by instead training a neural network $\mathbf{g}(\mathbf{x}_t, \mathbf{w}, t)$ to predict the noise level $\boldsymbol{\epsilon}_t$ (as a function of $\mathbf{x}_t$ and t) rather than the mean of the distribution. The metric used to train these models is simpler than the ELBO, taking the form

$$\mathcal{L}_{\text{MLE}}(\mathbf{w}; \mathbf{x}_0) = -\sum_{t=1}^{T} \|\mathbf{g}(\mathbf{x}_t, \mathbf{w}, t) - \boldsymbol{\epsilon}_t\|^2, \tag{2}$$

where the dependence on $\mathbf{x}_0$ is implicit in $\mathbf{x}_t$. While this new metric does not obviously correspond to the log-likelihood, Song et al. [23] proved that by adding a constant independent of $\mathbf{w}$ and correctly weighting each time-step, the metric lower bounds the log-likelihood. In our results, we will refer to this method as MLE training.

Given that our diffusion model training is approximately maximum likelihood estimation, we can apply VCL to the model by performing mean-field variational inference (MFVI), similarly to the variational autoencoder in Nguyen et al. [15]. We place a diagonal normal prior on our network parameters $q_d(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_d, \text{diag}(\boldsymbol{\sigma}_d^2))$, and the training metric for each task dataset $d = 1, \ldots, D$ becomes

$$\mathcal{L}_{\text{VCL}}^d(q_d(\mathbf{w})) = E_{q_d(\mathbf{w})}\left[\sum_{n=1}^{N_d} \mathcal{L}_{\text{MLE}}(\mathbf{w}; \mathbf{x}_{d(0)}^{(n)})\right] - \text{KL}(q_d(\mathbf{w})||q_{d-1}(\mathbf{w})). \tag{3}$$

Maximising this metric with respect to $\boldsymbol{\mu}_d$ and $\boldsymbol{\sigma}_d$ allows us to perform approximate Bayesian inference over the parameters of the diffusion model for each task, which should minimise catastrophic forgetting and allow the model to continue to generate high-quality samples from previous tasks.

To allow the model to distinguish between tasks, we use classifier-free diffusion guidance [8], and jointly train a conditional $\mathbf{g}(\mathbf{x}_t, \mathbf{w}, t, \mathbf{d})$ and unconditional $\mathbf{g}(\mathbf{x}_t, \mathbf{w}, t, \mathbf{0})$ diffusion model. The unconditional model is trained to generate samples from the data distribution of all tasks seen so far, while the conditional model takes in an embedding of the task label $\mathbf{d}$ and generates samples from the data distribution of the task. By taking a weighted combination of samples from the two models

$$\hat{\boldsymbol{\epsilon}}_d^t = (1+\alpha)\mathbf{g}(\mathbf{x}_t, \mathbf{w}, t, \mathbf{d}) - \alpha\mathbf{g}(\mathbf{x}_t, \mathbf{w}, t, \mathbf{0}), \tag{4}$$

we can generate samples from the data distribution of any task seen so far. Increasing α increases the weight of the sample conditioned on task d, generating more typical but less diverse samples from that task. Again following the lead of Nguyen et al. [15], who use a separate decoder head for each task, we use a separate set of trainable parameters for each task embedding $\mathbf{d}$ (but note that $\mathbf{g}$ and $\mathbf{w}$ are shared across tasks).

5 Experiments and results

For the experimental portion of this work we trained a U-Net-based diffusion model [9, 19] (see Appendix C) using both standard MLE training and VCL training, with and without experience replay. Our experience replay was implemented as a coreset of the data from the previous and current tasks. When we refer to a coreset of size S, this means we sampled S data-points uniformly at random from each of the previous tasks and the current task (so for task d the coreset is of size $d \times S$).[2] The models were trained on ten tasks/datasets in sequence, where each of the d tasks corresponds to the dth digit of the MNIST dataset, ranging from 0 to 9. We then evaluated the models by generating samples from each of the d tasks and measuring the classifier uncertainty of a classifier trained on the entire MNIST training dataset.

[2]The coreset method is the same as that Nguyen et al. [15], so for a full description see their paper.

We will only briefly comment on the training process here, but an exhaustive list of the hyperparameters used can be found in Table 2. We found that the VCL model was very sensitive to the parameter initialisation, particularly the log variance of the weight distribution. Following Swaroop et al. [26], we set the log variance to a very low initial value (-10.0), which was crucial for training stability. Due to the time- and compute-intensive nature of training diffusion models, we were limited to training the models for 25 epochs, and believe that image quality could be improved with longer training times. Diffusion models often benefit from longer training times [22], and Swaroop et al. [26] found that VCL models also benefit from training beyond loss convergence. Notably, the VCL model required far longer to train than the MLE model, even for the same number of epochs. On an NVIDIA RTX6000, the MLE model took ~ 1 hour to train on all 10 datasets, while the VCL model took ~ 5 hours.

(a) MLE. (b) VCL.

Figure 1: Generated images from models trained with standard diffusion MLE training and VCL training, with no coreset. Each row i shows generations from when the model had been trained on the first i tasks (digits), each column j shows generations for digit j. There is clear catastrophic forgetting in the MLE model, while the VCL model is able to generate reasonable samples from previous tasks. Images from models trained with different coreset sizes can be found in Appendix D.

The results of our classifier uncertainty experiments (Figure 2) show a clear benefit to using VCL with a small or no coreset. While the MLE model with no coreset has very high average uncertainty as the number of tasks grows, the VCL model maintains a low average uncertainty throughout ($\sim 100\times$ lower for 4 tasks or more). However, increasing the coreset size improves the performance of the VCL model far less than MLE model, such that by the time the coreset contains 200 datapoints per previous task, the VCL model is performing worse that the MLE model (though the gap is much less pronounced than with no coreset). A likely explanation for this is that the MNIST dataset is so simple that the MLE model can learn to create reasonable samples using only a few datapoints. Nguyen et al. [15] don't report any generative results using a coreset, but it is likely that the same thing would be observed for the variational autoencoder in their experiments. Classifier uncertainty does not evaluate sample variety (unlike FID score), so a model that can learn to generate one high-quality sample image per digit would achieve low classifier uncertainty, despite the lack of diversity in the generated images. Follow-up work could investigate the effect of coreset size on performance for more complex datasets, and using FID as a metric, where we would expect to see the VCL model outperform the MLE model until the coreset was much larger.

6 Discussion

In this work, we investigated the application of variational continual learning (VCL) to diffusion models, aiming to mitigate catastrophic forgetting when learning new tasks incrementally. Our experiments on the MNIST dataset demonstrated that VCL can significantly reduce catastrophic forgetting compared to naive online learning, maintaining low classifier uncertainty even as the number of tasks increases. However, we also found that increasing the size of the experience replay coreset improved the performance of the MLE model more than the VCL model, suggesting that the effectiveness of VCL may be less pronounced for simple datasets like MNIST.

While our results are promising, there are several limitations to our approach that should be addressed in future work. Firstly, the computational cost of training VCL models is significantly higher than MLE models, which may limit their practical applicability. Secondly, our experiments were limited to the relatively simple MNIST dataset, and it remains to be seen how well VCL performs on more

complex datasets. Future work should investigate the performance of VCL on more challenging datasets, such as ImageNet [5] or CIFAR-100 [13], and explore ways to reduce the computational cost of training VCL models, such as using more efficient architectures or training techniques. Additionally, it would be interesting to compare the performance of VCL to other continual learning methods for diffusion models, such as model merging, to better understand the relative strengths and weaknesses of each approach.

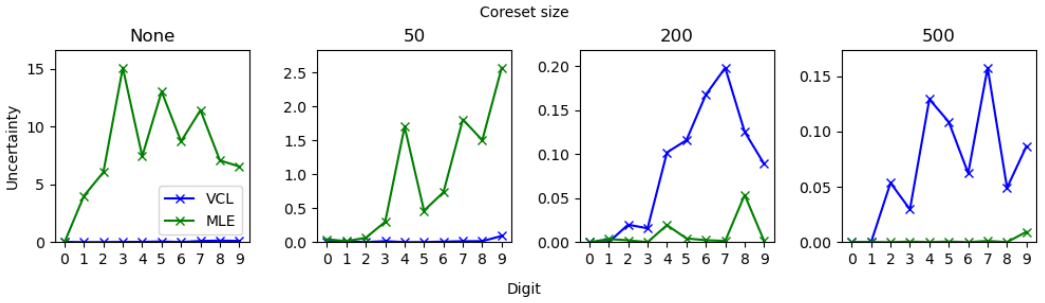


Figure 2: Average classifier uncertainty for all previously seen digits on samples generated from the MLE- and VCL-trained models with different coreset sizes (lower is better). Note that the y-axis takes a different range in each subplot, to better visualise the difference between the models. Figure 8 displays the same data on a log scale, and Appendix E contains per-digit uncertainty plots.

References

[1] Christopher M. Bishop and Hugh Bishop. *Deep Learning - Foundations and Concepts*. Springer, 2024. ISBN 978-3-031-45467-7. doi: 10.1007/978-3-031-45468-4. URL `https://doi.org/10.1007/978-3-031-45468-4`.

[2] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[3] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning, 2019.

[4] Nikolaos Kosmas Chlis. Image segmentation with monte carlo dropout unet and keras, Oct 2019. URL `https://nchlis.github.io/2019_10_30/page.html`.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL `https://doi.org/10.1109/CVPR.2009.5206848`.

[6] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[8] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[10] Jennifer A Hoeting, David Madigan, Adrian E Raftery, and Chris T Volinsky. Bayesian model averaging: a tutorial. *Statistical science*, 14(4):382–417, 1999.

[11] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023.

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

[13] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

[14] Sergi Masip, Pau Rodriguez, Tinne Tuytelaars, and Gido M. van de Ven. Continual learning of diffusion models with generative distillation, 2023.

[15] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning, 2018.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[17] Tim Pearce. Conditional diffusion MNIST, 2022. URL https://github.com/TeaPearce/Conditional_Diffusion_MNIST.

[18] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[20] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay, 2017.

[21] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[22] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[23] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models, 2021.

[24] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

[25] Yi-Lin Sung, Linjie Li, Kevin Lin, Zhe Gan, Mohit Bansal, and Lijuan Wang. An empirical study of multimodal model merging, 2023.

[26] Siddharth Swaroop, Cuong V. Nguyen, Thang D. Bui, and Richard E. Turner. Improving and understanding variational continual learning, 2019.

[27] Keras Team. Keras documentation: Keras 3 benchmarks, 2024. URL https://keras.io/getting_started/benchmarks/.

[28] Michał Zając, Kamil Deja, Anna Kuzina, Jakub M. Tomczak, Tomasz Trzciński, Florian Shkurti, and Piotr Miłoś. Exploring continual learning of diffusion models, 2023.

A Discriminative model experiments replication in JAX

We successfully replicated the discriminative model experiments from Nguyen et al. [15] in JAX. The model was trained on the split MNIST dataset, with the same experimental setup as in the original paper. We found that the JAX implementation was significantly faster than the Tensorflow (TF) implementation used in their paper, with the JAX model training in ~ 30 minutes compared to ~ 4 hours for the TF model (on the CPU). We were unable to train the TF model on the GPU as TF for Python 2 no longer provides compatible CUDA binaries. The results of the JAX model were consistent with the original paper, demonstrating the effectiveness of VCL for continual learning in discriminative models.

Table 1: Training times for the discriminative split MNIST experiment in JAX and TF (hh:mm).

	JAX	TF
CPU	00:30	04:04
GPU	00:05	–

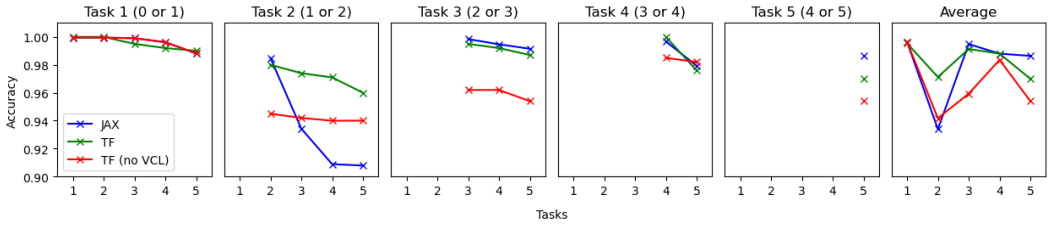


Figure 3: Test set accuracy on all tasks for the Split MNIST experiment using a random coreset of size 40. The last column shows the average accuracy across all tasks.

B ELBO for likelihood-based diffusion models

The ELBO for the likelihood-based diffusion model is given by [1]

$$\mathcal{L}(\mathbf{w}; \mathbf{x}_0) = E_{p(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\sum_{t=2}^{T} \ln \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{w})}{p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)} + \ln q(\mathbf{x}_0|\mathbf{x}_1, \mathbf{w}) \right]. \tag{5}$$

C Model architecture and hyperparameters

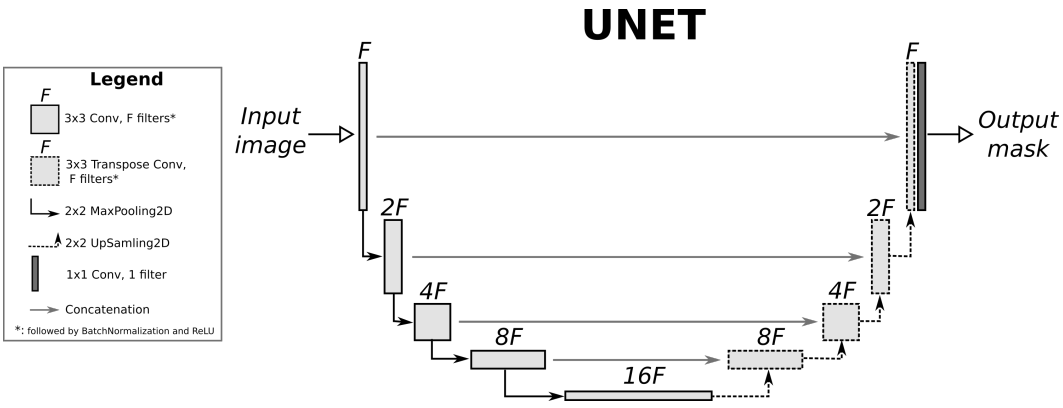


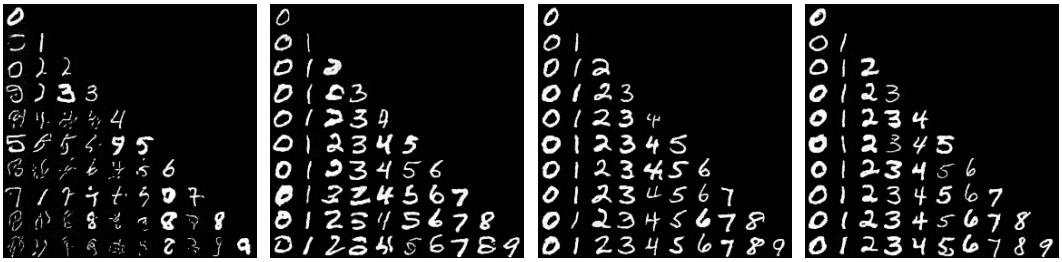
Figure 4: The U-Net architecture used for the diffusion model. Image reproduced from Chlis [4].

The U-Net architecture used as the base for the diffusion model is very similar to the one in Figure 4, with the main difference being that we also provide the time t and task embedding $\mathbf{d}$ as input to the model. The time is passed through a simple feedforward neural network to create a time embedding. As the image embedding is being passed into the decoder portion of the U-Net, it is multiplied by the task embedding, and the time embedding is added to it.

Table 2: Hyperparameters used in the experiments.

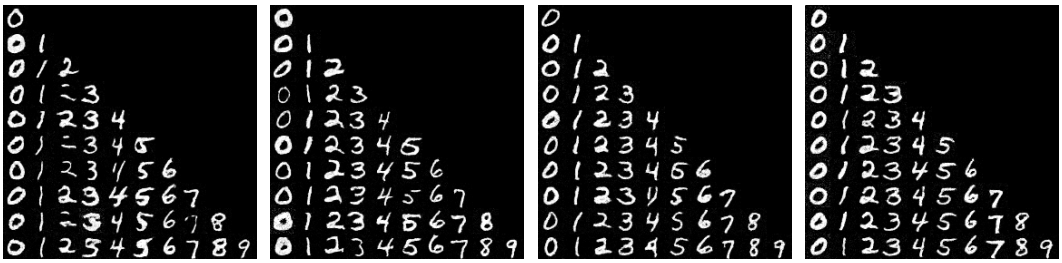
Hyperparameter	Value
Shared	
Epochs for dataset training	25
Epochs for coreset training	10
Diffusion timesteps	400
Batch size	128
# of hidden channels	128
Learning rate	0.0001
Weight initialisation (MFVI mean)	$\mathcal{U}(-\sqrt{15/\text{fan-in}}, \sqrt{15/\text{fan-in}})$
Sample combination weight α	2.0
VCL	
MFVI log variance initialisation	-10.0
# of training parameter samples	10
# of eval parameter samples	10

D Additional generated images



(a) MLE - no coreset. (b) MLE - 50. (c) MLE - 200. (d) MLE - 500.

Figure 5: Generated images from MLE trained diffusion models with different coreset sizes.



(a) VCL - no coreset. (b) VCL - 50. (c) VCL - 200. (d) VCL - 500.

Figure 6: Generated images from VCL trained diffusion models with different coreset sizes.

Figure 7: Generated images from an MLE trained diffusion model on the entire MNIST training dataset.

E Additional classifier uncertainty plots

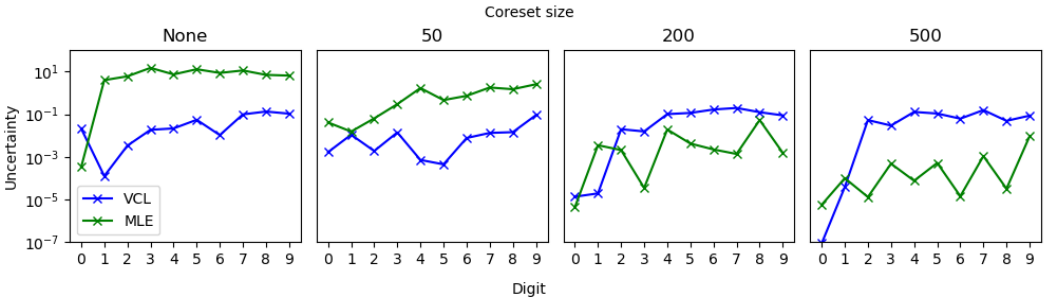


Figure 8: Average classifier uncertainty (log scale) for all previously seen digits on samples generated from the MLE- and VCL-trained models with different coreset sizes (lower is better).

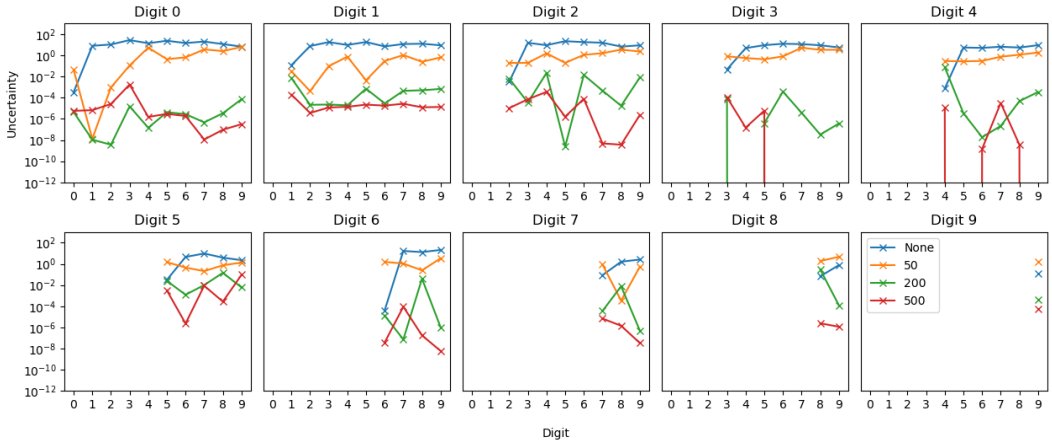


Figure 9: Classifier uncertainty (log scale) for each digit after seeing previous digits on samples generated from the MLE-trained models with different coreset sizes (lower is better).

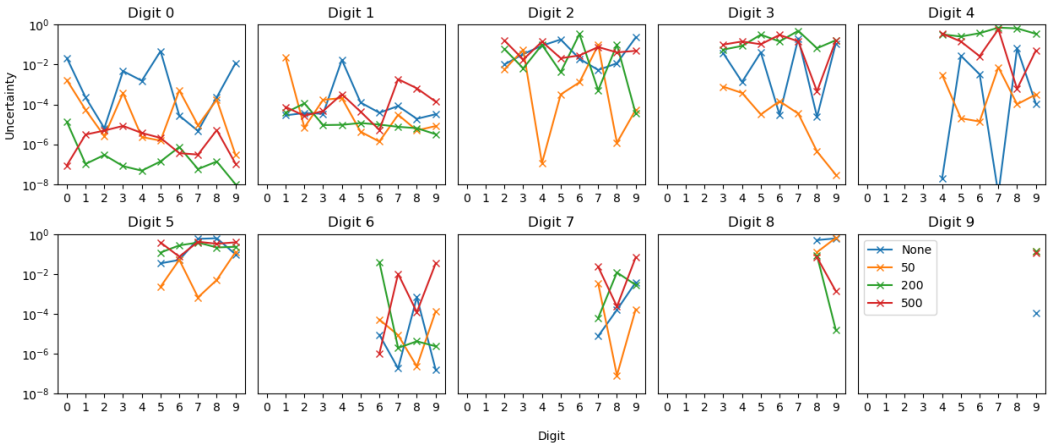


Figure 10: Classifier uncertainty (log scale) for each digit after seeing previous digits on samples generated from the VCL-trained models with different coreset sizes (lower is better).

In Figures 9 and 10, we can see that the uncertainty of the MLE model almost always decreases with a larger coreset, for all digits. The picture is not nearly as clear for the VCL model, with the uncertainty for some digits increasing with a larger coreset. However, inspecting the images generated in Figure 6, the image samples look significantly clearer for the VCL model with a coreset of 500 than 200. Again, these observations suggest that classifier uncertainty is not a perfect measure of sample quality, and that the MNIST dataset is so simple that even relatively poor samples have very low classifier uncertainty.