Diffusion Models for Synthetic Data Generation

Dec 15, 2022

Johan Ødum Lundberg
ITU
jolu@itu.dk

Abstract

In this work, I present a study on using synthetic data, sampled from a denoising diffusion probabilistic model, to train a ResNet model for image classification. I show that the ResNet model can be trained on a large number of synthetic images and achieve good results, although not as good as using real data. I also demonstrate that the performance of the classifier depends on the complexity of the images and the amount of available data. Finally, I demonstrate that adding a small amount of real data to a synthetic data set can improve the performance of the model. Overall, my findings suggest that synthetic data can be a valuable addition to real data for training image classification models. For my code, see Github.

1 Introduction

The denoising diffusion probabilistic model (Sohl-Dickstein et al., 2015) (DDPM) is a new type of generative model that uses a two-step process to produce samples from a given data distribution. In the first step, known as the *forward process*, Gaussian noise is gradually added to an input $\mathbf{x}_0$, resulting in a distribution $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$. A sample is then drawn from this distribution. In the second step, known as the *reverse process*, the DDPM uses a parameterized Markov chain to predict the noise that was added in the forward process and generate a data distribution $p_\theta(\mathbf{x}_{0:T})$. This process is illustrated in Figure 1.

The recent advances in the generative capabilities of generative models have spurred research into their possible use for image classification models as these are often rely on the expensive task of gathering and annotating data sets. Previous methods to improve image classification models include data augmentation (Perez and Wang, 2017), pretraining using real data and training on synthetic data generated from a data generation pipeline (Hinterstoisser et al., 2018) and using a graphic simulator to generated labelled data and mixing training data with synthetic data (Tremblay et al., 2018). Previous research using generative models has demonstrated that these methods, such as GANs ((Besnier et al., 2020); (Ravuri and Vinyals, 2019)), and BigGANs (Brock et al., 2018), can produce highly realistic images at a resolution of 512x512, indicating that generative models are capable of accurately capturing the data distribution $p(\mathbf{x})$.

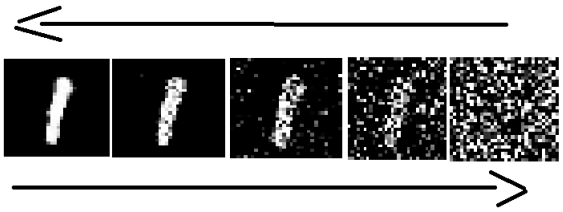

Figure 1: DDPM reverse and forward process.

Recent developments in DDPMs have demonstrated their ability to generate high-quality synthetic images (Ho et al., 2020) that are on par with GANs in terms of image quality (Dhariwal and Nichol, 2021) and mode coverage ((Kingma et al., 2021); (Xiao et al., 2021)). This makes it worthwhile to investigate whether DDPMs can be used to enhance the performance of an image classification model.

In this paper, I study the use of synthetic data generated by a DDPM to train a classifier. I compare the results of training the classifier using different amounts of real and synthetic data on the MNIST (Ciresan et al., 2011) and CIFAR-10 (Krizhevsky, 2012) data sets. I also investigate the effect of varying the parameters of the DDPM on the quality of the generated images. I evaluate the DDPM qualitatively by inspecting the generated images, and quantitatively by training a ResNet

model on the synthetic data and evaluating its performance on a hold-out test set using the f1 score.

I find that:

- The ResNet model can be trained with a large number of synthetic images and achieve good results, although not as good as using real data.
- The performance of the classifier depends on the complexity of the images and the amount of available data.
- Adding a small amount of real data to a synthetic data set can improve the performance of the model.

2 Related Work

In a recent study, (He et al., 2022) showed that synthetic data generated using the text-to-image generation model CLIP (Radford et al., 2021) can improve performance for zero-shot and few-shot image classification tasks, as well as for model pretraining. In fact, their results showed that using synthetic data for pretraining can outperform pretraining on the ImageNet (Deng et al., 2009) dataset. Furthermore, they found that training a model from scratch using synthetic data can achieve the same level of accuracy as training from scratch using real data, but with only a fraction of the data. Other researchers have begun using DDPMs to improve classifiers for thoracic abnormality and skin disease detection (Sagers et al.; Packhäuser et al., 2022).

3 Methods

In this section, I review the fundamental concepts of DDPMs that are relevant to my work, based on the research of (Sohl-Dickstein et al., 2015), (Ho et al., 2020), (Nichol and Dhariwal, 2021), (Dhariwal and Nichol, 2021), (Ho and Salimans, 2022). I then describe the implementation details of the DDPM and ResNet classifier, and provide details of the experiments I conduct to explore the impact of DDPMs on image classification. Specifically, I investigate the effect of changing the image size and number of time steps when training the DDPM, and compare the quality of the generated images when trained on different amounts of data. Finally, I evaluate the performance of a ResNet classifier by training it using different combinations of real and synthetic training data.

3.1 Denoising Diffusion Probabilistic Model

A diffusion probabilistic model (Sohl-Dickstein et al., 2015; Ho et al., 2020) is a latent variable generative model of the form $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T})d\mathbf{x}_{1:T}$ defining a forward and reverse Markov process consisting of T time steps. Given a data distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ as input, the forward diffusion process produces a joint distribution of latent variables $\mathbf{x}_1, \ldots, \mathbf{x}_T$ by adding noise at time step t using a variance schedule $\beta_t \in [0, 1]$ as follows:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t\geq 1} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{1}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}). \tag{2}$$

The variance schedule controls the amount of noise added at time step t and is defined such that $\beta_1 < \beta_2 < \cdots < \beta_{T-1} < \beta_T$. Given a well defined schedule, the forward process converts $\mathbf{x}_0$ into a standard Gaussian $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$. To sample $\mathbf{x}_t$ at an arbitrary time step $t \sim U(1, T)$ of the forward process, using the notation $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, equation 2 can be redefined as

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}). \tag{3}$$

The reverse denoising process samples from $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$, approximates each step of the chain using a neural network parameterized by θ and generates a target distribution as follows:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t\geq 1} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{4}$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)). \tag{5}$$

Equation 5 requires that we learn μ_θ and $\Sigma_\theta(\mathbf{x}_t, t)$. However, if we parameterize equation 3 as $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, μ_θ can be estimated as

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_\theta(\mathbf{x}_t, t)). \tag{6}$$

This means that the neural network would have to learn the noise ϵ_θ that was added.

To learn $\Sigma_\theta(\mathbf{x}_t, t)$, (Nichol and Dhariwal, 2021) shows that a scalar v can be learned during training to produce a linear interpolation of the minimum

and maximum beta $\Sigma_\theta(\mathbf{x}_t, t) = exp(v\, log\, \beta_t + (1-v) log\, \tilde{\beta}_t$. However, I choose to follow (Ho et al., 2020) who showed that we can use a diagonal covariance matrix $\sigma_t^2\mathbf{I}$ with σ_t^2 predefined as β or $\tilde{\beta}$ without a loss in image quality. Meaning $\mathbf{x}_{t-1}$ would be calculated as $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t)) + \sigma_t\epsilon$. As such, to produce a less noisy image, the neural network only has to learn the noise ϵ_θ. (Ho et al., 2020) showed that training of the neural network worked best, in terms of sample quality, by predicting the noise added during the forward step and minimizing the loss

$$L = \mathbb{E}_{t,\mathbf{x}_0,\epsilon}\left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2\right] \quad (7)$$

where $t \sim U(0,T)$, $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The training algorithm is shown in table 1. To sample from the model, an image consisting of random noise is sampled from a standard Gaussian. This image is then slowly denoised by subtracting $\epsilon_\theta(\mathbf{x}_t, t)$ scaled by a term defined by the parameters of the DDPM. At each step new noise is added to the image scaeld by σ_t. At the very last step the model simply returns the mean without adding new noise to the image. The algorithm can be seen in table 2.

Table 1: DDPM training algorithm.

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim$ Uniform$(1, \dots, T)$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient step on
    $\nabla||\epsilon - \epsilon_\theta((\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2$
until converged

Table 2: DDPM sampling algorithm.

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if t > 1, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x_{t-1}} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x_t}, t)) + \sigma_t\mathbf{z}$
end for
return $\mathbf{x}_0$

To control the data the DDPM generates, requires learning the conditional distribution $p(\mathbf{x}|y)$. This is often done using either classifier guidance (Dhariwal and Nichol, 2021) or classifier free guidance (Ho and Salimans, 2022).

Classifier guidance is defined as $p_{\theta,\phi}(\mathbf{x}_t|\mathbf{x}_{t+1}, y) = Z p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}) p_\phi(y|\mathbf{x}_t)$ where Z is a normalizing constant, θ is the parameters of the diffusion model and ϕ represents the parameters of a classifier trained to predict the label given a noisy image.

Classifier free guidance involves training the diffusion model with and without the labels some percentage of the time. The noise estimate ϵ_θ is then calculated as a linear combination of the conditional and unconditional noise estimates $\hat{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) = (1+w)\epsilon_\theta(\mathbf{x}_t, \mathbf{c}) - w\epsilon_\theta(\mathbf{x}_t)$. The advantages of using classifier-free guidance includes not having to learn a separate classifier and the ease of implementation.

3.2 Experiments

To investigate the impact of DDPMs on image classification, I conduct two preliminary experiments. In the first experiment, I explore the effect of changing the image size and number of time steps when training the DDPM. I train and validate the DDPM on the CIFAR-10 train set, using 90% of the data for training and 10% for validation.

In the second preliminary experiment, I train two DDPMs using different amounts of the CIFAR-10 data set. One DDPM uses half of the CIFAR-10 data set while the other uses the entire data set. I compare the resulting quality of the synthetically generated images by both DDPMs. For both models, I use 90% of the available training data for training and 10% for validation.

Finally, to evaluate the effects of using synthetic training data when training a ResNet model, I conduct five different experiments using the MNIST and CIFAR-10 data sets. In each experiment, the ResNet model is trained using a combination of real and synthetic training data. I use 50% of the original training data for each data set to train a DDPM to generate the synthetic training data. The remaining 50% of the data is used to train the ResNet model. The details of the data splits for each experiment are shown in table 3, and the ratio of real to synthetic data used for each experiment is shown in table 5. The data used for validation of the ResNet model is 10% of the available real data. The synthetic data sets generated by the DDPM consist of 26 880 images for MNIST and 22 400 images for CIFAR-10.

This final experiment allows me to evaluate the performance of the ResNet model when trained using a combination of real and synthetic data, and compare it to the performance of the ResNet model trained using only real data. I will quantitatively evaluate the performance of the ResNet model us-

Table 3: This table shows the percentage of a data set used for each model with the train/validation splits and number of images in the data set.

Model	Data Set	% of data set available for training and validation	Train/Val Split	# images in total
DDPM	MNIST	50%	90%/10%	30 000
DDPM	CIFAR-10	50%	90%/10%	25 000
ResNet	MNIST	50%	90%/10%	30 000
ResNet	CIFAR-10	50%	90%/10%	25 000

ing the f1 score on a hold-out test set.

In all experiments the images were scaled to be between [-1, 1], see table 4 for details.

Table 4: MNIST and CIFAR-10 train and test sets showing the number of images in each data set and their respective image sizes and image channels used during training.

Data set	# imgs	img size	img channels
MNIST train set	60k	32	1
MNIST test set	10k	32	1
CIFAR-10 train set	50k	32	3
CIFAR-10 test set	10k	32	3

Table 5: Ratio of real to synthetic data used in each training experiment for each data set.

Experiment	Real Data	Synthetic Data
Exp 1	100%	0%
Exp 2	0%	100%
Exp 3	50%	50%
Exp 4	10%	90%
Exp 5	90%	10%

3.3 Implementation Details

In this subsection, I describe the implementation details of the DDPM and the ResNet classifier, which were both implemented in PyTorch. Both models had a maximum number of training epochs of 100, and would stop training if the validation loss did not decrease for 10 epochs.

I followed the approach of (Ho et al., 2020) to implement a DDPM with a linear beta schedule ranging from 0.0001 to 0.02, and set the value of σ^2 equal to β. The loss was minimized by comparing the predicted noise with the added noise, as shown in equation 7. Unlike (Ho et al., 2020), who used $T = 1000$, I set $T = 400$, as this improved the results.

To guide the diffusion process I train the model using classifier-free guidance (Ho and Salimans, 2022) with the labels 0.9 percent of the time and without labels 0.1 percent of the time. The classifier-free guidance scale was set to 3 to balance image-quality and the guidance, and was not changed in any experiments. The model was trained using automatic mixed precision, with a learning rate of 0.0005, a batch size of 64, and was optimized using the Adam optimizer.

To predict the noise, I used a U-Net (Ronneberger et al., 2015) with four levels and skip connections of dimensionality 64, 128, 256, and 512. Each level consisted of ResNet blocks with a time-embedding layer, self-attention, and pooling/-ConvTranspose layers.

The ResNet classifier was based on the approach of (He et al., 2016), using ResNet blocks and attention layers. The final two layers were a global average pooling layer across the last dimension (height × width), followed by a linear layer and a softmax activation function. The classifier was trained using cross-entropy loss, automatic mixed precision, a learning rate of 0.0005, a batch size of 64, and was optimized using the Adam optimizer.

4 Results

From figure 3, it can be seen that the DDPM was able to learn to generate new digits. These images were sampled during training at epoch 10, which was the first generation of images without obvious mistakes. Interestingly, as the model began to converge (as shown in figure 4), the quality of the generated images started to decrease, indicating that minimizing the loss does not necessarily result in higher quality images (see appendix A for further details).

Figure 3: MNIST digits generated from DPM during training at epoch 10 using img size 32 and T=400.

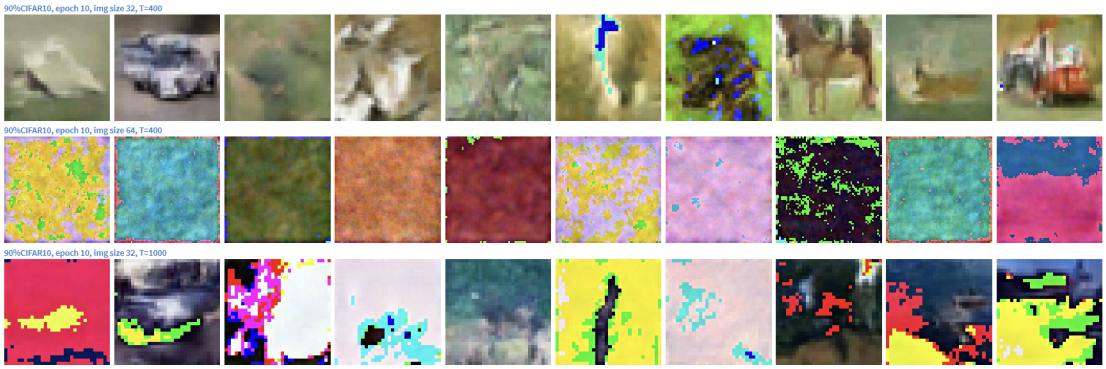

Figure 2: Images generated during training at epoch 10 from a DDPM trained on 90% of the CIFAR-10 training data Top row: image size=32 and T=400. Middle row: image size=64, T=400. Bottom row: image size=32, T=1000. Image clases from left to right: plane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

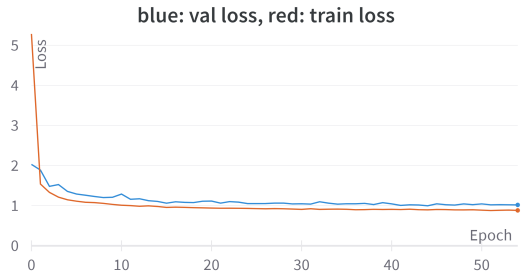

Figure 4: DDPM MNIST train and validation loss.

The results of the experiments in Figure 2 show that using an image size of 32x32 with $T = 400$ is the optimal parameter for the DDPM. This is in contrast to the findings of (Ho and Salimans, 2022), who discovered that increasing the number of time steps improved the quality of the samples. The top row of the figure demonstrates that this parameter allows the model to learn, with the images showing less stark colors and the beginnings of recognizable objects such as a car and a deer. In contrast, the middle row shows that resizing the images to 64x64 makes it difficult for the diffusion model to learn, likely due to the loss of detail that occurs during the resizing process. The third row of the figure shows that using an image size of 32x32 with $T = 1000$ results in less noisy images, but they still lack structure and have overly bright colors. Based on these findings, I will now investigate the effect of training the DDPM with less data using the optimal parameter settings.

The experiments shown in Figure 5 demonstrate that using half of the data set for training results in lower-quality samples and faster model convergence. The top row of the figure shows that the images generated using only half of the data set have unnatural colors and appear blurry and unstructured. Despite this, some objects are starting to become recognizable, such as a horse and a boat in the third and second rightmost images. The bottom row, on the other hand, shows that using the entire data set produces samples of higher quality that are less blurry and have a more defined structure. This suggests that while using less data may result in faster convergence, it ultimately hinders the overall quality of the samples.

The results in table 6 suggest that the ResNet model is unable to learn effectively when trained on CIFAR-10 data, as indicated by the low f1 score for experiment 1. This may be due to the limited amount of training data available. Previous experiments using the full CIFAR-10 dataset yielded better results for both CIFAR-10 and MNIST. The poor performance in experiment 1 is reflected in the results of experiments 2, 3, 4, and 5, which all use synthetic data.

For the experiments on MNIST, the ResNet model was able to learn effectively using both real and synthetic data. However, using only synthetic data yielded lower performance compared to using only real data or a combination of real and synthetic data. This suggests that the quality of the generated synthetic images is not sufficient. Experiment 4 shows that this issue can be addressed by including some real images in the training set.

5 Conclusions

I have demonstrated that it is possible to train a ResNet using synthetic data generated from a denoising diffusion probabilistic model. However, achieving results comparable to using real data re-

(a) DPPM trained on 90% of the 50% of CIFAR-10 training data set. Images sampled after 38 epochs.

(b) DDPM trained on 90% of CIFAR-10 training data set. Images sampled after 62 epochs.

Figure 5: DPPM image generations with image size 32 and T=400. Images were sampled after model convergence. Classe are (left to right): Plane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

Table 6: CIFAR-10 and MNIST Experiments.

Experiment	% real images	% synthetic images	CIFAR10 test set f1 score	MNIST test set f1 score
Exp1	100%	0 %	0.69	0.97
Exp2	0 % %	100 %	0.36	0.71
Exp3	50 %	50 %	0.50	0.98
Exp4	10 %	90 %	0.44	0.95
Exp5	90 %	10 %	0.36	0.99

quires a larger amount of data or the addition of a small amount of real data. Additionally, the effectiveness of using synthetic data diminishes as the complexity of the images increases, requiring even more data to achieve good performance.

6 Future Work

In my work on diffusion models, I have focused on a limited set of parameters. It could be beneficial to study the impact of varying the classifier-free guidance scale parameter, as it has been shown to affect the quality of the final images (Ho and Salimans, 2022). In my final experiment, I split the data set into two parts for the DDPM and classifier. However, in a real-world scenario, it would be more practical to train the DDPM on the full data set and the classifier on the full data set as well as the synthetic data set.

References

Victor Besnier, Himalaya Jain, Andrei Bursuc, Matthieu Cord, and Patrick Pérez. 2020. This dataset does not exist: Training models from generated images. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Dan C. Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. 2011. High-performance neural networks for visual object classification. *ArXiv*, abs/1102.0183.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. 2022. Is synthetic data from generative models ready for image recognition? *arXiv preprint arXiv:2210.07574*.

Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. 2018. On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. 2021. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707.

Alex Krizhevsky. 2012. Learning multiple layers of features from tiny images. *University of Toronto*.

Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR.

Kai Packhäuser, Lukas Folle, Florian Thamm, and Andreas Maier. 2022. Generation of anonymous chest radiographs using latent diffusion models for training thoracic abnormality classification systems. *arXiv preprint arXiv:2211.01323*.

Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Suman Ravuri and Oriol Vinyals. 2019. Seeing is not necessarily believing: Limitations of bigGANs for data augmentation.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Luke William Sagers, James A Diao, Matthew Groh, Pranav Rajpurkar, Adewole Adamson, and Arjun Kumar Manrai. Improving dermatology classifiers across populations using images generated by large diffusion models. In *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. 2021. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*.

A

Figure 6: MNIST digits generated during training of diffusion model at epoch 40.