Class Embeddings Enter Class-conditional Diffusion Models

Chuanyang Jin [1] Peiqi Liu [1] Zixuan Dong [1]

Abstract

Guided diffusion incorporates class information into the diffusion models to trade off mode coverage and sample fidelity. To perform the guidance without a classifier, classifier-free diffusion guidance is recently introduced where a conditional and an unconditional diffusion model are jointly trained. However, it remains unclear how to embed class-specific information into the conditional training process. Thus, we propose several potential class embedding patterns such as Uniform embedding, Pyramid embedding, and Bottleneck embedding under the framework of Class Embedding Networks (CEN), which explicitly generate learnable mappings from the class labels. During the reverse diffusion process, the generated mappings from CENs are concatenated with each layer of the noise estimator. The concatenation of class embeddings allows the noise estimator to be class conditional. In our experiment, we evaluated different embedding patterns and trade-offs between Pyramid embedding and Bottleneck embedding. In the end, we discuss about our attempt to use the noisy-label dataset to perform class-conditional diffusion.

1. Introduction

In the past several years, deep generative models have achieved great success in the area of image generation, natural language processing, and speech recognization. Particularly, in the task of generating high-fidelity images, the Generative Adversarial Network (GAN) (Goodfellow et al., 2014) uses a generator-discriminator structure, where the generator tries to produce high-quality fake images to confuse the discriminator, while the discriminator wants to distinguish between those generated fake images and the ground-truth source images. However, there are two major downsides to the GAN model. First, the model structure entails that either the generator part or the discriminative part is trained against a nonstatic adversary. As a result, the model often collapses or needs very careful hyperparameter tuning. Second, due to the design of the loss function, although the generated images have high fidelity compared with the source images, the generation of the same source image cannot give diverse results varying on the local details.

Recently, the Denoising Diffusion Probabilistic Model (DDPM) (Ho et al., 2020), which is also a deep generative model, has been shown to outperform the GAN model in the sense that it generates images with both high fidelity and diversity. Typically, the DDPM has two processes. In the forward noising pass, it keeps adding Gaussian noises to the input image, and eventually, the corrupted image will converge to a Gaussian distribution. Then, in the reverse denoising process, the model wants to undo the first process by estimating the noise added in each step and then sampling an image in the previous time step. In terms of the model architecture, the DDPM employs the U-Net (Ronneberger et al., 2015) to estimate the noise. With a symmetric architecture (inputs and outputs are of the same spatial size), the U-Net consists of WideResNet encoder and decoder blocks with skip connections between them, group normalization, and self-attention blocks. One remarkable breakthrough where the DDPM beats the performance of the GAN model is made by the Classifier-guidance DDPM (Dhariwal & Nichol, 2021). The novelty of this method is to add gradient guidance computed from a trained classifier to the estimated mean of the distributions during the sampling process. By manipulating the strength of the guidance, the Classifier-guidance DDPM can trade off image fidelity and diversity. As a result, a well-chosen classifier-guidance strength can enable the model to generate real images with rich detail diversity.

However, if one does not have access to a classifier, can we still boost the performance of the diffusion models while trading off the image fidelity and diversity? This comes to the Classifier-free DDPM (Ho & Salimans, 2022). Different from the Classifier-guidance DDPM, the model is now conditioned on the image labels with some probability during training. And when we use the trained model to sample images, the sampling distributions in each time step are estimated through a linear combination of the conditioned and unconditioned noise predictions. Note that here the class

[*] Equal contribution [1] New York University. Correspondence to: Chuanyang Jin <cj2133@nyu.edu>.

Preliminary work.

label serves as guidance during the sampling process and by manipulating the coefficients of the conditioned and unconditioned noise estimations, we can trade off the fidelity and diversity as the classifier-guidance DDPM. In this project, we wonder how different structures of image class embedding affect the performance of the classifier-free DDPM. Therefore, in the rest of this report, we proceed with our discussion as follows. In section 2, we briefly summarize the math background of the diffusion models. Then, section 3 elaborates on the algorithmic and architectural details of the classifier-free DDPM. After that, we show our experimental results on different class embedding structures in section 4. Additionally, we share our experience of using a noisy-label dataset to train a class-conditional diffusion model in section 5. In the end, we provide some further discussions and our conclusion.

2. Background

Our researches belong to the large family of Diffusion models (Sohl-Dickstein et al., 2015). The models are inspired by *diffusion process*, which is a kind of random process $(X_t)_{t\geq 0}$ that pictures the location of a particle moving at random but also governed by a drift and a random noise. Unlike other generative models which use discriminators (GAN) or encoders (VAE), etc., diffusion models use a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \ldots, \beta_T$ as the approximate posterior $q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)$, called the *forward diffusion process*:

$$
\begin{aligned}
q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) &:= \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right), \\
q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) &:= \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)
\end{aligned}
\tag{1}
$$

This way, the sample $\mathbf{x}_0$ gradually loses its features and as $T \to \infty$, $\mathbf{x}_T$ becomes an isotropic Gaussian distribution. The variances β_t can be learned by reparameterization or held constant as hyperparameters, and expressiveness of the reverse process is ensured in part by the choice of Gaussian conditionals in $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$, because both processes have the same functional form when β_t are small.

Now we want to reverse the above process to recreate the sample $\mathbf{x}_0$ from the heavily noised $\mathbf{x}_T$. The model then predicts the distriubtion $p_\theta\left(\mathbf{x}_0\right) := \int p_\theta\left(\mathbf{x}_{0:T}\right) d\mathbf{x}_{1:T}$, where $\mathbf{x}_1, \ldots, \mathbf{x}_T$ are latents of the same dimensionality as the data $\mathbf{x}_0 \sim q\left(\mathbf{x}_0\right)$. The joint distribution $p_\theta\left(\mathbf{x}_{0:T}\right)$ is the *reverse process* as a Markov chain with learned Gaussian transitions starting at $p\left(\mathbf{x}_T\right) = \mathcal{N}\left(\mathbf{x}_T; \mathbf{0}, \mathbf{I}\right)$:

$$
\begin{aligned}
p_\theta\left(\mathbf{x}_{0:T}\right) &:= p\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right), \\
p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) &:= \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)
\end{aligned}
\tag{2}
$$

3. Method

In this section, we will first summarize how the classifier-free guidance diffusion model is derived. Next, we will propose different methods to insert class embeddings into the conditional Diffusions model to make it class conditional, which is an implementation detail missed in the original paper.

Algorithm 1 Joint training a diffusion model with classifier-free guidance (Ho & Salimans, 2022)

```
Require: $p_{\text{uncond}}$: probability of unconditional training
repeat
    $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ {Sample data with conditioning from the dataset}
    $c \leftarrow \varnothing$ with probability $p_{\text{uncond}}$ {Randomly discard conditioning to train unconditionally}
    $\lambda \sim p(\lambda)$ {Sample log SNR value}
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\sigma}$ {Corrupt data to the sampled log SNR value}
    Take gradient step on $\nabla_\theta \|\boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon}\|^2$ {Optimization of denoising model}
until converged
```

Algorithm 2 Conditional sampling with classifier-free guidance (Ho & Salimans, 2022)

```
Require: $w$: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \ldots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}, \lambda_T = \lambda_{\max}$
$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = 1, \ldots, T$ do
    $\tilde{\boldsymbol{\epsilon}}_t = (1+w)\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t, \mathbf{c}\right) - w\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t\right)$ {Sampling step (could be replaced by another sampler, e.g. DDIM)}
    $\tilde{\mathbf{x}}_t = \left(\mathbf{z}_t - \sigma_{\lambda_t}\tilde{\boldsymbol{\epsilon}}_t\right) / \alpha_{\lambda_t}$
    $\mathbf{z}_{t+1} \sim \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), \left(\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t}\right)^{1-v}\left(\sigma^2_{\lambda_t|\lambda_{t+1}}\right)^{v}\right)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$
end for {Form the classifier-free guided score at log SNR $\lambda_t$}
return $\mathbf{z}_{T+1}$
```

3.1. Classifier-Free Guidance Diffusions

Recently some researchers are attempting to synthesize images belonging to a specific class and to generate an augmented image classification dataset using the Diffusion model. (Certainly, with some small modification, we can finetune these methodologies to make them fit in cases other than image classification) One such example is classifier-free guidance Diffusion.

The basic idea of classifier-free Guidance Diffusions (Ho & Salimans, 2022) comes from classifier-guided Diffusions (Dhariwal & Nichol, 2021). Assume the Diffusions score $\epsilon_\theta(z_\lambda, c) \approx -\sigma_\lambda \nabla_{z_\lambda} \log p(z_\lambda|c)$ where c is the class label. During sampling steps of classifier guided Diffusions (Dhariwal & Nichol, 2021), in each reverse forward process, classifier guided Diffusions (Dhariwal & Nichol, 2021) composes a new noise estimation in this manner:

$$\begin{aligned}\tilde{\epsilon}_\theta(z_\lambda, c) &= \epsilon_\theta(z_\lambda, c) - w\sigma_\lambda \nabla_{z_\lambda} p_\theta(c|z_\lambda) && (3)\\ &\approx -\sigma_\lambda \nabla_{z_\lambda}[\log p(z_\lambda|c) + w \log p_\theta(c|z_\lambda)] && (4)\end{aligned}$$

where $p_\theta(c|z_\lambda)$ is a pre-trained classifier that recognize z_λ into some class. If we consider $\tilde{\epsilon}_\theta(z_\lambda, c)$ as $\nabla_{z_\lambda} \log \tilde{p}(z_\lambda|c)$, then this noise estimation is equivalent to push sampled images in a direction that maximizes $\tilde{p}(z_\lambda|c)$ where

$$\tilde{p}(z_\lambda|c) \propto p(z_\lambda|c) p_\theta(c|z_\lambda)^w \tag{5}$$

Classifier free guidance utilizes this formula and combines it with Bayesian Theorem. By Bayesian Theorem we have $p_\theta(c|z_\lambda) \propto \frac{p(z_\lambda|c)}{p(z_\lambda)}$, if we insert it back to Equation 5, we can derive a new formula

$$\tilde{p}(z_\lambda|c) \propto \frac{p(z_\lambda|c)^{w+1}}{p(z_\lambda)^w} \tag{6}$$

Classifier free guidance Diffusions (Ho & Salimans, 2022) then suggested that we can train an conditional noise estimator $\epsilon_\theta(z_\lambda, c) \approx -\sigma_\lambda \nabla_{z_\lambda} \log p(z_\lambda|c)$ and an unconditional noise estimator $\epsilon_\theta(z_\lambda) \approx -\sigma_\lambda \nabla_{z_\lambda} \log p(z_\lambda)$. According to Equation 6, we can therefore derive a new noise estimator

$$\tilde{\epsilon}_\theta(z_\lambda, c) = (1 + w)\epsilon_\theta(z_\lambda, c) - w\epsilon_\theta(z_\lambda) \tag{7}$$

Classifier-free guidance Diffusions (Ho & Salimans, 2022) therefore proposed a training algorithm training both a conditional noise estimator $\epsilon_\theta(z_\lambda, c)$ and an unconditional noise estimator $\epsilon_\theta(z_\lambda, \emptyset)$. They also proposed a corresponding sampling algorithm generating images belonging to a specific class. We have recalled these two algorithms in this paper.

3.2. Class embedddings in denoising models

Even though training and sampling algorithms of classifier free guidance network have already been proposed, it is still under discussion what model architecture should be used to estimate noise. In DDPM paper (Ho et al., 2020), a modified version of UNet (Ronneberger et al., 2015) was proposed to estimate the noise unconditionally in the reverse process. This modified version of UNet replaces some of the original UNet's convolutional neural network (CNN) layers with Multi-head attention layers (Vaswani et al., 2017). We design a similar UNet architecture to estimate noise (both unconditionally and conditionally) in (Ho & Salimans, 2022). In this version, we add class embedding (we also encode class embedding of $\emptyset$ in order to fit this architecture to unconditional noise estimation) to each layer of the network. The high-level view of our network architecture is shown in Figure 1 (The channel number, image embedding's widths, and heights are noted in the figure). The design details (of the unconditional noise estimator) come as follows:

1. In each layer of the encoder (including the middle bottleneck), we sequentially apply CNN, multi-head Attention, and max pooling on image embeddings.
2. In each layer the decoder, we apply concatenation of image embeddings from the encoder (Figure 1 uses arrows connecting blocks in the encoder and the decoder to indicate which layer in the encoder will be concatenated to which layer in the decoder), upsampling, CNN, multi-head Attention on image embeddings.
3. We also apply skip connections between different sizes of image embeddings in the encoder and the decoder (not shown in Figure 1).

If we are going to include class embeddings (derived from class label using Class Embedding Networks (CEN)) in this network to make it class-conditional, after we conduct CNN and multi-head Attention on each layer, we concatenate a class embedding vector encoding different features into the image embeddings. (Purple blocks attached to each layer block in Figure 1).

But how large should class embedding in each layer be? We have proposed three different patterns:

1. Uniformly concatenate the same size of class embedding vector in each layer. We name this class embedding pattern as Uniform class embedding.
2. Make the size of class embedding vector equal to the number of channels of each layer. The closer each layer is to the bottleneck block, the larger class embeddings will be concatenated to that layer. We name this class embedding pattern as Pyramid class embedding.
3. Insert class embedding only to the bottleneck block. If we still retain a small number of class embeddings in other layers, we call it Bottleneck class embedding;

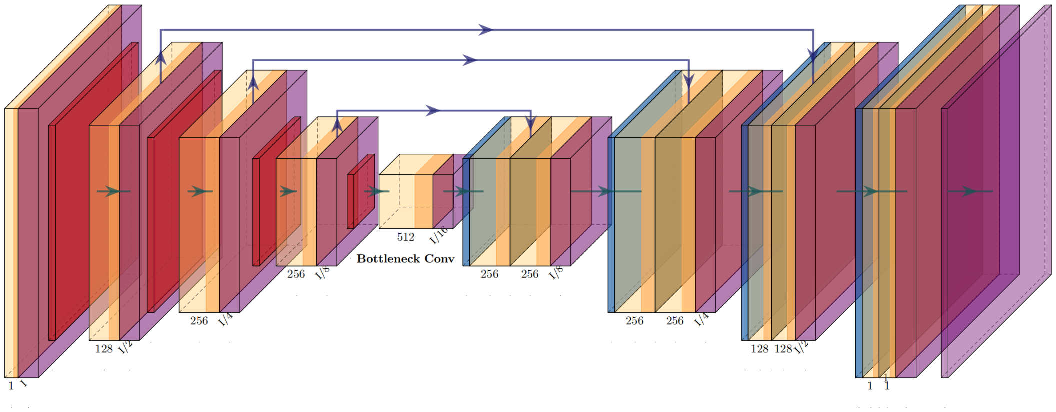

Figure 1. UNet model architecture used in Diffusions reverse process

if all class embeddings in layers other than bottleneck are eliminated, we call it Complete Bottleneck class embedding.

We compare these three types of class embedding patterns carefully in section 4. In summary, the final result shows that Uniform class embedding fails to improve generated images' quality as expected as we increase the number of features in each layer, which objects to our common sense that the more features considered, the better the model is. Bottleneck and Complete Bottleneck class embedding seem to achieve the best generated image quality while Pyramid class embedding achieves only slightly worse image quality but better diversity.

Class Embed Type	FID	IS	NLL
Uniform-1	3,253	3.575	3.084
Uniform-16	3.240	2.717	1.851
Uniform-256	3.240	4.671	0.392
Complete Bottleneck	3.223	7.416	**0.003**
Bottleneck	3.219	**7.711**	0.005
Pyramid	**3.217**	7.472	0.027

Table 1. FID, IS, NLL scores for Class Embedding types mentioned in section 3.2

4. Experiments

In this section, we display our experiments conducted on the MNIST dataset. Firstly we explain what metrics we are using to evaluate generated images. Next, we will use these metrics to compare the performances of different class embedding patterns.

4.1. Metrics

Three metrics we evaluate generated images by are Fréchet Inception Distance (FID) (Heusel et al., 2017), Inception Score (IS) (Salimans et al., 2016), and Negative Log Likelihood (NLL) of an image belonging to a specific class.

FID and IS consider both image diversity and image qualities. However, according to a series of research starting from (Brock et al., 2018), empirically, FID is more specialized in image diversity, and IS is more specialized in image quality. IS measures images' quality by running a classifier classifying the images $p(y|x)$ where x is the image and y is the image label and also measures images' diversity considering $p(y)$ across classification results of all generated images; in short, it measures

$$KL(p(y|x)||p(y))$$

FID measures images' quality by comparing the mean of generated images' and real images' feature vectors (μ_w, μ respectively) and measures images' diversity by comparing the variance of generated image's and real images' feature vectors (Σ_w, Σ); more specifically it measures

$$|\mu - \mu_w| + tr(\Sigma + \Sigma_w - 2(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w)^{\frac{1}{2}}))$$

In our experiment, we train a classifier to compute IS and uses the pre-trained Inception-V3 (Szegedy et al., 2016) to compute FID score.

Even though IS is famously used to evaluate generative models, it has certain drawbacks when used to evaluate a class-conditional generative model. Firstly, Inception score cannot be evaluated class by class. (Assume the class-conditional generative models can generate enough high-quality images) If we evaluate IS on one class of image, then $p(y)$ will be extremely unbalanced and IS will be unreasonably small. However, evaluation class by class is

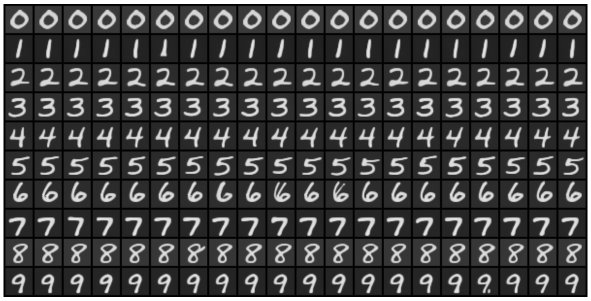

Figure 2. Generated Images using Bottleneck class embedding

an important ability for class-conditioned metrics to have, given the consideration that researchers might want to know exactly which class the model does terribly on.

Secondly, the model has methods to cheat on IS. Suppose the trained class conditional model is not class sensitive, namely, the images it generates on one class might actually belong to many different classes. In this case, if images it generates for every class together form a uniform distribution $p(y)$ and most generated images can be well classified, even though the model fails to be class conditional, it can still obtain high IS through cheating.

To overcome these two drawbacks, we propose NLL score. We use the same classifier we use in IS to compute NLL score. NLL score measures the probability of one image belonging to one specific class. Suppose the classifier implies the conditional distribution of the class label given the generated image $p(y|x)$. NLL score equals to

$$-\log p(y = c_{\text{correct}}|x)$$

So the smaller the average NLL score is, the better class conditional generative model is. NLL score can be evaluated class by class, and it is class sensitive. Therefore NLL can fix the drawbacks of IS in evaluating the class conditional generative model.

4.2. Varying class embedding and Fix everything else

We conduct our comparison experiment on the MNIST dataset. We only research on class embedding patterns and select the hyperparameters according to experiments already conducted by classifier-free guidance Diffusions (Ho & Salimans, 2022). Some Diffusions specific hyperparameters are selected as follows:

1. When classifier free guidance Diffusions (Ho & Salimans, 2022) was trained on ImageNet 128×128, IS score became the highest when $w = 4.0$ while FID became the highest when $w = 0.3$. In our experiment, we just fix $w = 4.0$
2. Classifier free guidance Diffusions (Ho & Salimans, 2022) performed equally well when unconditional model was trained with probability $p_{\text{uncond}} \in \{0.1, 0.2\}$. In our experiment, we just fix $p_{\text{uncond}} = 0.2$
3. When classifier free guidance Diffusions (Ho & Salimans, 2022) was trained on ImageNet 64×64, v was set to 0.2, and we also set v to this number as well
4. Sampling steps in our experiment are selected to be $T = 1000$, which is close to $T = 1024$, the best sampling steps selected by classifier-free guidance Diffusions (Ho & Salimans, 2022).
5. Variance Scheduler are cosine scheduler where $\alpha_t = \cos\left(\frac{t/T+s}{1+s}\frac{\pi}{2}\right)$ and s is set to be 0.0008
6. Time embeddings added to each layer of noise estimator is set to be size 100

Besides these Diffusions specific hyperparameters, other parameters are selected as follows:

1. The number of channels of each noise estimator layer follow Figure 1, the encoder channel numbers are 1, 128, 256, 256 respectively, so are the decoder channel numbers. The bottleneck block has a channel of size 1024.
2. All Multi-head Attentions (Vaswani et al., 2017) in noise estimator are set to have only one head.
3. Each model is trained only 25 epochs to prevent overfitting

All class embeddings are added in a symmetric manner; that is the number of class embeddings added to one noise estimator's encoder layer is the same as that added to the corresponding decoder layer. For simplicity, we use a length five (the first four entries represent the class embedding size for four encoder layers and the last entry represents the class embedding size for one bottleneck layer) array to indicate the class embedding size. Uniform-i in Table 1 can be described by vector $[i, i, i, i, i]$; Pyramid in our experiment can be described by vector $[1, 128, 256, 256, 512]$; Bottleneck and Complete Bottleneck in this experiment can be described by vector $[1, 1, 1, 1, 1024]$ and $[0, 0, 0, 0, 1024]$. In this way, Pyramid, Bottleneck and Complete Bottleneck encode a similar number of class features in all of their class embeddings.

The results of the experiments are shown in Table 1. As we can see, even though we encode more and more class features from Uniform-1 to Uniform-256, FID, IS, and NLL does not improve as expected and Uniform-16 even exhibits a decreased IS score. Pyramid, Bottleneck, and Complete Bottleneck achieve the best FID, IS, and NLL respectively.

We personally think Complete Bottleneck is worse than Bottleneck and Pyramid since neither its IS nor FID is better than those of Bottleneck and Pyramid and its low NLL fails to outperform Bottleneck greatly. Perhaps this indicates that we still want to preserve some class embeddings in layers other than a bottleneck. As for Bottleneck and Pyramid, it seems that Bottleneck is better in image quality while Pyramid is better in image diversity.

4.3. Generate images

In the paper (Ho & Salimans, 2022), the authors proposed a methodology requiring future work to investigate. They wished to use both conditional and unconditional Diffusions to denoise only in the very early stages and to switch back to unconditional only Diffusions in later stages. We use Bottleneck class embeddings, switch the number of heads of every multi-head attention (Vaswani et al., 2017) into 4, and train for only 12 epochs (to further prevent overfitting) on the MNIST dataset. We generate some samples with $w = 4.0$ and $v = 3.0$. The generated images are shown in Figure 2. Even though we do not measure FID, IS, and NLL on these images, these images have high visual quality and are class distinguishable. We, therefore, believe that the ideas proposed in future work part in (Ho & Salimans, 2022) should be able to work well.

5. Noisy Label Class-conditional Diffusions

In this section, we locate ourselves in the setting where some images have the wrong labels (noisy labels) but we do not know exactly which ones are like that [1]. Since we still want to perform the class-conditional diffusion method as mentioned in the previous sections, we wonder how we should detect the potential noisy label and mitigate their negative influence on the performance.

Our attempt is to borrow ideas from Curriculum Learning (CL) (Bengio et al., 2009). Generally speaking, CL describes a type of learning in which the model is first trained with only easy examples of a task and then gradually increase the task difficulty, which is in contrast to training the model on the entire dataset for every epoch. In particular, we adopt the method introduced by Zhou et al. (2021) to sample training batch according to the following score:

$$a_t(i) := \left\langle y_i - f(x_i;\theta_t), \frac{\partial f(x_i;\theta_t)}{\partial t}\Big|_{\mathcal{D}} \right\rangle$$

where for each training datapoint i in the training set $\mathcal{D}$ in t-th epoch, the first term represents the residual (difference between the model output and the target) and the second represents the linear dynamics computed on the whole dataset (how the model f changes over different epochs in terms of speed and direction).

We want to choose samples with a high score, since, on the one hand, such samples typically have high residuals indicating the model is not well-trained on them; on the other hand, the inner product also needs the residuals to match the linear dynamics (e.g. this hard sample is not adversarial to the whole learning process). However, although a data point with a noisy label has a large residual, it does not satisfy the second point mentioned above. Thus, we expect those noisy data to have a low score so that we do not sample them very often.

Our naive approach above fails to train a class-conditional diffusion model which can generate decent images. Note that when we apply the above score to the diffusion task, y_i becomes the noise ϵ_i added in the forward process and $f(x_i;\theta_t)$ now takes the noisy label and becomes $\tilde{\epsilon}_{\theta_t}(x_i, y_i)$. Although we expect the intuition mentioned before still makes sense in the way that $\tilde{\epsilon}_{\theta_t}(x, y_{noisy})$ does not match ϵ_i and the linear dynamics computed on the majority of clean data, training the model $\tilde{\epsilon}_\theta$ alternately with the class label or not makes the linear dynamics itself unstable. As a result, the inner product between the residual and the linear dynamics can no longer reflect the quality of a sample as we want. In the future, we plan to conduct experiments where the linear dynamics is only computed unconditionally so that we hope the above conjecture can be fixed.

6. Discussion

Our paper represents progress in proposing new architectures and making conditional diffusion models more specific and more powerful. Yet there still remain some improvements awaiting future work.

Firstly, we show that Pyramid embedding performs well in FID while Bottleneck does well in IS and NLL on the MNIST dataset. However, the differences in the experiment results are not very obvious and may be disturbed by experimental randomness to some extent. Due to the limitation of our computational power, the number of experiments conducted in this study may not be sufficient to draw definitive conclusions. In the future, we expect to conduct more experiments on larger datasets such as CIFAR-10 and ImageNet. We expect more research to further explore those classes of embedding patterns and clarify the strengths and weaknesses of each class of embedding patterns.

Secondly, only DDPM models (Ho et al., 2020) are considered during training and sampling, which rely largely on the Markov processes. Other state-of-art research such as (Song et al., 2020) proposes the use of non-Markov processes in order to increase the sampling speed of the images. We hope the embedding patterns we propose can be applied to

[1] Our code can be found through this link https://github.com/Konokiii/Noisy-Label-Diffusion

more types of diffusion models in the future.

Finally, we have assumed that the internal architectures of our Class Embedding Networks have sufficient representation power, and our comparative experiments on different embedding patterns are mostly focused on the networks' output sizes. Further research may be needed to evaluate the combined effect of internal architectures and output sizes.

7. Conclusion

We present high-quality image samples using diffusion models with CNN-Attention mixed U-Net Architecture. We start with the classifier-free guidance Diffusions model. On the basis of that research, we further experiment with different classes of class embedding methods, such as Pyramid embedding, Bottleneck embedding, and Complete Bottleneck embedding. We have shown that the Pyramid embedding performs well in terms of the Fréchet Inception Distance (FID) and therefore image diversity, while Bottleneck performs better in terms of the Inception Score (IS) and the Negative Log Likelihood (NLL) and therefore image quality. We encourage future work to confirm that our class embedding framework can be flexibly applied to other conditional diffusion models and other datasets as well.

References

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018. URL http://arxiv.org/abs/1809.11096.

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 2672–2680, 2014.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL http://arxiv.org/abs/1706.08500.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. URL https://arxiv.org/abs/2006.11239.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL http://arxiv.org/abs/1606.03498.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015. URL http://arxiv.org/abs/1503.03585.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020. URL https://arxiv.org/abs/2010.02502.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Zhou, T., Wang, S., and Bilmes, J. Curriculum learning by optimizing learning dynamics. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 433–441. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/zhou21a.html.