ZERO-SHOT IMAGE CLASSIFICATION WITH DIFFUSION MODELS

Valerii Startsev
Yandex Research
Moscow, Russia
sharfikeg@yandex-team.ru

Suraj Singh
Yandex Research
Moscow, Russia
jsingh@yandex-team.ru

ABSTRACT

Diffusion probabilistic models have recently demonstrated state-of-the-art generative performance in a wide range of applications, including inpainting, super-resolution, and semantic editing. While these models have proven highly effective in generative tasks, there is a growing interest in leveraging their generative approach for discriminative tasks, which has shown promising results. This paper focuses on the ability of large-scale text-to-image diffusion models to serve as a zero-shot classifier, which is a type of discriminative task that involves classifying images based on textual descriptions, without any prior training on the specific image classes. To evaluate the performance of these models, we compare them to the current state-of-the-art zero-shot classifier, CLIP. To the best of our knowledge, this is the first study to investigate the potential of large-scale text-to-image diffusion models for zero-shot classification. The code for the current work is publicly available at https://github.com/gracious-patience/zero_shot_diff_class.

1 INTRODUCTION

Since their first introduction, Denoising Diffusion Models (Sohl-Dickstein et al. (2015), Ho et al. (2020)) took a leading place in modeling natural data's distribution (Dhariwal & Nichol (2021)) and became a frontier of researchers' interest. That resulted in different applications of DDPMs in the variety of generative tasks. Moreover, with the recent success of the large-scale text-to-image models (Ho et al. (2021), Rombach et al. (2021), Saharia et al. (2022)) and a public release of Stable Diffusion (stableAI (2022)) comes a widespread boom in the image editing research (Gal et al. (2022), Voronov et al. (2023), Hertz et al. (2022), Ruiz et al. (2022), Zhang et al. (2022)), as well as non-scientific usage among the people all around the globe.

On the other hand, generative models appeared to efficiently solve discriminative computer vision problems. Not only Generative Adversarial Networks (GANs) and Autoregressive models are used in this scope, but also diffusion models found their applications in such problems as the semantic segmentation (Baranchuk et al. (2021)), object detection (Chen et al. (2022)).

The main goal of this research is to, having an evidence from (Baranchuk et al. (2021)), examine how well can contemporary diffusion models deal with a zero-shot image classification task, especially in comparison to contrastive language-image pretraning (Radford et al. (2021)).

2 RELATED WORK

In this section we will overview the existing approaches to generative modeling, paying attention to differences and similarities between them. We will also recall what lies underneath the first diffusion models and shed light on the implementation of the state-of-the-art ones. The paragraph will be ended with an overview of CLIP.

2.1 GENERATIVE MODELING

The main goal of generative models is to learn a probability distribution $\mathcal{X}$ defined over $\mathbb{R}^n$, where n is typically large. Having a number i.i.d samples from $\mathcal{X}$ we can obtain a generator

$$g : \mathbb{R}^q \to \mathbb{R}^n$$

that maps a tractable distribution $\mathcal{Z}$, called latent, to points in $\mathbb{R}^n$. Commonly, when we are speaking about the continuous data, $\mathcal{Z}$ is assumed as a standard normal in $\mathbf{R}^q$. The generator g is approximated with a deep neural network g_θ, where θ denotes network's learnable parameters.

When the generator is known we can generate new data from $\mathcal{X}$ by sampling $\mathbf{z} \sim \mathcal{Z}$ and computing $g(\mathbf{z})$.

It is important to note that latent space dimension q is generally much less than the n: many high-dimensional data sets that occur in the real world actually lie along low-dimensional manifolds inside that high-dimensional space.

Sometimes (what is actually needed in our case), generator can be used to compute the likelihood of a sample $\mathbf{x}$:

$$p_{\mathcal{X}}(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_g(\mathbf{x}|\mathbf{z}) p_{\mathcal{Z}}(\mathbf{z}) d\mathbf{z} \tag{1}$$

which is usually intractable due to a high-dimension of the integral.

2.2 NORMALISING FLOWS

The key idea **Normalising Flows** (Rezende & Mohamed (2015), Kingma & Dhariwal (2018)) is to model generator g_θ as a diffeomorphic and orientaion-preserving function. Therefore, the dimension q of the latent space is assumed to be equal to n, which is a significant restriction. However, under these assumptions we can approximate the likelihood of a given datapoint $\mathbf{x}$ using the change of variable formula:

$$p_{\mathcal{X}(\mathbf{x})} \approx p_\theta = p_{\mathcal{Z}}(g_\theta^{-1}(\mathbf{x}))\, det \nabla g_\theta^{-1}(\mathbf{x}) = (2\pi)^{-\frac{n}{2}}\, exp\left(-\frac{1}{2}||g_\theta^{-1}(\mathbf{x})||^2\right) det \nabla g_\theta^{-1}(\mathbf{x}) \tag{2}$$

and no integration is needed. For the efficient work of the network we must choose the architecture of g_θ so that we can compute g_θ^{-1}, its Jacobian and evaluate $g_\theta(\mathbf{z})$.

Training of generator is carried out by the maximizing the likelihood samples from $\mathcal{X}$ under p_θ or equivalently minimizing the negative log-likelihood

$$J_{ML}(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[-log\, p_\theta(\mathbf{x})] \approx \frac{1}{s}\sum_{i=1}^{s}\left(\frac{1}{2}||g_\theta^{-1}(\mathbf{x}^{(i)})||^2 - log\, det \nabla g_\theta^{-1}(\mathbf{x}^{(i)}) + \frac{n}{2} log(2\pi)\right) \tag{3}$$

where $x^{(i)}$ are i.i.d samples from $\mathcal{X}$ and s is a batch size. Recalling the Kullback-Leibler (KL) divergence between $p_{\mathcal{X}}$ and p_θ

$$KL(p_{\mathcal{X}}||p_\theta) = \int p_{\mathcal{X}}(\mathbf{x}) log \frac{p_{\mathcal{X}}(\mathbf{x})}{p_\theta(\mathbf{x})} dx = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[log \frac{p_{\mathcal{X}}(\mathbf{x})}{p_\theta(\mathbf{x})}\right]$$

we see that $KL(p_{\mathcal{X}}||p_\theta) = J_{ML}(\theta) + \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[-log\, p_{\mathcal{X}}(\mathbf{x})]$ and since the second term is independent of θ, minimizing $J_{ML}(\theta)$ is equivalent to minimize KL-divergence.

2.3 VARIATIONAL AUTOENCODERS

Variational Autoencoders (simply, **VAEs**) (Kingma & Welling (2013), Rezende et al. (2014)) are a type of models that use a latent space of a much smaller dimension than the data space, $q << n$. Since generator is not invertible, it is not possible to compute the negative log-likelihood loss directly.

Let's notice that, using Bayes' rule, the $p_\theta(\mathbf{x})$ can be re-written as follows

$$p_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_{\mathcal{Z}}(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \tag{4}$$

Looking back to the previous section , we would like to train our model using maximum likelihood approach. While computing $p_\theta(\mathbf{x}|\mathbf{z})$ is straightforward, the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is generally intractable.

In VAE we approximate posterior with a family of parametrized probability distributions that are tractable (usually Gaussian), so we can sample from it and calculate the needed probabilities. Parameters of this distribution are given by the second neural network $e_\psi(\mathbf{z}|\mathbf{x})$. This network takes $\mathbf{x}$ as an input and yields parameters (mean, covariance) of the posterior.

With the introduction of the approximation $e_\psi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$ we now have to accomplish two goals: maximize the approximate likelihood and minimize error in posterior's approximation. To derive the appropriate objective let's see that:

$$log\, p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim \mathbf{e}_\psi(\mathbf{z}|\mathbf{x})}\left[log\, p_\theta(\mathbf{x})\right] = \mathbb{E}_{\mathbf{z}\sim \mathbf{e}_\psi(\mathbf{z}|\mathbf{x})}\left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]$$

After multiplying and devising the expression inside the logarithm by the $e_\psi(\mathbf{z}|\mathbf{x})$ and applying the "mul to sum" property we obtain that

$$log\, p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim \mathbf{e}_\psi(\mathbf{z}|\mathbf{x})}\left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{e_\psi(\mathbf{z}|\mathbf{x})}\right] + \mathbb{E}_{\mathbf{z}\sim \mathbf{e}_\psi(\mathbf{z}|\mathbf{x})}\left[\frac{e_\psi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]$$

The second term here is the $KL(e_\psi(\mathbf{z}|\mathbf{x})\,||\,p_\theta(\mathbf{z}|\mathbf{x})) \geq 0$. Dropping it provides us with a lower bound on the likelihood (also called evidence) $p_\theta(\mathbf{x})$. The first term is thus called *evidence lower bound* (ELBO). Simultaneously minimizing the negative-ELBO with respect to θ and ψ brings us to the loss used to train VAE:

$$J_{ELBO}(\boldsymbol{\psi}, \boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[log\, p_\theta(\mathbf{x}, \mathbf{z}) - log\, e_\psi(\mathbf{z}|\mathbf{x})] \tag{5}$$

2.4 Generative Adversarial Networks

Generative adversarial networks (**GANs**) were popularized the seminal work of (Goodfellow et al. (2014)) that casts GAN training as a two-sample test problem. Here, the discriminator, $d_\phi : \mathbb{R}^n \to [0, 1]$, is trained to predict the probability that a given example was part of the training dataset. This leads to a binary classification problem for the discriminator. Here, it is important to recall that we only sample from $g_\theta(\mathcal{Z})$ but do not attempt to estimate the likelihood $p_\theta(\mathbf{x})$. Clearly, the training of the discriminator is coupled with the training of the generator whose goal is to provide samples that are indistinguishable from the true dataset.

$$J_{GAN}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[log(d_\phi(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}[log(1 - d_\phi(g_\theta(\mathbf{z})))] \tag{6}$$

The discriminator seeks to maximize this function (indicating low classification errors) while the generator seeks to minimize this function (corresponding to a confused discriminator). In other words, training the generator and discriminator using the loss function is equivalent to finding a Nash equilibrium (θ^*, ϕ^*) such that

$$\phi^* \in \arg\max_\phi J_{GAN}(\theta^*, \phi) \text{ and } \theta^* \in \arg\min_\theta J_{GAN}(\theta, \phi^*).$$

The critical hurdle during training is balancing between the two subproblems. Consider the beginning of training, when it is relatively easy to distinguish the actual and generated samples. On the one hand, training the discriminator to optimality would make it impossible for the generator to improve, since the gradient $\nabla_\theta J_{GAN}(\theta, \phi^*)$ would be close to zero. On the other hand, not training the discriminator well enough would make it challenging to update the weights of the generator.

Another problem that also presents challenges to the GAN formulation is known as mode collapse. Consider the extreme case when the generator maps the entire distribution $\mathcal{Z}$ to a single data point $\mathbf{x}(1) \sim \mathcal{X}$, that is,

$$g_\theta(\mathbf{z}) = \mathbf{x}^{(1)} \text{ for almost all } \mathbf{z} \sim \mathcal{Z}.$$

In this case, the optimal discriminator would yield $d_{\phi^*}(\mathbf{x}^{(1)}) = \frac{1}{2}$ and $d_{\phi^*}(\mathbf{x}^{(j)}) = 1$ for all $j > 1$ and the training would terminate.

2.5 DIFFUSION MODELS

Contrary to VAEs, Denoising Diffusion Probabilistic Models (**DDPMs**) (Sohl-Dickstein et al. (2015), Ho et al. (2020)) have highly dimensional latent space (the same, as the image itself). The main idea comes from the non-equilibrium thermodynamics: the image is being noised for several time steps until it becomes a white noise and then a neural network is trained to reverse the process. Firstly, we will explain the most general mechanism that underlies the DDPM.

Given an image x_0 from the real data distribution $x_0 \sim q(x)$ we can define a diffusion process as a Markov chain, that gradually adds small amounts of Gaussian noise to the image and produces a sequence of noisy samples $x_1, ..., x_T$. Each transition kernel $q(x_t|x_{t-1})$ is defined by the variance β_t from the variance schedule $\{\beta_t\}_{t=1}^T$:

$$q(x_t|x_{t-1}) \sim \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$$

with joint transition

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

As $T \to \infty,\ x_T$ is equivalent to the isotropic Gaussian.

Using properties of Gaussian distribution, we can write the result of noising process at arbitrary step t in a closed form. Let $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_{t-1} = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\overline{\varepsilon}_{t-2} = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon, \quad (7)$$

where $\varepsilon_{t-1}, \varepsilon_{t-2}, ... \sim \mathcal{N}(0, I)$; $\overline{\varepsilon}_s$ merges two Gaussians. So, we can write

$$q(x_t|x_0) \sim \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, \sqrt{1-\overline{\alpha}_t}\mathbf{I}).$$

Now, let's get down to the generation process. If we can reverse the initial noising process and sample from $q(x_{t-1}|x_t)$, we will be able to recreate true sample from the Gaussian noise input $x_T \sim \mathcal{N}(0, I)$. We cannot easily estimate $q(x_{t-1}|x_t)$, therefore we need to learn a model p_θ to approximate these conditional probabilities in order to run the reverse diffusion process.

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t),$$

where $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_t; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$.

The reverse conditional probability is tractable when conditioned on x_0:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I).$$

Using Bayes' rule, we have:

$$q(x_{t-1}|x_t, x_0) = q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} =$$

$$= exp(-\frac{1}{2}((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\overline{\alpha}_{t-1}}x_{t-1}^2) - (\frac{2\sqrt{\alpha_t}}{\beta_t} + \frac{2\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_{t-1}}x_0)x_{t-1} + C(x_t, x_0))).$$

Thus, the mean and variance can be parameterised as follows:

$$\tilde{\beta}_t = 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\overline{\alpha}_{t-1}}) = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\,\beta_t$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_{t-1}}x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}x_0.$$

Using (7), we can represent $x_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1-\overline{\alpha}_t}\varepsilon_t)$ and plug into the above equation to obtain:

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon_t) \tag{8}$$

To train our model we can, similarly to VAEs, use variational lower bound and optimize the negative log-likelihood:

$$-logp_\theta(x_0) \leq -logp_\theta(x_0) + D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)) =$$

$$= -logp_\theta(x_0) + \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}[log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T}/p_\theta(x_0))}] = \mathbb{E}_q[log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}].$$

Let $L_{VLB} = \mathbb{E}_q[log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}] \geq -\mathbb{E}_{q(x_0)}logp_\theta(x_0)$

We can rewrite L_{VLB} as a sum of separate parts:

$$L_{VLB} = L_T + L_{T-1} + ... + L_0,$$

where

$$L_T = D_{KL}(q(x_T|x_0)||p_\theta(x_T))$$

$$L_t = D_{KL}(q(x_t|x_{t+1}, x_0)||p_\theta(x_t|x_{t+1})) \text{ for } 1 \leq t \leq T-1.$$

As we need to approximate conditional probabilities $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ of the reverse process, we would like to train μ_θ to predict $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon_t)$. x_t is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict ε_t from the x_t at time step t:

$$x_{t-1} = \mathcal{N}(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}((x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon_\theta(x_t, t)), \Sigma_\theta(x_t, t)).$$

If we represent loss term L_t as KL-divergence between two Gaussians, then it becomes parameterized to minimize difference in $\tilde{\mu}$:

$$L_t = \mathbb{E}_{x_0,\varepsilon}[\frac{1}{2||\Sigma_\theta(x_t, t)||_2^2}||\tilde{\mu}_t(x_t, t) - \mu_\theta(x_t, t)||^2] =$$

$$= \mathbb{E}_{x_0,\varepsilon}[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\overline{\alpha}_t)||\Sigma_\theta||_2^2}||\varepsilon_t - \varepsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon_t, t)||^2].$$

Empirically, (Ho, citation) found that training the diffusion model works better with a simplified objective that ignores weighting term:

$$L_t^{simple} = \mathbb{E}_{t\sim[1,T],x_0,\varepsilon_t}[||\varepsilon_t - \varepsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon_t, t)||^2].$$

Note, that this objective has a nice and clear interpretation: given image x_0 from the initial dataset and a time step t, network is trained to predict noise ε that was used to make a noisy sample x_t.

2.6 LATENT DIFFUSION

Latent diffusion model (Rombach et al. (2021)) runs the diffusion process in the latent space instead of pixel space, making training cost lower and inference speed faster. It is motivated by the observation that most bits of an image contribute to perceptual details and the semantic and conceptual composition still remains after aggressive compression. LDM loosely decomposes the perceptual compression and semantic compression with generative modeling learning by first trimming off pixel-level redundancy with autoencoder and then manipulate/generate semantic concepts with diffusion process on learned latent.

The diffusion and denoising processes happen on the latent vector $\mathbf{z}$. The denoising model is a time-conditioned U-Net, augmented with the cross-attention mechanism to handle flexible conditioning information for image generation (e.g. class labels, semantic maps, blurred variants of an image). The design is equivalent to fuse representation of different modality into the model with cross-attention mechanism. Each type of conditioning information is paired with a domain-specific encoder τ_θ to project the conditioning input y to an intermediate representation that can be mapped into cross-attention component, $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V},$$

where $\mathbf{Q} = \mathbf{W}_Q^{(i)} \cdot \varphi_i(\mathbf{z}_i), \quad \mathbf{K} = \mathbf{W}_K^{(i)} \cdot \tau_\theta(y), \quad \mathbf{V} = \mathbf{W}_V^{(i)} \cdot \tau_\theta(y)$, and $\mathbf{W}_Q^{(i)} \in \mathbb{R}^{d \times d_\varepsilon^i}$, $\mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d \times d_\tau}$, $\varphi_i(\mathbf{z}_i) \in \mathbb{R}^{N \times d_\varepsilon^i}$, $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$.

2.7 CLIP MODEL

While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP (Radford et al. (2021)) jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Given a batch of N (image, text) pairs, CLIP is trained to predict which of the $N \times N$ possible (image, text) pairings across a batch actually occurred. To do this, CLIP learns a multi-modal embedding space by jointly training an image encoder and text encoder to maximize the cosine similarity of the image and text embeddings of the N real pairs in the batch while minimizing the cosine similarity of the embeddings of the $N^2 - N$ incorrect pairings. A symmetric cross entropy loss is optimized over these similarity scores.

The zero-shot classification with CLIP is then really straight forward: given a picture and a set of classes, one should get CLIP embeddings of the picture

$$\varphi_{\mathcal{I}}(\mathbf{x})$$

and of the prompts of the form "a photo of a **class**"

$$\varphi_{\mathcal{T}}(\tau_1), ..., \varphi_{\mathcal{T}}(\tau_n),$$

then calculate n scalar products

$$s_i = (\varphi_{\mathcal{I}}(\mathbf{x}), \varphi_{\mathcal{T}}(\tau_i)), \quad i \in 1, ..., n,$$

and choose the greatest one:

$$\underset{i \in 1,...,n}{\arg\max}\, s_i.$$

3 METHOD

We are now prepared to delve into the proposed methods of using a diffusion model as a potential alternative to CLIP for the task of zero-shot classification. However, it's important to note that at the outset of our research, we conducted a series of preliminary experiments on the class conditional diffusion model, which was trained on CIFAR-10 (Krizhevsky et al. (2009)), in order to verify its classification abilities (not in a zero-shot setup). These preliminary methods will be referred to as "toy methods," and included:

1. Exact Likelihood Computation in the image space
2. Simple metric-based methods in the latent space of DDIM (Song et al. (2020))

Following this, we developed several methods for real zero-shot classification, which we will list below before providing a detailed description of each one.

1. Noise-denoise and equivalent image-to-image pipelines
2. Cross-attention maps
3. Textual inversion

3.1 Exact Likelihood Computation

Denoting set with class labels as $\mathcal{C}$ and a given image as x, one can formalize a classification task in the following way:

$$c^* = \arg\max_{c \in \mathcal{C}} p(c|x). \tag{9}$$

Therefore, if $p(x|c)$ is known, one can use Bayes' rule to derive the desired probability:

$$p(c|x) = p(x|c)\frac{p(c)}{p(x)}.$$

As long as $p(x)$ does not influence $\arg\max_{c \in \mathcal{C}}$ and $p(c)$ are assumed to be equal to $\frac{1}{|\mathcal{C}|}$, it is sufficient to calculate $p(x|c)$. Here comes diffusion model. With the help of DDIM and Hutchinson's trick it is possible to get exact log-likelihood $logp(x|c)$ (Grathwohl et al. (2018)). Due to the monotonicity of the logarithm, to solve the initial argmax problem, one has to compute $logp(x|c)$ for each $c \in \mathcal{C}$ and choose the log-likelihood with the largest value. A high-level description of the algorithm can be found here (1).

Algorithm 1 Exact Likelihood

Require: image x,set with class labels $\mathcal{C}$, DDIM encoder $\mathcal{E}()$, Hutchinson's transform $\mathcal{H}()$
Ensure: $c^* = \arg\max_{c \in \mathcal{C}} p(c|x)$
  $c^* \leftarrow 0$
  $ll \leftarrow []$
  for all c $\in \mathcal{C}$ **do**
    $z \leftarrow \mathcal{E}(x, c)$
    ll.append $\mathcal{H}(z, c)$
  end for
  $c^* \leftarrow max(ll)$
    return c^*

3.2 Simple metric-based methods in the latent space of DDIM

Because DDIM enables us to have a one-to-one correspondance between pixel and latent spaces and intuition that originates from here: Figure 3, several methods based on proximity properties of latent vectors can be developed. However, these algorithms are of phenomenological nature. To read more about the intuition behind it, proceed to the section **4.1**.

For example, a kNN-based algorithm (2) to find the right latent, assuming its "out of distribution" behaviour.

3.3 Noise-denoise approach

Very intuitive algorithm that bases on the ability of the diffusion model to reconstruct images from the given corruption state. Crucial idea here is that the starting point of the reverse diffusion process (that is, our noised image) conditions the output of the network and that this process is also conditioned on the text prompt (in our case, the capture of the class, such as "a photo of a dog").

Algorithm 2 kNN

Require: image x,set with class labels $\mathcal{C}$, DDIM encoder $\mathcal{E}()$, L2-kNN function $kNN()$, number of neighbours to check k

Ensure: $c^* = \arg\max_{c \in \mathcal{C}} p(c|x)$

```
c* ← 0
n = ∞
for all c ∈ C do
    z ← E(x, c)
    if kNN(z, k) < n then
        n ← kNN(z, k)
        c* ← c
    end if
end for
    return c*
```

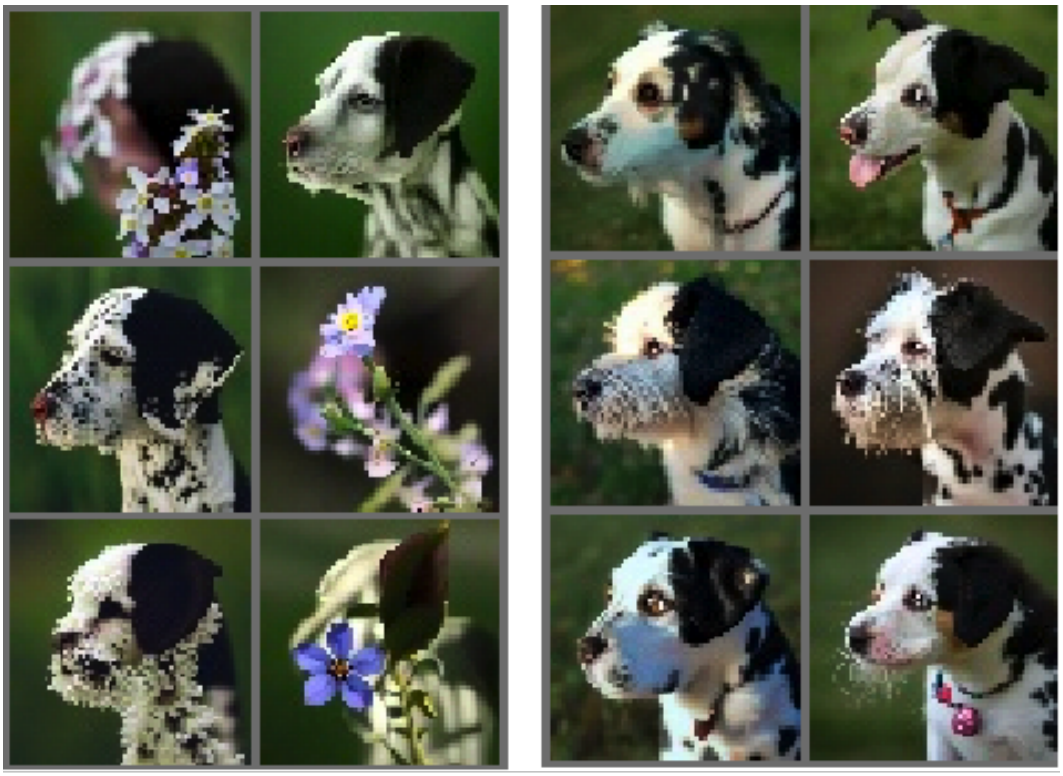


Figure 1: A block with first 6 images shows 64x64 Imagen reconstructions of a dog photo with wrong "flower" caption. The second block contains reconstructions with right "dog" label.

Therefore, algorithm makes decision founded on the "similarity" of the original image and the reconstructions. With the right amount of noise one can reach the situation when the reconstruction generated with the wrong prompt is changed accordingly (resulting either in a complete different image or a picture of surreal creature(1)), while the one . with the true prompt stays almost unchanged. This "similarity" or "dissimilarity" was measured via pixel-wise L2-difference and L2-difference between CLIP embeddings of the original picture and its reconstructions.

This approach can be split into two schemes: single noise level (3) and multiple noise level with voting (4). Both include quite a number of hyperparameters to be set: a noise level in the first case

Algorithm 3 Single noise-denoise scheme

Require: image x,set with class labels $\mathcal{C}$, noise level $l \in [0, 1]$, diffusion model $\mathcal{D}()$, distance function $\mathcal{F}()$
Ensure: $c^* = \arg\max_{c \in \mathcal{C}} p(c|x)$
  $c^* \leftarrow 0$
  $dists \leftarrow []$
  for all c $\in \mathcal{C}$ **do**
    $z \leftarrow \mathcal{D}.noise(x, l)$
    $\tilde{x} \leftarrow \mathcal{D}.denoise(z, c, l)$
    $dists$.append($\mathcal{F}(x, \tilde{x})$)
  end for
  $c^* \leftarrow min(dists)$
    return c^*

and a sequence of noise levels in the second, a number of inference steps. Note, that these pipelines

Algorithm 4 Multi noise-denoise scheme

Require: image x,set with class labels $\mathcal{C}$, noise levels ls, diffusion model $\mathcal{D}()$, distance function $\mathcal{F}()$
Ensure: $c^* = \arg\max_{c \in \mathcal{C}} p(c|x)$
  $c^* \leftarrow 0$
  $dists \leftarrow [][]$
  for $l \in ls$ **do**
    for c $\in \mathcal{C}$ **do**
      $z \leftarrow \mathcal{D}.noise(x, l)$
      $\tilde{x} \leftarrow \mathcal{D}.denoise(z, c, l)$
      $dists[l][c] \leftarrow (\mathcal{F}(x, \tilde{x}))$
    end for
    $dists[l] \leftarrow argmin(dists[l])$
  end for
  $c^* \leftarrow argmin(dists.distinct)$
    return c^*

are applicable to both cascade and latent diffusion implementations.

3.4 Cross-attention maps

Having an evidence from (Hertz et al. (2022)), we propose a method which deals with binary masks of the deep UNet layers. Namely, cross-attention blocks from the backbone of text-to-image diffusion models are used to fuse together two modalities: text and image, assigning each textual token a weighted pixel matrix of downsampled image. This shows us which token refers to which part of a picture. Method (5) stacks this masks for every class-defining token from a big prompt of a type "a photo of a class1 or a photo of a class2 ..." and for a given time step t and calculate one of the following aggregations: mean along the mask or the number of mask pixels with intensities greater than a given threshold.

3.5 Textual inversion

Papers (Gal et al. (2022),Ruiz et al. (2022)) suggest learning (frozen model, UNet loss) a brand new token to represent an object on the given images (usually, 3-4 pictures). Here we also propose 2 slightly different variants of possible classifying algorithms. The first (6) is just to take the original idea from (citation) unchanged, feed the given picture (or, may be, a bunch of its augmentations, as the initial textual inversion requires a number of images with the object one need to depict) into it and get the learned token τ. Then CLIP embeddings are calculated, one for the obtained new token and $|\mathcal{C}|$ others for the considered classes. The output is then chosen as the $argmin_{c \in \mathcal{C}} ||\psi(\tau) - \psi(c)||_2$, where $\psi()$ is the operation of CLIP embedding.

Algorithm 5 Cross-attention maps aggregation

Require: image x, prompt with all classes $\mathcal{P}$, time step $t \in [1, 1000]$, UNet $\mathcal{U}()$, aggregation $\mathcal{A}()$
Ensure: $c^* = \arg\max_{c \in \mathcal{C}} p(c|x)$
  $c^* \leftarrow 0$
  $values \leftarrow []$
  for c $\in \mathcal{C}$ **do**
    $m \leftarrow \mathcal{U}(x, t, \mathcal{P})$
    $values.append(\mathcal{A}(m))$
  end for
  $c^* \leftarrow argmax(values)$
    return c^*

Algorithm 6 Textual inversion

Require: image x, set of all classes $\mathcal{C}$, inversion algorithm $\mathcal{I}()$, number of epochs T, augmentations $\mathcal{A}()$, CLIP embedder $\psi()$
Ensure: $c^* = \arg\max_{c \in \mathcal{C}} p(c|x)$
  $c^* \leftarrow 0$
  $values \leftarrow []$
  $\tau \leftarrow \mathcal{I}(\mathcal{A}(x), T)$
  for c $\in \mathcal{C}$ **do**
    $values.append(||\psi(\tau) - \psi(c)||_2)$
  end for
  $c^* \leftarrow argmin(values)$
    return c^*

This method, however, can lead to overfitting and learning the token that stands alone. To address such a problem, the following scheme 7 can be applied. One can take a weighted sum of class tokens as the textual prompt, calculate the gradient of the diffusion loss with respect to the vector of weights and update it through backpropagation.

Algorithm 7 Weighted optimization

Require: image x, set of all classes $\mathcal{C}$, inversion algorithm $\mathcal{I}()$, number of epochs T, augmentations $\mathcal{A}()$, CLIP embedder $\psi()$
Ensure: $c^* = \arg\max_{c \in \mathcal{C}} p(c|x)$
  $c^* \leftarrow 0$
  $v \leftarrow rand(|\mathcal{C}|)$
  $\tau \leftarrow \mathcal{I}(\mathcal{A}(x), T)$
  repeat
    $values.append(||\psi(\tau) - \psi(c)||_2)$
  until converged
  $c^* \leftarrow argmin(values)$
    return c^*

4 EXPERIMENTS

Throughout the research we conducted several experiments which we will describe here.

4.1 THE SIMPLIEST CIFAR-10 CLASSIFICATION

As a kind of sanity check, it was proposed to try and exploit conditional DDPM trained on CIFAR-10 (Krizhevsky et al. (2009)) dataset. Following the Bayesian classification formulation and using the probabilty flow ODE 10 exact log-likelihoods were computed for several images, encoded with 10 different labels to choose

$$\arg\max_{y \in \{0,..,9\}} p(y|x)$$

Unfortunately, for this setup all the likelihoods appeared to be almost the same.

Nevertheless, my experiments showed that the difference in latents of differently encoded images is indeed exists. Namely speaking: latent codes, obtained by encoding the original image with wrong labels, resulted in less diverse and plausible pictures when decoded (Figure 2):

Figure 2: When image is encoded with the wrong label, final examples contain information about the original one.

Another interpretation of the effect is that wrongly labeled reconstructions try to preserve high-level information of the given image (Figure 3). It can be seen that the most of lines (except the "car"(false) and "truck"(true) lines) lack diversity in color and contain reconstructions that can be ambiguously classified. We can express this as: **Inception Score of false-encoded→decoded reconstructions is less than the one of true-encoded→decoded pictures**. From this example we also can conclude, that the model confuses semantically close classes.

Figure 3: The row that corresponds to the true label encoding is the most "redless". Interestingly to note: due to semantically closeness of "car" and "truck" object the corresponding lines are quite similar, although the last one is more authentic

Nevertheless, some results are not as good as the shown here. For example, in the Figure 7 all lines are almost the same regardless of the label used to encode the image. For other image examples please refer to the Appendix section.

The main task here is to understand observed effect from the point of latent codes.

The almost same experiment was to interpolate latents in the following setups:

1. True decoding with every variant of encoding
2. False decoding with every variant of encoding

We used linear spherical interpolation which was also used in (Song et al. (2020)). Results are shown in Figure 4 are quite the same as for the original experiment.

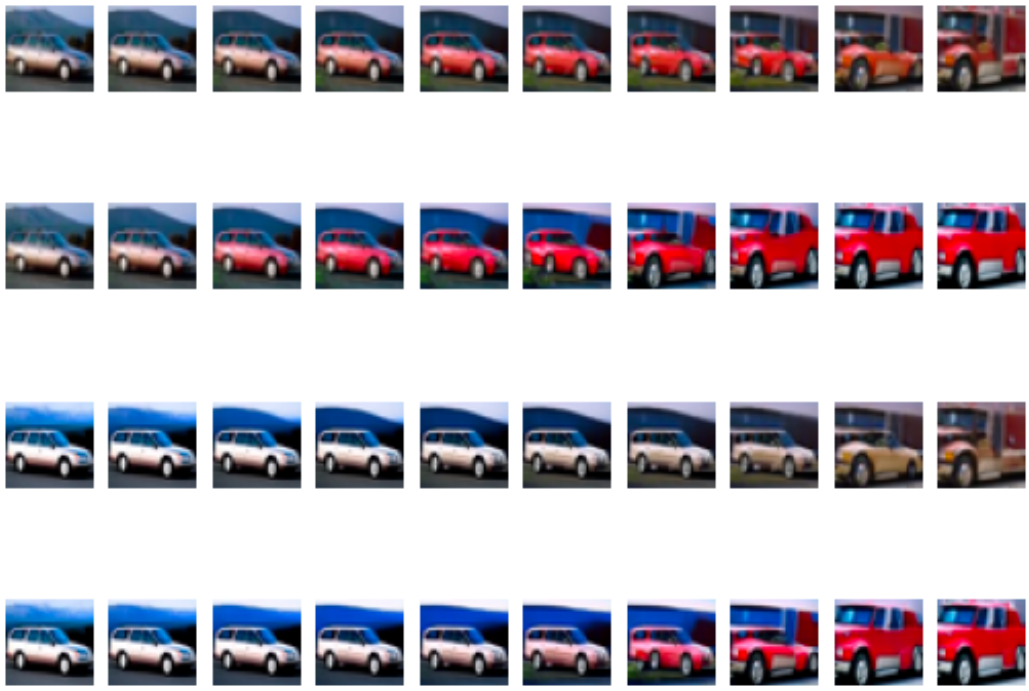


Figure 4: Images were encoded in the following pattern: true-true, true-false, false-true, false-false

4.2 NOISE-DENOISE CLASSIFICATION

After experiencing limited success with simple metric-based methods, we recognized the importance of conditioning the denoising process and turned to the noise-denoise method. We initially tested this method on our class conditional model and observed significant improvement, achieving a maximum accuracy of 0.92 compared to the best result of 0.81 with the previous method. However, we used the mean squared error (MSE) metric to measure the distance between the original image and its reconstructions, which led to poor performance of the Imagen64-based noise-denoise approach on CIFAR-10.

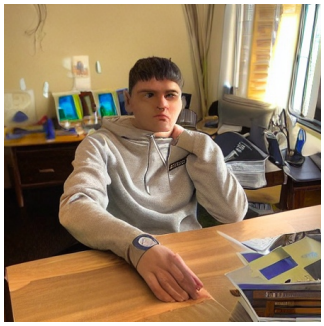
To address this issue, we decided to use a better metric that was more closely related to the semantic differences in images, rather than pixel-wise. Specifically, we measured the L2-distance between the CLIP embeddings of the original image and its reconstructions. Unfortunately, our method was only 1 percent better than CLIP on the easy dataset (Chaladze (2017)) and significantly worse on more class-rich datasets, and the classification process was very slow.

In an attempt to reduce the influence of the fixed 64×64 resolution, we introduced flexible latent diffusion as a backbone instead of Imagen. However, this did not improve the results, and we only achieved a maximum accuracy of 0.7661 on Flowers10 (Deepnets (2022)), compared to CLIP ViT-L/14's accuracy of 0.9189.

The interesting phenomenon of disenslembling was noted by our studies. As an example taken from the real-world setup, a CLIP misclassifies a young man originally wearing glasses with the one that does not.

(a) Real Daniel.

(b) Prompt: "A photo of a young man with glasses."



(c) Prompt: "A photo of a young man without glasses."

Figure 5: Classification of Daniel whether he is wearing glasses or not.

4.3 Cross-attention classification

After the noise-denoise method proved to be relatively unsuccessful, we sought to address two primary issues:

1. Our use of CLIP as a metric may have limited our method's ability to outperform CLIP. Therefore, we aimed to identify a new functional metric to use in our approach.
2. We sought to find ways to potentially boost the classification process.

Furthermore, our research was inspired by a paper by Baranchuk et al. (2021) that utilized the inner workings of the UNet backbone of the diffusion model. The paper introduced the idea of cross-attention maps, which showed promise in passing the image through the UNet conditioned on a time step and a text-prompt containing all the classes of interest, and calculating simple aggregations on the cross-attention maps. Despite our model's success in differentiating bears from wolves, it struggled with distinguishing between wolves and foxes.

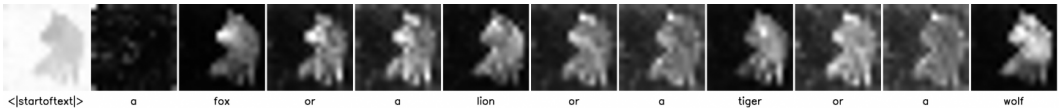


Figure 6: The token "wolf" has a higher aggregation value than the ground-truth "fox". Prompt: "a fox or a lion or a tiger or a wolf".

5 Discussion

5.1 Simple CIFAR-10 classification

The most trivial idea was to analyze raw latent codes from a metric point of point. Basing on the experimentally derived information, We assumed the true-encoded latent to be a kind of an outlier with respect to the others with them being grouped. However, implementation of a kNN-like classifier resulted in 70-80% accuracy, which is quite bad. The methods works well for examples illustrated in Figure 8 and Figure 3. Other ones have more complicated structure. In this scope, studying the robustness of DDPM to images' transformations. We may find a class of them, which cause a destructive effect on false-DDPMs.

5.2 Conclusion

This paper has presented several methods for zero-shot classification utilizing a large-scale text-to-image diffusion model as a backbone. The goal was to investigate the ability of such a model to classify, to generalize to unseen data, and to compare its performance with CLIP. However, the experiments have revealed that contemporary diffusion models are not yet capable of capturing enough information for successful classification in a zero-shot setup, despite the intuition that could be derived from Baranchuk et al. (2021).

While the results of this study may seem discouraging, they provide important insights into the limitations of current diffusion models and highlight the need for further research to enhance their performance. In particular, there are several potential areas of future work that could build upon the findings of this paper:

- Firstly, it would be interesting to explore "inversion"-type algorithms more thoroughly to investigate their ability to improve the performance of diffusion models for zero-shot classification.
- Secondly, the phenomenon of disensembling with CLIP could be studied more carefully, and a method for enhancing CLIP in a similar way to Ge et al. (2022) could be proposed.
- One can investigate the robustness of diffusion based zero-shot classifiers to the distortions of the input. This can be heavily utilized in autonomous driving systems, when unseen objects may occur in various circumstances (morning/evening, sunny/rainy/snowy weather)

By addressing these and other possible avenues for research, it may be possible to overcome the current limitations of large-scale text-to-image diffusion models for zero-shot classification and advance the field of generative models towards more robust and powerful solutions.

Acknowledgements

We would like to thank Valentin Khrulkov for sharing his Imagen, class-conditional diffusion model and exact likelihood computation implementations for the sake of this project and Dmitriy Baranchuk for sharing his vision and giving us advice.

References

Dmitry Baranchuk, Ivan Rubachev, Andrey Voynov, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models, 2021. URL https://arxiv.org/abs/2112.03126.

G. Kalatozishvili L. Chaladze. Linnaeus 5 dataset for machine learning, 2017. URL http://chaladze.com/l5/.

Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection, 2022. URL https://arxiv.org/abs/2211.09788.

Deepnets. Flower classification — 10 classes —, 2022. URL https://www.kaggle.com/datasets/utkarshsaxenadn/flower-classification-5-classes-roselilyetc.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022. URL https://arxiv.org/abs/2208.01618.

Yunhao Ge, Jie Ren, Yuxiao Wang, Andrew Gallagher, Ming-Hsuan Yang, Laurent Itti, Hartwig Adam, Balaji Lakshminarayanan, and Jiaping Zhao. Improving zero-shot generalization and robustness of multi-modal models, 2022. URL https://arxiv.org/abs/2212.01758.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models, 2018. URL https://arxiv.org/abs/1810.01367.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation, 2021. URL https://arxiv.org/abs/2106.15282.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions, 2018. URL https://arxiv.org/abs/1807.03039.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research), 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2015. URL https://arxiv.org/abs/1505.05770.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014. URL https://arxiv.org/abs/1401.4082.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021. URL https://arxiv.org/abs/2112.10752.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. URL https://arxiv.org/abs/2205.11487.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2020. URL https://arxiv.org/abs/2010.02502.

stableAI. Stable diffusion public release, 2022. URL https://stability.ai/blog/stable-diffusion-public-release.

Anton Voronov, Mikhail Khoroshikh, Artem Babenko, and Max Ryabinin. Is this loss informative? speeding up textual inversion with deterministic objective evaluation, 2023. URL https://arxiv.org/abs/2302.04841.

Zhixing Zhang, Ligong Han, Arnab Ghosh, Dimitris Metaxas, and Jian Ren. Sine: Single image editing with text-to-image diffusion models, 2022. URL https://arxiv.org/abs/2212.04489.

A SIMPLE CIFAR-10 CLASSIFICATION

The original image is located on the diagonal (in this case encoding-decoding process is just $g(g^{-1}(x)) = x$). We set num_steps to 200 and ode_steps to 6 in ODE solver.

Figure 7: All the rows are almost the same.

B NOISE-DENOISE CLASSIFICATION

In our experiments with stable diffusion we used weights and implementation from Hugging Face, namely runwayml/stable-diffusion-v1-5.

C CROSS-ATTENTION CLASSIFICATION

In our experiments with cross-attention maps we used notebook from Hertz et al. (2022) and stable diffusion backbone from Hugging Face, namely runwayml/stable-diffusion-v1-5.

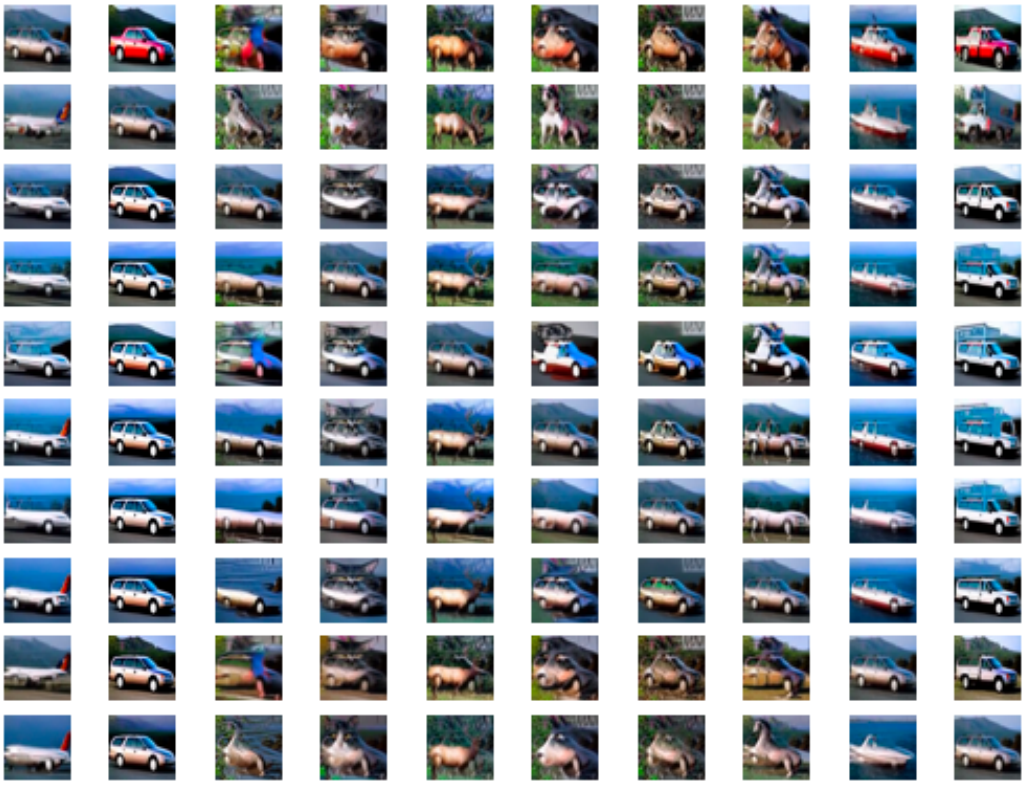


Figure 8: A pretty good example when a human can distinguish the class.

Figure 9: An example when it is difficult to make a correct decision about the class.

Figure 10: Almost collapsed with a neck-line pattern.

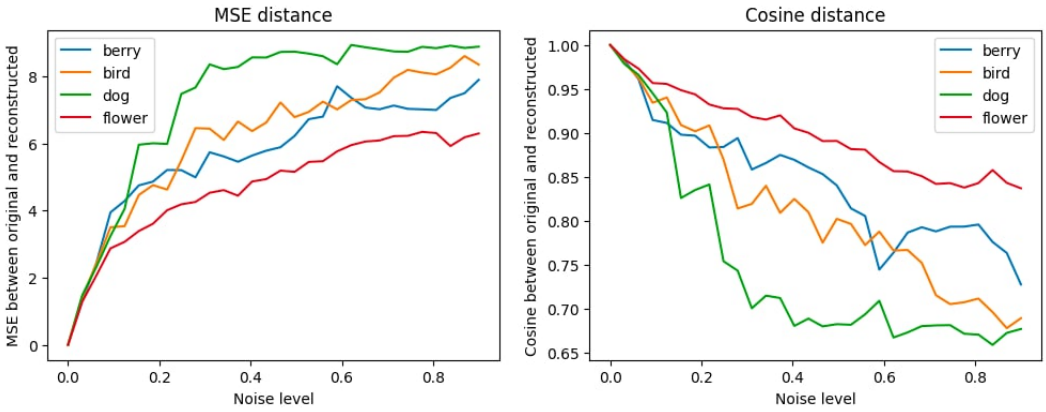


Figure 11: Example of how distance between the original image and its reconstructions evolves with the noise level in forward process in the noise-denoise method. Image from Chaladze (2017).