Reconstructing Natural Image from Brain Activity

Andrea Bosia
520600843
abos6574@uni.sydney.edu.au

Abstract

The following paper presents my research work in the scope of a Summer Vacation Research Internship for the University of Sydney. My work focus on the challenging task of reconstructing Visual Stimuli from brain signal captured via fMRI data.

1 Paper Structure

This report is structured as follow. Firstly in 2 I define what is the specific task of interest and which are the related challenges. Then follows a vast section 3 in which I present an extensive literature review covering the various methodologies adopted from the earliest experimentation up to the current state of the art. Next comes a section 4 dedicated to introducing what is, as of now days, the most vast dataset that bridges cognitive neuroscience and artificial intelligence. After follows a section 5 that goes into the theoretical details of diffusion models and the various conditioning techniques. Lastly in section 6 I introduce my implementation of a stable diffusion pipeline.

2 Task Identification

The end goal is to produce a model capable of reconstructing an image stimuli from brain signals. The reconstructed image should resemble as closely as possible the original image retaining the most semantic and structural information. Specifically the technique adopted to capture the brain activity is fMRI scan which at the current state represent the least invasive and most informative technology available. In terms of learning an fMRI representation, an effective algorithm should guarantee robustness across different subjects. Hence the same model should be able to deal with individual variations specific to a given subject. Moreover, beside successfully learning to reconstruct the visual stimuli from the brain activity an important result consists in understanding which specific areas of the brain can be mapped to which (extracted) features. A side goal is hence to achieve a biological understanding of how reconstructed features map to the brain. Researcher refer to this task as the "neurobiological plausibility of a model". Neighbouring research areas focus on reconstructing videos rather than images but the core approach remains the same.

3 Literature Review

3.1 Overview of Previous Works

The "conventional" approaches rely on using fMRI data directly in the training, together with image features. The main differentiation that ought to be done is between methods that adopt a linear regressors and end-to-end models.

Linear regressors are used to map fMRI signals to the pre-extracted image features. Such mapping is then used to reconstruct images from the brain stimuli in 2 alternative fashions: features reconstructed via the linear regressor are compared to a "database" of deep features to find the closest matching; alternatively optimisation methods are used in which an image is iteratively modified so that its deep feature resemble more and more closely those decoded from the fMRI signal.

As for the end-to-end paradigm a DNN architecture is used to map fMRI **directly** to images (or vice versa), so both the feature extraction and mapping is captures in one go at training time. In general methods that follow this structure achieve reasonably good alignment between the reconstructed image and the original one in term of structural details (in the pixel space) but struggle when it come to the semantic information.

Early attempts adopted **handcrafted features** to reconstruct the visual stimuli from fMRI signals. More recently DNN, trained on vast dataset of naturalistic images, have been used to extract valuable features. With a further step forward, researches attempted to introduce categorical or textual information to guide the training of generative models (e.g. GANs, VAEs) in order to increase the semantic accuracy of the final output. This latter approach is however far form optimal. Indeed, in this specific setting the training of a generative model, already challenging per se, is additionally hindered by the scarcity of training pairs (fMRI, image). The same issue re-propose itself, perhaps amplified, when trying to train an end-to-end model.

To face such hurdle different approaches have been proposed over time. Unsupervised learning emerged as a possible solution. Specifically self-supervision allows for training on unpaired data adopting a re-configurable encoder-decoder architecture. This approach partly solves the issues faced. Indeed the training is no longer limited to the scarce training pairs (fMRI, image) but can leverage also the individual fMRI information. As done in conventional approaches, fMRI signals are then directly used during the training process.

An alternative to the "conventional" approaches (and the self-supervised variations) consist in finding a latent representation[1] of the fMRI instead of using its "raw" version directly (to find a mapping to the image features). The obtained latent space is then used in a generative model for image generation. Early on, regression models have been used to find a latent representation. Over time more sophisticated approaches emerged. CLIP has been used to produce a pre-aligned vision-language latent space. Another promising (self-supervised) technique is Masked Signal Modeling, essentially a denoising autoencoder that reconstruct the original input after it has been masked. At first, for the downstream task of image reconstruction GANs and VACs were the go to models then overtaken by diffusion models (DMs and LDMs). Generally, the downstream diffusion model is then conditioned using various methods such as Cross-attention and time embedding. Compared to the conventional approach, leveraging a latent representation is less computationally intensive and better captures se-

[1]It's important to distinguish between individual feature maps and the overall latent space. A feature map that highlights horizontal edges is a part of the latent representation, but the full latent space usually encompasses a wide range of features and abstractions combined together.

mantic information within the reconstructed image. On the other hand such methods struggle with structural information.

The previously introduced limitation are overcame by "mind diffuser", which seems to be the present state of the art.

3.2 End-to-End Models

In [1] by Shen et al. the approach followed consisted in directly training a model with the fMRI signal and the corresponding stimolus images. The architecture presented is a GAN trained on 6000 training samples and tweaked with an ad-hoc loss term named "feature loss". The main obstacle to such approach is that generally the dimension of the training dataset is considered to be insufficient to train a generative model in an end-to-end fashion. Indeed, whereas in other computer vision domains large scale datasets might be available, when it come to fMRI data, scarcity is a major hurdle. Regardless of the mentioned hurdles training an end-to-end DNN for this task might be beneficial and perhaps more effective than adopting a pre-trained DNN (for feature extraction) as the latter does not consider the information regarding the brain activity (fMRI data). Hence the extracted features might not be ideal for the downstream task of decoding such features from the fMRI data itself. Instead a direct mapping of features and brain activity, theoretically, has the potential to reduce the information loss.

In order to effectively reduce the information loss the choice of a suitable loss function is crucial. Numerous studies have been produced, each of which proposed different solutions:

- Loss in Image Space. Results in poor reconstruction performances (i.e. blurry images) due to the fact that the final output is obtained as an average of all the images having the same distance in the image space.
- Feature Loss. It is also called perceptual loss and has the effect to constrain the produced image to be perceptually similar to the original one.
- Adversarial Loss. Constrained the distribution of the reconstructed image to be close to the distribution of the original input image.

Successive studies have proved how adopting both Feature and Adversarial Loss produces the best results. The architecture adopted In [1] by Shen et al. consists of 3 DNNs: Generator, Discriminator, Comparator. At training time, the generator learns to reconstruct the original image stimulus from the fMRI data (received in input). The Discriminator at every step of the training attempts to differentiate between the original image and the one reconstructed from the Generator. The Comparator produces the feature loss comparing once again the original image and the reconstructed one.

3.3 Regression Based Models

Two slightly different approached have emerged. The first one consists of performing regression between fMRI data and handcrafted image-features. An alternative, and more interesting option, consists in performing regression between fMRI data and Deep image-features. Both these methods consist in computing "a linear model that relates fMRI voxels to image feature representation."[2]. This can be achieved by predicting the brain activity (i.e. voxel's responses) using the features extracted by the images as input; or following the inverse task, mapping the brain activity to the image features. Of course, for this approach to be effective, the feature representation learnt has to be as close as possible to the activity in the visual cortex (allowing for a "1-to-1" mapping). The DNN that serves as a proxy to extract features form the image stimulus is, as previously mentioned, pre-trained on a large image dataset. In the literature body it appears clear how these supervised approaches have poor generalisation performances due the limited amount of training pairs available. An improvement has been proposed by Lin et al. [3]. Their DCNN-GAN follows the previously introduced approach. A pre-trained encoder extracts features from the images. The decoder is built as a linear least squared regression model (regularised with Tikhonov) and the DCNN-GAN which is responsible to reconstruct the image to be outputted. The latter is made up by a reconstruction network that produces a coarse images. Such intermediate image is used , together with the information regrading the class to which the image belongs, as the input for the GAN. Doing so the GAN introduces the information regarding the prior of the image based on the category to which the image to be reconstructed belongs.

3.4 Self-supervised Learning

When dealing with scarce training data and domain adaptation the machine learning community has produced various solutions drawing from different areas such as: transfer learning, unsupervised and self-supervised learning, semi-supervised and transductive learning [2]. However as for these approaches, a widely adopted assumption is that the source domain does have a sufficient amount of labeled data, which is not the case in this setting. Hence the method proposed by Beliy et al. . In [2] it is proposed an architecture that allows to train on unlabelled fMRI data (i.e. unavailable information regarding the pair fMRI-image). Such architecture consists of an "Encoder-Decoder & Decoder-Encoder" structure. The Encoder maps natural images to fMRI responses, and vice versa for the Decoder. Combining the networks in a back-to-back fashion:

- E-D $\longrightarrow$ Unsupervised training on unlabeled images.
- D-E $\longrightarrow$ Unsupervised training on unlabeled fMRI.

Such approach showed improvement in the generalisation capability of the model. The training of the model is performed as follows.

- Training Phase 1:
  An Encoder is trained on the pairs (fMRI, images) where the response is the fMRI to be predicted from the image.
- Training Phase 2:
  The pre-trained encoder is freezed and a decoder is trained using labeled (image-fMRI) pairs (supervised), unlabelled images (unsupervised) and unlabelled fMRI (unsupervised).

3.5 Training with Additional Text Modality

An approach proposed to mitigate the disruptive effect, partly caused by the scarcity of training data, consists in adopting a textual description of the image stimuli during training. [4] In this setting the training sample consists of (fMRI, image, text). The textual descriptive information related to the images is provided by the COCO dataset. The first step consist in mapping fMRI signals to CLIP embeddings (of the image and its caption). Secondly a conditional generative model is used to reconstruct the image. Such generative model is conditioned with the output of the mapping model which is pre-trained on the pairs (fMRI, CLIP embeddings). It is noted that pre-trained model can be adopted making the pipeline more efficient.

3.6 latent representation of fMRI and Diffusion models

[5] by Chen et al. produce what seems to be one of the best performing models. The architecture proposed has 2 stages:

- **Stage A: Sparse-Coded MBM (SC-MBM)** Firstly the fMRI signal is broken down into patches, then section of it are masked. The masked fMRI patches are then fed into a encoder, which is designed to handle the high-dimensional fMRI data and produce a sparse but high-dimensional latent representation. Successively, a decoder, attempts to reconstruct the original, unmasked fMRI data from the latent representation. Through this process (autoencoder), the model learns to create a more accurate and informative latent representation that capture essential information about the fMRI data.
- **Stage B: Double-Conditioned Latent Diffusion Model (DC-LDM)** After the encoder has learnt to produce a meaningful latent representation, this representation is used to condition a U-net that is part of a Latent Diffusion Model (LDM).The U-net receives this conditioned latent representation and operates on an image that has undergone a diffusion process (introducing noise). The U-net's role is to reverse this diffusion process, effectively denoising the image to recover the original visual stimulus.Through the denoising process, the U-net produces the final reconstructed image. The conditioning from the fMRI-derived latent representation guides the U-net to reconstruct an image that is semantically aligned with the brain activity patterns.

It is noted how the (pre-trained) LDM, the fMRI (pre-trained) encoder, the cross-attention [2] heads and projection heads [3] are jointly fine-tuned leveraging fMRI-image pairs. This model achieves high fidelity in terms of preserving semantic information however struggles with pixel level performance as it lacks "pixel-level guidance". It also come short in providing a biological understanding of the features learned by the encoder.

3.7 State of the Art

MindDiffuser presented in [6] by Lu et al. is a recent paper that achieved state of the art performances managing to preserve both semantic and structural information in the image reconstructed from brain signals. The architecture proposed follows 2 stages.

First Stage
The goal of the first stage is to reconstruct an initial image that capture as well as possible the semantic information. Conversely the second stage focuses on aligning the structural information of the reproduced image with the original one. Initially 3 different encoders leverage the input image to produce CLIP text embedding, VQ-VAE latent representation, and the visual feature of layer i in CLIP extracted from the training set[4]. Three linear regression models (decoders) are trained to produce the previously listed outputs from the fMRI signal. The latent representation reconstructed from the decoder firstly undergo a forward diffusion process (that gradually adds noise) and then feeds it into a U-net conditioned with the CLIP text embedding (previously obtained from the decoder) by means of cross-attention. Finally (after VQ-VAE decoding) an image, highly rich in semantic content is outputted.

Second Stage
The Second Stage encompasses an iterative optimisation process. The CLIP text embedding (so far not used) is used as a constraint to continuously improve the CLIP text embedding and the VQ-VAE latent representation to more

[2]In this setup, projection head are used to project the high-dimensional output of the fMRI encoder into a latent space dimension which is suitable for conditioning the LDM. This projection is crucial for retaining the sparsity and information capacity of the fMRI representations, which are important characteristics of fMRI data. By optimizing these projection heads in conjunction with the cross-attention heads, the model can learn a more effective mapping from the fMRI latent space to the image generation space.

[3]In the attention mechanism within U-Net for image reconstruction from fMRI data Q,K,V are used as follows: The network takes a part of the image (or its abstract representation) as a query. It then uses the encoded fMRI data as keys to determine which features of the brain activity are most relevant to the part of the image we're querying. The values would be aligned with the keys and contain the information necessary to help reconstruct or enhance the image based on the fMRI data. Cross-attention is applied by calculating the dot product of the queries and keys (to find relevance), applying a softmax to get probabilities (which can be seen as attention weights), and then using these weights to create a weighted sum of the values. This results in an output that is a combination of the relevant information from the values based on the input queries and the context provided by the keys.

[4]The idea is that these features can provide a bridge between the raw pixel space of images and the complex, high-level cognitive processes represented by fMRI signals.

closely resemble the original input in terms of structure and semantic. Using the architecture from stage 1 an image is produced and fed to the CLIP image decoder. The same is done feeding the original image to the CLIP image decoder. The two outputs are then compared producing a "structure loss". Such loss is back-propagated adjusting both the CLIP text embedding and the VQ-VAE latent representation which are then fed to the image generator (of stage 1) to produce an updated image. The loop is repeated until convergence.

An interesting result produced by the authors of [6] shows the "upper bound" image reconstructed from the **true** CLIP text embedding and the VQ-VAE latent representation. It can be seen how the image reconstructed form such input resembles really closely the original image. This proves that the model produced achieves a very high reconstruction ability and suggests that the bottleneck, at this point, might be represented by the accuracy in encoding the brain activity in the latent space.

4 Dataset

The challenge of collecting extensive labeled Image, fMRI datasets is substantial due to the constraints on how long a subject can be scanned within an MRI machine. Consequently, the majority of available datasets contain only a few thousand of these labeled pairs. This quantity is insufficient to cover the vast diversity of natural images and the corresponding breadth of fMRI responses. Additionally, the intrinsic limitations of fMRI technology, such as its modest spatio-temporal resolution and low signal-to-noise ratio (SNR), compromise the quality and interpretability of the limited data we can gather. Compounding this, variations in SNR and other statistical characteristics between the training and testing sets pose further challenges to model generalization.

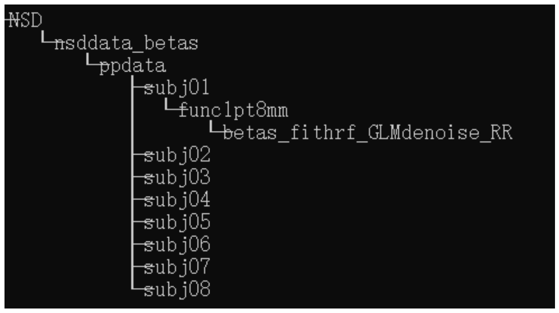
To address the intensive data requirements of modern deep learning frameworks, the Natural Scenes Dataset (NSD) was developed, boasting an unparalleled scale of sampling, superior resolution, and improved SNR compared to previous datasets. Furthermore, the NSD sources its imagery exclusively from the MS-COCO database, ensuring a broad and diverse collection of visual stimuli and the availability of captioned images. NSD data were recorded over the course of 30–40 sessions in which three repetitions of 10,000 images were shown to each of the 8 subjects.

For practical processing, the following approach is typically adopted: Each fMRI signal associated with a distinct image constitutes a three-dimensional data array, where the value at a specific coordinate (i, j, k) represents the level of brain activation in that voxel due to the stimulus. To isolate the most relevant signals, a region of interest (ROI) mask is applied to this volumetric data (focusing on cortical voxels pertinent to the task with acceptable SNR levels). In addition, to distill a more stable representation, each fMRI measurement is temporally averaged, yielding a consolidated voxel activation profile.

In [7] both the early visual cortex and the higher visual cortex (ventral cortex) are used. More precisely the first brain area is used to decode image structures, whereas the second one is used for decoding semantic information.

The URL here after provides a detailed video description of the file directory for the Natural Scene Dataset `https://www.youtube.com/watch?v=LfHowycsXLI`

The specific fMRI data used by MindDiffuser are fMRI signal pre-processed as "betas_fithri_GLMdenoise_RR" as can be see in figure **??** which is directly taken form the GitHub-repo of the project.



```
NSD
  └nsddata_betas
      └ppdata
          ├subj01
          │  └func1pt8mm
          │      └betas_fithrf_GLMdenoise_RR
          ├subj02
          ├subj03
          ├subj04
          ├subj05
          ├subj06
          ├subj07
          └subj08
```

5 Diffusion Models Bsckground

5.1 Introduction

Following the extensive literature review presented in the previous paragraph I hereafter focus on a specific type of generative model, Diffusion Models. The choice is guided by two main observations. To begin with the generative model is at the very heart of the pipeline required to successfully undertake the task analysed in this paper. Indeed, the ultimate goal being to reproduce a visual stimuli requires an efficient and effective generative model. Secondly, Diffusion Models have recently emerged as the leading approach to generative tasks beating the state of the art results previously detained by GANs [8]. Hence, my interest in focusing on this recent technology to try and dive as deep as possible into the theoretical backbone and practical implementation. Moreover I have extended my effort into investigating how the various conditioning mechanism work. As a result I have implemented in PyTorch, from the ground up, a Latent Diffusion Model. Such model leverages the pre-trained weights of the original stable diffusion project [7] that can be conditioned by a positive and negative textual prompt as well as a mask for in-painting tasks. The model I implemented can be used for instance and can as well be fine tuned on a specific dataset improving the generative performances achieved.

There are three main types of diffusion Models: GAN, VAE, and Diffusion Models. GAN models might be unstable during training and suffer from a reduced diversity in generation (due to their adversarial training methodology).

VAEs rely on a surrogate loss. Denoising Diffusion Probabilistic Models comprise 2 main stages: the Forward and the Reverse diffusion process.

5.2 Forward Diffusion Process

The Forward Diffusion Process consist in gradually adding Gaussian Noise to an input until the point where we are left with only noise. It is important to notice how the noise is not added all at once but rather in small amounts over the course of T step. The step sizes are controlled by what is called a schedule. [9]

$$q(x_t|x_{t-1})x = \mathcal{N}(\underbrace{x_t}_{output}; \underbrace{\sqrt{1-\beta_{t-1}},}_{Mean} \underbrace{\beta_t I}_{Variance}) \tag{1}$$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{2}$$

Thanks to a reparametrisation trick it possible to sample x_t at any time step t in a closed form as can be seen in the following formula:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t x_0}, (1-\overline{\alpha}_t)I) \tag{3}$$

5.3 Reverse Diffusion Process

To goal is to reverse the previous process so that we can sample from $q(x_{t-1}|x_t)$ and thereafter recover the original (clean) sample from the noisy one. It should be noted how if β_t is small enough, then $q(x_{t-1}|x_t)$ will also have a Gaussian distribution. However, $q(x_{t-1}|x_t)$ can not be estimated, therefore we learn a (denoiser) model that approximates it.

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

It should be noticed how estimating the reverse conditional probability $q(x_{t-1}|x_t)$ usually requires knowledge about the entire trajectory of the forward process, which is computationally expensive or even infeasible. If, however, the model is also conditioned on the original data point x_0, the estimation becomes tractable, meaning that it can be computed more efficiently. This is because having access to x_0 allows for a better prediction of what x_{t-1} should be, since x_0 provides a reference point for what the data looked like before any noise was added.

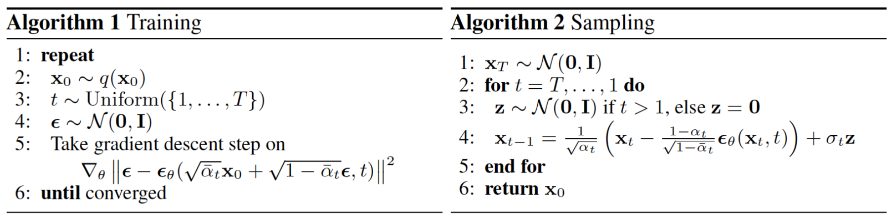
5.4 Loss Function

For a dataset X and a model parameterized by θ, the true log-likelihood is $\log_p \theta(X)$. In many cases, directly computing $p_\theta(X)$ is intractable because it involves integrating over all possible configurations of latent variables Z (in the case of VAEs) or all possible noise patterns (in the case of diffusion models). The variational lower bound is introduced as an auxiliary function which is easier to compute and is used to approximate the intractable true posterior. Ultimately the variational lower bound is used to derive a tractable loss function.

Empirically Ho et al.(2020) [9] found that diffusion models can be effectively trained adopting the following simplified loss function (where the waiting therm is ignored):

$$L_t^{\text{simple}} = \mathbb{E}_{t\sim[1,T]|x_0,\epsilon}\left[\|\epsilon_t - \mathcal{E}_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t)\|^2\right] + C$$

The overall algorithm can be see in the following figure. The first Algorithm consist in an iterative process that is repeated until convergence. Throughout such a process a sample is drawn from the clean data distribution, then a varying amount of noise (at a given time step) is applied. Gradient descent is used to minimise the MSE between the estimated noise and the actual noise sampled. The second Algorithm consist in sampling x_t from the end state of the diffusion process. Then the learned noise prediction model is used to recover x_{t-1}. The process is repeated until $t = 1$, hence all the noise has been removed and the original clean distribution has been recovered.



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \text{Uniform}(\{1,\dots,T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
6: until converged
```

Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
2: for $t = T,\dots,1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

5.5 Text-to-Image techniques

Conditioning (or Guidance) can be achieved leveraging class labels, text, images and other.

One of the earliest mechanism adopted to implement guidance is known as Classifier Guidance.

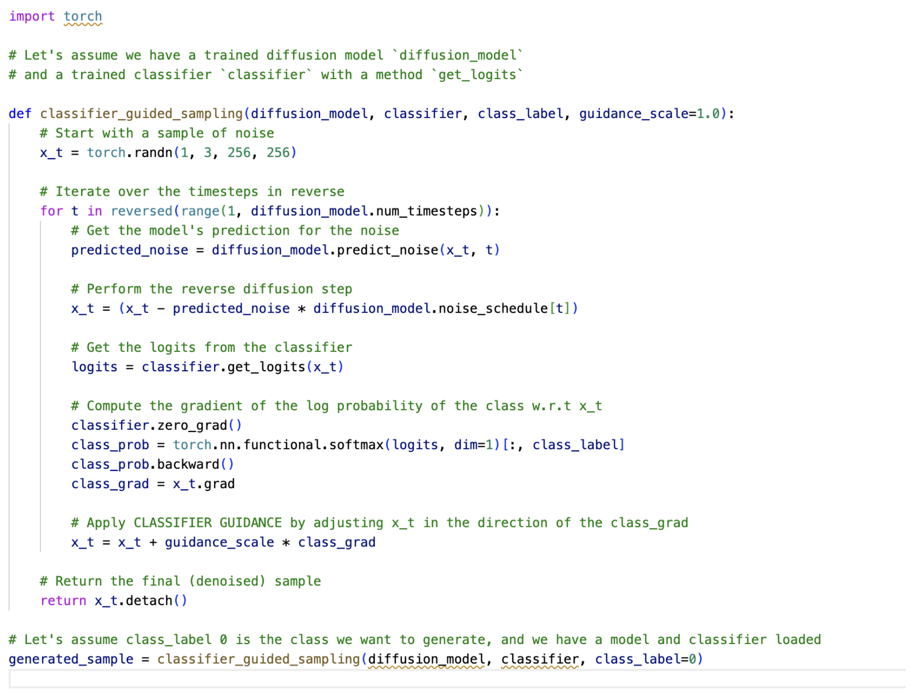
Working in pixel space is to computationally expensive so we work in a latent space (in which the original data is compressed)

Conditional diffusion models make use of an additional input, compared to the "normal" DDPM. The signal y (eg. a class label or a text sequence) is used to model the conditional distribution p(x | y) instead. This allows us to generate data conditioned not only from the image from the previous time-step but also by y as follows: $p(x_{t-1}|x_t, y)$.

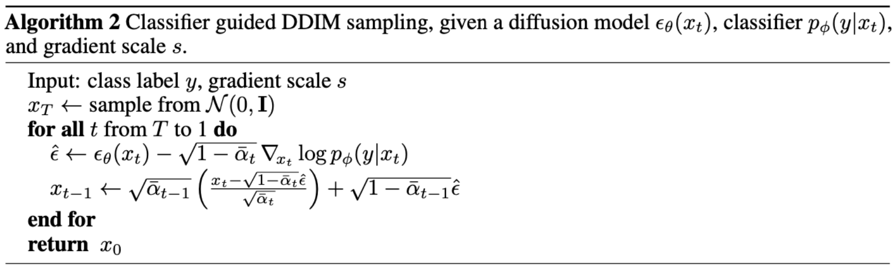
In the following subsection different type of guidance are introduced.

5.5.1 Classifier Guidance

Classifier guidance is a technique used to steer the generative process of a diffusion model towards creating samples that are more likely to belong to a certain class as determined by a classifier. This is particularly useful in conditional generation tasks where you want the model to generate samples with specific attributes or classes. The general idea is to use a trained classifier in conjunction with the diffusion model to bias the generation process. During the reverse diffusion steps (when generating new samples), the gradient of the log probability with respect to the data, given the desired class, is added to the generation process. This gradient is obtained from the classifier and points in the direction that would make the sample more likely to be classified as the desired class. During the sampling process of the diffusion model, at each step, you compute the gradient of the log probability of the class given by the classifier with respect to the current sample. You then adjust the sample in the direction of this gradient. There is usually a hyperparameter that controls the strength of the guidance. If this parameter is set too high, the model may ignore the structure learned during the unsupervised training and simply follow the classifier's gradients, potentially leading to less diverse or lower-quality samples.



```
import torch

# Let's assume we have a trained diffusion model `diffusion_model`
# and a trained classifier `classifier` with a method `get_logits`

def classifier_guided_sampling(diffusion_model, classifier, class_label, guidance_scale=1.0):
    # Start with a sample of noise
    x_t = torch.randn(1, 3, 256, 256)

    # Iterate over the timesteps in reverse
    for t in reversed(range(1, diffusion_model.num_timesteps)):
        # Get the model's prediction for the noise
        predicted_noise = diffusion_model.predict_noise(x_t, t)

        # Perform the reverse diffusion step
        x_t = (x_t - predicted_noise * diffusion_model.noise_schedule[t])

        # Get the logits from the classifier
        logits = classifier.get_logits(x_t)

        # Compute the gradient of the log probability of the class w.r.t x_t
        classifier.zero_grad()
        class_prob = torch.nn.functional.softmax(logits, dim=1)[:, class_label]
        class_prob.backward()
        class_grad = x_t.grad

        # Apply CLASSIFIER GUIDANCE by adjusting x_t in the direction of the class_grad
        x_t = x_t + guidance_scale * class_grad

    # Return the final (denoised) sample
    return x_t.detach()

# Let's assume class_label 0 is the class we want to generate, and we have a model and classifier loaded
generated_sample = classifier_guided_sampling(diffusion_model, classifier, class_label=0)
```



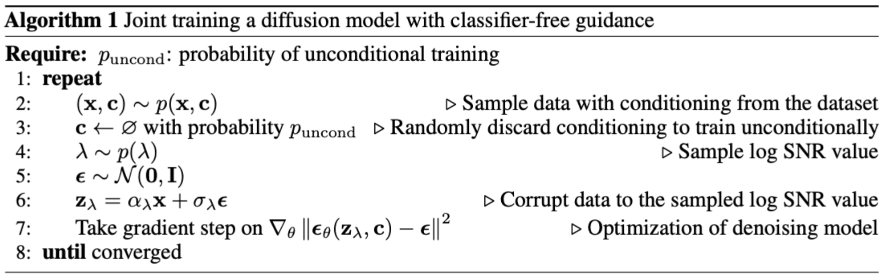
Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s.

```
Input: class label $y$, gradient scale $s$
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all $t$ from $T$ to 1 do
    $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$
    $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left( \frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return $x_0$
```

Figure 1: Algorithm from Dhariwal et al. 2021 [8]

5.5.2 Classifier-free guidance

Classifier-free guidance is a way to train a single model that can function both conditionally and unconditionally balancing out coverage and sample fidelity. During training, the conditioning signal (such as a class label) is randomly dropped with a certain probability. This means the model learns to generate data both with and without the conditioning signal. This method simplifies the training process because it avoids the need to train a separate classifier; moreover such a classifier would need to be trained on noisy data which precludes the possibility of adopting a pre-trained one.



Algorithm 1 Joint training a diffusion model with classifier-free guidance

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

Figure 2: Algorithm from Ho et al. 2022 [10]

To adapt this for generic text prompts instead of specific labels, the same principle is applied. During training, the model occasionally replaces text captions with an empty sequence to simulate the absence of a label. When it comes time to guide the model towards generating content based on a text prompt. The equation used to predict the noise level is as follows (where c represents a generic text prompt)[11]:

$$\hat{e}_\theta(x_t|c) = e_\theta(x_t|\phi) + s \cdot (e_\theta(x_t|c) - e_\theta(x_t|\phi)) \tag{4}$$

At inference time, on the other hand, for each de-noising step the model produces 2 different outputs. One conditioned via the text prompt and an unconditioned one. Then the 2 output are combined as shown in equation 5 with a weight that expresses how much attention to pay to the textual prompt

$$\text{output} = w \cdot (\text{output}_{\text{conditioned}} - \text{output}_{\text{unconditioned}}) + \text{output}_{\text{unconditioned}} \quad (5)$$

Classifier-free Guidance, when used in combination with a textual prompt, has shown to be an effective method to retain the desired semantic information in the output. Although a textual prompt is hardly effective in constraining the structural content of the reconstructed image.

Q.Avrahami et al. in [12] presented an interesting implementation of classifier-free guidance that does indeed allow to achieve a more fine grained structural control over the output. The novelty they introduce is referenced to as a CLIP-based-spatio-textual representation. In essence at training time a panoptic segementation model (pre-trained) is used to extract local regions, which are then fed as input to a CLIP image encoder.

5.5.3 Classifier Based Score-model based guidance

Score Model Based Guidance are an attempt to improve the results achieved with classifier guidance. Indeed the latter have the "tendency to overfit without coordinating with the underlying unconditional distribution." [13]. A pre-trained score model is used as guidance. Given a conditional model $p(x_t|y)$, the gradients from an extra score model $p(y|x_t)$ are used during the sampling phase (i.e. to de-noise the input).

5.5.4 CLIP Guidance

"CLIP embeds text and image in the same space via a projection layer. Thus, it can efficiently learn visual concepts, in the form of text, via natural language supervision and perform zero-shot classification." [14]

A CLIP architecture is composed of two distinct elements: an image encoder denoted as f(x) and a text encoder represented by g(c). The training process involves selecting pairs of images and captions (x, c) from a vast dataset. The goal of the training is to fine-tune the model using a contrastive cross-entropy loss function. This function is designed to maximize the dot product f(x) · g(c) when an image x is correctly matched with its associated caption c, and to minimize it when the pairing is incorrect. When integrating CLIP into diffusion models for guidance purposes, the typical classifier is substituted with the CLIP model. Specifically, we modify the mean of the reverse process by using the gradient of the dot product between the image and caption encodings in relation to the image. In a process akin to classifier guidance, it's crucial to train the CLIP model using images with added noise, denoted as x_t, to ensure the accuracy of the gradient used during the reverse phase. [11]

$$\hat{\mu}_\theta(x_t|c) = \mu_\theta(x_t|c) + s \cdot \Sigma_\theta(x_t|c) \nabla_{x_t} \log(f(x_t) \cdot g(c)) \quad (6)$$

It shall be noted how classifier-free-guidance tends to produce more realistic compared to those produced via the CLIP-guidance method aforementioned (i.e. GLIDE).

An alternative approach (which is also adopted in MindDiffuser) is proposed by Ramesh et al.(2022). They proposed a 2 stages architecture called unCLIP that encompasses a Prior and a Decoder. The Prior produces CLIP image embedding z_i conditioned on captions y. Whereas the Decoder produces images x conditioned on CLIP image embedding z_i and captions y. The prior is used to produce a generative model of the image embedding whilst the decoder is leveraged to reproduce images given their CLIP embedding. Throughout the training, using 2 separate encoders, we learn a representation that combines both the image and its caption. The model is trained to maximize the similarity between correct text-image pairs and minimize it for mismatched pairs (i.e. CLIP objective). Subsequently the Prior (eg. a diffusion model which starts with noise and gradually shapes that noise into an image embedding that corresponds to the text) takes the text embedding produced by the CLIP text encoder and generates an intermediate image embedding. Once the prior has generated an image embedding, the decoder takes over to produce the final image. The decoder is a diffusion model conditioned on the image embedding. It works by starting with noise and gradually refining that noise into an image, guided by the image embedding from the prior.[15].

6 Stable Diffusion Implementation

6.1 Overview

The model developed leverages the pre-trained weights made available from the original stable diffusion project [7] and downloadable on Hugging Face. Moreover the model includes a fine tuning method that allows toe re-train on a specific dataset using the original weights as initialisation. The model developed can be used at inference time for Text-To-Image, Image-to-Image, and In-painting tasks. The various modules are implemented in PyTorch and closely follow the architecture suggested by [7].

6.2 Architecture

The piepline implemented includes 3 separate modules: a Variational Auto Encoder (VAE), a CLIP-encoder and a U-Net.

The overall model implemented is known as a latent diffusion model. This mean that instead of learning the distribution p(x) of the images in a dataset we learn the distribution of the respective latent representation. This is extremely convenient in term of computational efficiency compared to working in the pixel space. Indeed we can now perform the forward diffusion process and the sampling steps in feature space that is arbitrarily compressed compared to the original one.

The VAE is used to produce a latent space from an image. A latent space is a feature space that represents the parameter of a multivariate distribution. Doing so we capture the semantic relationship existing among the data. The CLIP (Contrastive Lenguage-Image Pre-Training) encoder sostantially consists of 2 encoder that produce an encoding of an image and the relative caption, successively the 2 feature spaces are aligned so that the model learns which features in text correspond to which features in image. Lastly a U-net is adopted with the classic down sampling and up sampling stages. More in details the U-net consists in a series of convolutional, residual and attention (self-attention and cross-attention) blocks.

Image **??** shows how the various model interact within the pipeline. In particular there are 4 inputs that varies depending on the specific inference task being performed. Input 1 consists or random noise for Text-to-Image task, whereas it consists of the original image to be modified in Image-To-Image and In-Painting tasks. Input 2 is the prompt based on which an image will be generated, in case of Text-to-Image; or based on which an image will be modified in the case of Image-to-Image and In-Painting. Input 3 is the sinusoidal embedding of the information regarding the time-step. This information is crucial for the U-net in order to correctly predict the amount of noise that has been added to the input. Lastly input 4 is used only for In-Paint and consists of the original image to be modified and the mask to be applied in order to specify which parts of the image should not be modified and which should. For what concerns the U-net, it receives 4 inputs. The latent representation deriving from input 1. The text embedding which is used to condition the generative model. The time embedding and, at inference time, the output of the previous iteration passed over by the scheduler.

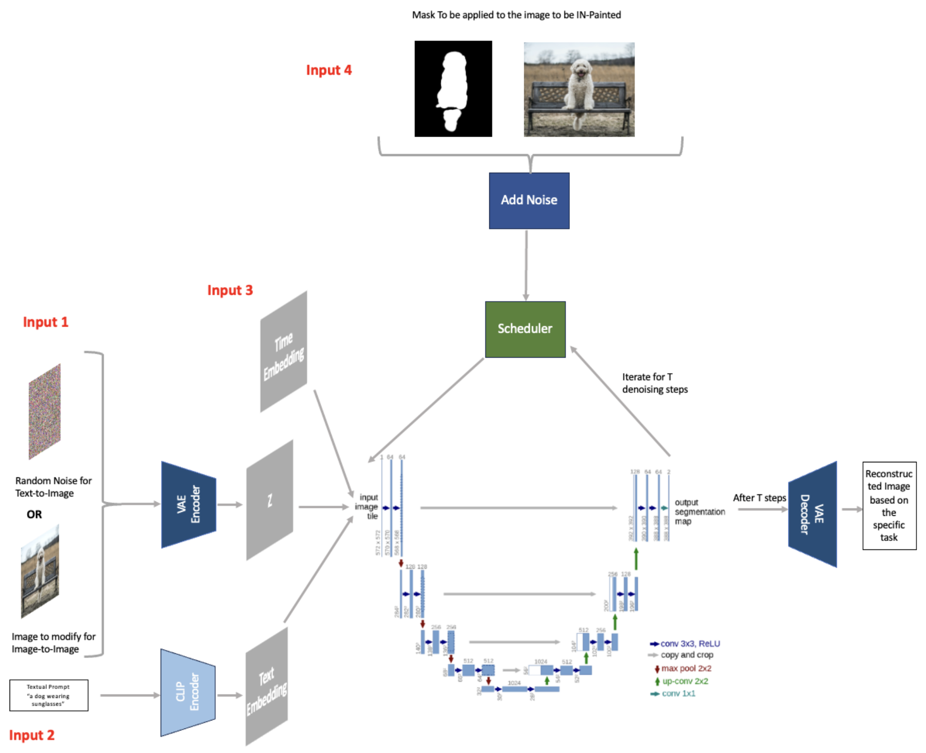

Mask To be applied to the image to be IN-Painted
Input 4
Add Noise
Input 3
Input 1
Time Embedding
Scheduler
Iterate for T denoising steps
Random Noise for Text-to-Image
OR
VAE Encoder
Z
input image tile
output segmentation map
After T steps
VAE Decoder
Reconstructed Image based on the specific task
Image to modify for Image-to-Image
Textual Prompt "a dog wearing sunglasses"
CLIP Encoder
Text Embedding
Input 2
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Figure 3: Architecture of Stable diffusion

6.3 Results

In Figure are reported some of the results obtained using the model for inference. It is noted how the number of inference steps adopted is 50 and the parameter determining how much attention to pay to the textual prompt is set at 8 on a scale from 1 to 14. The higher this hyper-parameter the more closely the prompt content will be represented in the output. For Image-to-Image and In-Painting the noise added to the original input is set to 0.7 on a scale from 0 to 1. The higher the strength the more drastic would be the modification to the original image.

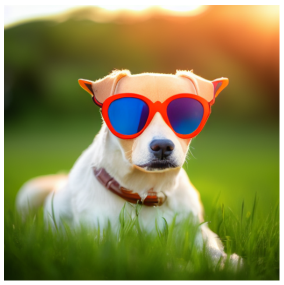
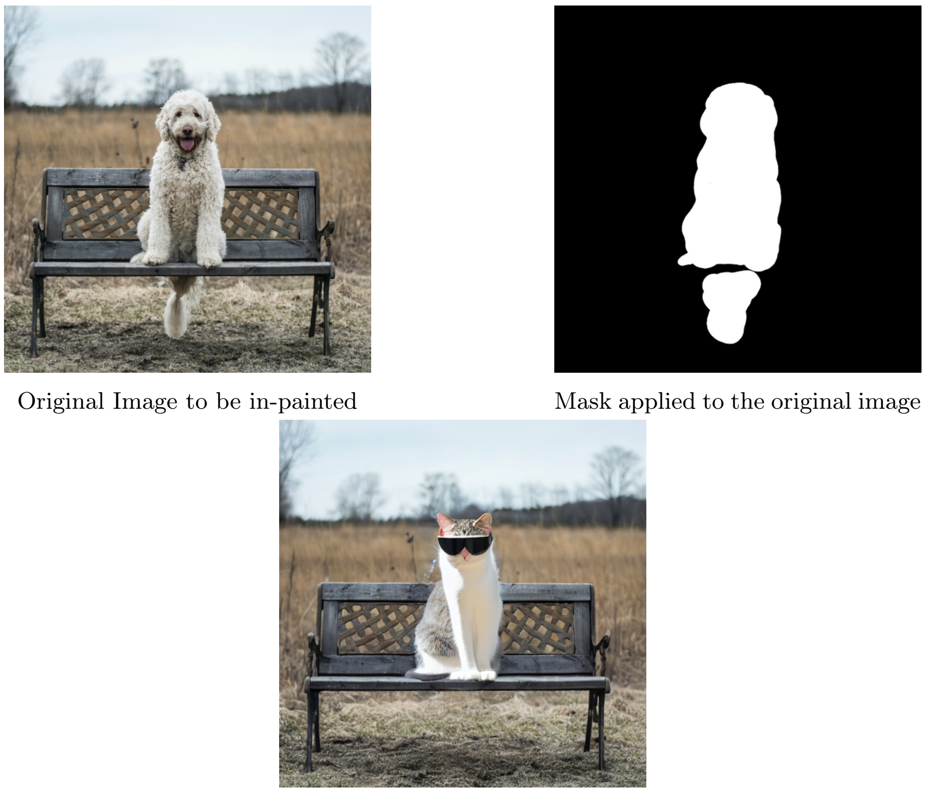


Figure 4: Text-to-Image result. The prompt used was "A dog with sun glasses in a grass field, highly detailed, ultra sharp, cinematic, 100mm lens, 8k resolution."



Original Image to be in-painted

Mask applied to the original image

In-painted image using the prompt "A cat sitting on a bench wearing sun glasses, ultra sharp, cinematic, 100mm lens, 8k resolution."

Figure 5: Results from In-Painting

References

[1] Guohua Shen, Kshitij Dwivedi, Kei Majima, Tomoyasu Horikawa, and Yukiyasu Kamitani. End-to-end deep image reconstruction from human brain activity. *Frontiers in Computational Neuroscience*, 13, 2019. This article is part of the Research Topic Frontiers in Computational Neuroscience – Editors' Pick 2021.

[2] Roman Beliy, Guy Gaziv, Assaf Hoogi, Francesca Strappini, Tal Golan, and Michal Irani. From voxels to pixels and back: Self-supervision in natural-image reconstruction from fmri. *Neural Information Processing Systems (NeurIPS)*, 2019. arXiv:1907.02431 [eess.IV].

[3] Yunfeng Lin, Jiangbei Li, and Hanjing Wang. Dcnn-gan: Reconstructing realistic image from fmri. *Journal/Conference Name*, 2023.

[4] Sikun Lin, Thomas Sprague, and Ambuj K Singh. Mind reader: Reconstructing complex images from brain activities. 2022.

[5] Zijiao Chen, Jiaxin Qing, Tiange Xiang, Wan Lin Yue, and Juan Helen Zhou. Seeing beyond the brain: Conditional diffusion model with sparse masked modeling for vision decoding. *arXiv preprint arXiv:2211.10965*, 2023.

[6] Yizhuo Lu, Changde Du, Qiongyi Zhou, Dianpeng Wang, and Huiguang He. Minddiffuser: Controlled image reconstruction from human brain activity with semantic and structural diffusion. In *Proceedings of the 31st ACM International Conference on Multimedia*, page 5899, Ottawa, ON, Canada, 2023. ACM. Available at: https://github.com/RedOnePeck/MindDiffuser.

[7] Yu Takagi and Shinji Nishimoto. High-resolution image reconstruction with latent diffusion models from human brain activity. *bioRxiv*, 2023. Preprint at `https://doi.org/10.1101/2022.11.18.517004`.

[8] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, Jun 2021. Accessed: 2024-01-10.

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Journal or Conference Name*, Volume Number(Issue Number):Page Range, 2024.

[10] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, Jul 2022. Accessed: 2024-01-11.

[11] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741v3*, cs.CV, Mar 2022.

[12] Omri Avrahami, Thomas Hayes, Oran Gafni, Sonal Gupta, Yaniv Taigman, Devi Parikh, Dani Lischinski, Ohad Fried, and Xi Yin. Spatext: Spatio-textual representation for controllable image generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2023.

[13] Paul Kuo-Ming Huang, Si-An Chen, and Hsuan-Tien Lin. Score-based conditional generation with fewer labeled data by self-calibrating classifier guidance. *arXiv preprint arXiv:2307.04081v2*, cs.CV, Sep 2023.

[14] Ziyou Yan. Text-to-image: Diffusion, text conditioning, guidance, latent space. *eugeneyan.com*, Nov 2022.

[15] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125v1*, cs.CV, Apr 2022.