

# Greedy-DiM: Greedy Algorithms for Unreasonably Effective Face Morphs

Zander W. Blasingame  
Clarkson University  
Potsdam, NY, USA  
blasinzw@clarkson.edu

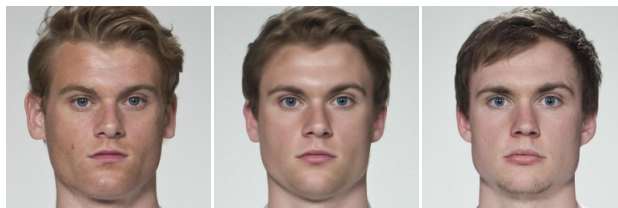
Chen Liu  
Clarkson University  
Potsdam, NY, USA  
cliu@clarkson.edu

## Abstract

*Morphing attacks are an emerging threat to state-of-the-art Face Recognition (FR) systems, which aim to create a single image that contains the biometric information of multiple identities. Diffusion Morphs (DiM) are a recently proposed morphing attack that has achieved state-of-the-art performance for representation-based morphing attacks. However, none of the existing research on DiMs have leveraged the iterative nature of DiMs and left the DiM model as a black box, treating it no differently than one would a Generative Adversarial Network (GAN) or Variational AutoEncoder (VAE). We propose a greedy strategy on the iterative sampling process of DiM models which searches for an optimal step guided by an identity-based heuristic function. We compare our proposed algorithm against ten other state-of-the-art morphing algorithms using the open-source SYN-MAD 2022 competition dataset. We find that our proposed algorithm is unreasonably effective, fooling all of the tested FR systems with an MMPMR of 100%, outperforming all other morphing algorithms compared.*

## 1. Introduction

Face Recognition (FR) systems have been deployed in a wide range of settings from unlocking phones to border control [1], and the performance of such systems keeps improving. Due to its wide adoption, FR systems have been the target to various types of attacks [2]. Most recently, it has been shown that these systems are vulnerable to the face morphing attacks [3–7], an emerging type of attack that aims to blend the biometric identifiers of multiple faces into a single image that would trigger a false acceptance with any of the identities used to make the morph, see Figure 1 for an illustration. Application scenarios where the system allows users to submit self-generated images to obtain a valid passport or visa creates a significant attack vector for a malicious agent to enroll morphed images [8]. The potency of these attacks have led to face morphing becoming a focal



(a) Identity *a* (b) Morphed image (c) Identity *b*  
Figure 1. Example of a morph created using Greedy-DiM. Samples are from the FRLI dataset [15].

point in recent research on FR systems [9–14], in an effort to secure the identity control process.

Traditional morph generation methods are based on pixel-level features, *e.g.*, aligning facial landmarks, which could result in numerous artefacts [16]. Recent work has focused on using generative AI models for representation-based attacks as the morphing occurs in the representation space rather than the image space. Generative Adversarial Networks (GANs) [6, 16–18] can generate powerful morphs with less artefacts, when compared to the pixel-level morphs [18]. However, diffusion models have come to supersede GANs as the state-of-the-art backbone for generative AI research [19]. Recent work has shown that **Diffusion Morphs (DiM)** can achieve state-of-the-art morphing performance [20–22]. While GAN-based morphs blend the latent representations of two component identities, either by an element-wise average or by optimizing the blended latent with an identity-based loss [17], the latent representation of diffusion models is not inherently suited for this process for two reasons. One, the latent space is a high dimensional space with the same size as the image space and two, the forward encoding process is stochastic and not deterministic. To remedy this, the primary neural network backbone in the diffusion model is conditioned on a conditional embedding of the target image and a deterministic forward pass [23] is used to encode the target image. These twin representations, the conditional embedding and the deterministic latent, are then used in the morphing process. The current state-of-the-art methods sim-

ply blend the two latent representations, either with a fixed blend value [20, 21] or a large batch of morphs with different blend values [22], with the best performing blend value selected out of the batch.

While GAN-based methods can directly optimize the latent representation used to produce the morphed image [17], the iterative generative process of diffusion models, often requiring many steps to sample a high quality sample [20], does not immediately provide such an analog. We propose to use a greedy strategy to solve this optimization problem by locally choosing the optimal solution at each timestep in the sampling process. We call the family of algorithms which use the greedy strategy: Greedy-DiM. Greedy-DiM optimizes the diffusion process for face morphing by using a greedy search strategy in order to create high-quality morphs while retaining minimal computational overhead when compared to the original DiM approach [20] or less overhead than later work [22]. We summarize our contributions in this work as follows:

1. We propose a novel family of face morphing algorithms, Greedy-DiM.
2. We present thorough experimental and theoretical analysis of the proposed morphing algorithms.
3. To the best of our knowledge Greedy-DiM is the first representation-based morphing algorithm that *consistently* outperform landmark-based morphing algorithms.

## 2. Prior Work

The naïve approach to create a face morph is to simply take a pixel-wise average of the original images. Due to its simplicity, this approach can result in many noticeable artifacts. An extension on this would be to align the landmarks of the face, warping the pixels to fit this new representation, and then averaging the pixels. The blending value is usually set to 0.5 so that both of the original images could contribute equally to the morphed image. Later work [17, 22] questions if this assumption is valid. Further improvements can be made to these landmark-based morphs by additional post-processing to remove the artefacts from the morphing process [24].

Morphing through Identity Prior driven GAN (MIPGAN) and in particular MIPGAN-II improve upon the original GAN-based morphs [25] and StyleGAN2-based morphs [16] by using optimization methods to search for a latent representation that fools the FR system maximally [17]. The MIPGAN algorithm works by taking the two bona fide images  $x_a, x_b$ , and encoding them into latent representations,  $z_a, z_b$ . An initial morphed latent  $z_{ab} = \frac{1}{2}(z_a + z_b)$  is then constructed. The initial latent is optimized

such that the generated morph  $x_{ab} = G(z_{ab})$  minimizes an identity loss based on the ArcFace FR system [26] along with a perceptual loss term. This optimization procedure produces a more potent morphing attack than simply using the morphed latent  $z_{ab}$ .

### 2.1. DiM

Blasingame *et al.* [20] proposed to use diffusion-based methods to construct high-quality face morphs. This family of morphing attacks are known as Diffusion Morphs (DiM). Diffusion models function by modeling the diffusion process wherein noise is progressively added to the data until the resulting distribution reassembles a Gaussian distribution [27, 28]. To sample the data distribution, a random sample is drawn from the Gaussian distribution and then the diffusion process is run in reverse to iteratively remove noise until a clean sample from data distribution is produced. Given the data distribution  $p_{data}(\mathbf{x})$  on the data space  $\mathcal{X} \subseteq \mathbb{R}^n$ , the diffusion process is given by the Itô stochastic differential equation (SDE)

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t) d\mathbf{w}_t \quad (1)$$

where  $t \in [0, T]$  is the time with fixed constant  $T > 0$ ,  $f(\cdot)$  and  $g(\cdot)$  are the drift and diffusion coefficients, and  $\mathbf{w}_t$  denotes standard Brownian motion. In this paper we employ the variance preserving (VP) formulation of diffusion models [29] which uses the following drift and diffusion coefficients

$$f(t) = \frac{d \log \alpha_t}{dt} \quad (2)$$

$$g^2(t) = \frac{d\sigma_t^2}{dt} - 2 \frac{d \log \alpha_t}{dt} \sigma_t^2 \quad (3)$$

where  $\alpha_t, \sigma_t$  form the noise schedule of the diffusion model such that

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t | \alpha_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}) \quad (4)$$

with a signal-to-noise-ratio (SNR)  $\alpha_t^2/\sigma_t^2$ , which is strictly decreasing with respect to  $t$ .

The distribution of  $\mathbf{x}_t$  is denoted as  $p_t(\mathbf{x}_t)$ , therefore  $p_0(\mathbf{x}_0) \equiv p_{data}(\mathbf{x})$ . Song *et al.* [29] showed the existence of an ordinary differential equation (ODE) dubbed the Probability Flow ODE (PF-ODE) whose marginal distributions  $p_t$  follow the same trajectories as the diffusion SDE. The PF-ODE is given as

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t) \quad (5)$$

where  $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$  denotes the score of  $p_t(\mathbf{x}_t)$ . The score function can be learned by a neural network  $\epsilon_{\theta}(\mathbf{x}_t, t) \approx -\sigma_t \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$ . To sample the model, an initial noise

Table 1. Comparison of existing DiM methods in the literature and our proposed algorithm.

	DiM [20]	Fast-DiM [21]	Morph-PIPE [22]	Ours (Greedy-DiM)
ODE solver	DDIM	DPM++ 2M	DDIM	DDIM
Forward ODE solver	DiffAE	DDIM	DiffAE	DiffAE
Number of sampling steps	100	50	2100	20
Heuristic function	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{L}_{ID}^*$	$\mathcal{L}_{ID}^*$
Search strategy	$\mathcal{X}$	$\mathcal{X}$	Brute-force search	Greedy optimization
Search space	$\emptyset$	$\emptyset$	Set of 21 blend values	Image space
Probability the search space contains the optimal solution	$\mathcal{X}$	$\mathcal{X}$	0	1

sample  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  is drawn. Then a numerical ODE solver is employed to solve the ODE from time  $T$  to time 0.

DiM models employ score predictor model conditioned on a latent representation of the target image, *i.e.*,  $\epsilon_\theta(\mathbf{x}_t, \mathbf{z}, t)$  with an encoding network  $\mathbf{z} = \mathcal{E}(\mathbf{x}_0)$ . Previous DiM approaches [20–22] use a Diffusion Autoencoder model [23] as the backbone for the score prediction model. As per the Diffusion Autoencoder technique, the ODE solver  $\Phi$  is reversed now solving from  $t = 0$  to  $t = T$ , resulting in a deterministic forward pass to an approximation of  $p_T(\mathbf{x}_T)$ . We denote this forward ODE solver, so called as it is evaluated as time runs forwards from 0 to  $T$ , as  $\Phi^+$ . The noisy bona fide images are morphed using spherical linear interpolation, *i.e.*,  $\mathbf{x}_T^{(ab)} = \text{slerp}(\mathbf{x}_0^{(a)}, \mathbf{x}_0^{(b)}; 0.5)$ . The conditionals are averaged and this morphed information is used as the input to the diffusion model. The DiM generative pipeline is provided in Appendix B.1.

Zhang *et al.* [22] proposed Morph-PIPE, a simple extension on the DiM method by using an identity-based loss to select between a range of blend values. They use an identity loss defined as the sum of two sub-losses:

$$\mathcal{L}_{ID} = d(v_{ab}, v_a) + d(v_{ab}, v_b) \quad (6)$$

$$\mathcal{L}_{diff} = |d(v_{ab}, v_a) - d(v_{ab}, v_b)| \quad (7)$$

$$\mathcal{L}_{ID}^* = \mathcal{L}_{ID} + \mathcal{L}_{diff} \quad (8)$$

where  $v_a = F(\mathbf{x}_0^{(a)})$ ,  $v_b = F(\mathbf{x}_0^{(b)})$ ,  $v_{ab} = F(\mathbf{x}_0^{(ab)})$ , and  $F : \mathcal{X} \rightarrow V$  is an FR system which embeds images into a vector space  $V$  which is equipped with a measure of distance,  $d$ . To create the morph  $B$ , different morphs are generated with different blend coefficients  $\{w_n\}_{n=1}^B \subseteq [0, 1]$ . They then choose the  $w_n$  which minimizes  $\mathcal{L}_{ID}^*$ . By leveraging a powerful FR system like ArcFace, the selected morphs are more potent than the original DiM implementation with a fixed blend parameter  $w = 0.5$ .

Fast-DiM is a concurrent work which proposes an improvement on DiM with the goal of reducing the number of Network Function Evaluations (NFE) needed to produce high-quality face morphs using DiM [21]. This reduction in NFE is primarily accomplished by using alternative ODE solvers used in solving the Probability Flow ODE and replacing the “stochastic encoder” with an ODE solver for the Probability Flow ODE as time runs forward from 0 to  $T$ .

### 3. Greedy-DiM

We propose Greedy-DiM, a novel family of face morphing attacks that greedily choose the optimal solution at each timestep in solving the PF-ODE. In Table 1 we present a high-level overview of our proposed approach and how it differs from existing [20, 22] and concurrent [21] work. We evaluate the proposed morphing attacks on the open-source SYN-MAD 2022 dataset [30] with the ArcFace [26], ElasticFace [31], and AdaFace [32] FR systems via the Mated Morph Presentation Match Rate (MMPMR) metric [33] and number of Network Function Evaluations (NFE). For our heuristic function  $\mathcal{H}$  we use the identity loss outlined in Equation (8) with the ArcFace FR system. Further details on the experimental setup are provided in Section 4.

#### 3.1. Greedy Search

We begin by applying a greedy search during the sampling process of diffusion models. The information from the identity losses of Equations (7) and (8) provides a powerful heuristic function for guiding the creation of face morphs. While Zhang *et al.* [22] only uses this information to select from a pre-selected set of morphed image candidates, we propose to use this heuristic during the sampling process. Now, the identity losses require a morphed image as input; however, at time  $t > 0$ , the image  $\mathbf{x}_t$  is not fully denoised. To remedy this, we propose to use the estimate of  $\mathbf{x}_0$  at each timestep  $t$  as input to the identity losses. This can be found from the noise prediction network via

$$\mathbf{x}_\theta(\mathbf{x}_t, \mathbf{z}, t) = \frac{\mathbf{x}_t - \sigma_t \epsilon_\theta(\mathbf{x}_t, \mathbf{z}, t)}{\alpha_t} \quad (9)$$

In our greedy strategy we find the optimal blend parameter  $w$  at each step between two trajectories of the PF-ODE, the first trajectory being the model conditioned on  $\mathbf{z}_a$  and the second being conditioned on  $\mathbf{z}_b$ . This strategy is detailed in Equation (10) where  $\mathcal{H}$  is the heuristic function which measures how “optimal” the morph is relative to the bona fide images.

$$\begin{aligned} \epsilon_t^{(w)} &= \text{slerp}(\epsilon_\theta(\mathbf{x}_t^{(ab)}, \mathbf{z}_a, t), \epsilon_\theta(\mathbf{x}_t^{(ab)}, \mathbf{z}_b, t); w) \\ w'_t &= \arg \min_{w \in \{w_n\}_{n=1}^B} \mathcal{H}\left(\frac{1}{\alpha_t}(1 - \sigma_t \epsilon_t^{(w)}), \mathbf{x}_0^{(a)}, \mathbf{x}_0^{(b)}\right) \end{aligned} \quad (10)$$

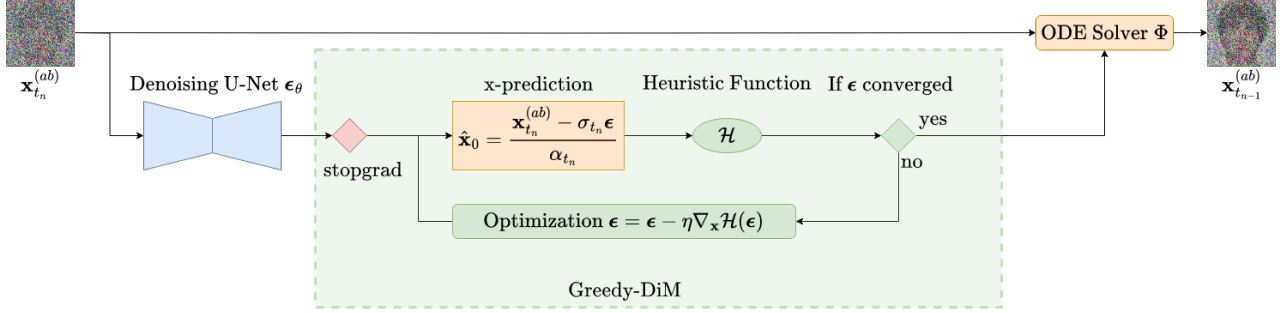


Figure 2. Overview of a single step of the Greedy-DiM\* algorithm. Proposed changes highlighted in green.

This results in a sequence of blend parameters  $w_t'$  for each step in the PF-ODE solver. Ultimately, the  $\epsilon_t^{(w_t')}$  for each timestep  $t$  is used as the noise prediction when solving the PF-ODE. We refer to this model as Greedy-DiM-S as it is the search variant of the Greedy-DiM family.

Table 2. Comparison of search strategies with the identity loss as the heuristic function.

Search Strategy	NFE( $\downarrow$ )	MMPMR( $\uparrow$ )		
		AdaFace	ArcFace	ElasticFace
None	350	92.23	90.18	93.05
Candidate	2350	<b>95.91</b>	92.84	<b>95.5</b>
Greedy	350	95.71	<b>93.87</b>	95.3

In Table 2 we compare the greedy search strategy to use no search strategy, as the case in the vanilla DiM approach in [20], and to search across several morphed candidates generated by the diffusion process, as the strategy of Morph-PIPE [22]. Note both search strategies used  $B = 21$  blends. We observe that incorporating the information from the identity loss increases the effectiveness of the morphs. Moreover, we notice a slight decrease in MMPMR when compared to searching the outputs of the diffusion process in two FR systems; however, there is an increase in MMPMR on the ArcFace FR system. In comparison with searching across the morph candidates, this approach does not increase the NFE, as the greedy search is simply over different blends of the noise prediction at each timestep  $t$ , rather than the outputs of multiple diffusion processes. *N.B.*, the noise predictions of the two trajectories can be achieved in a single NFE by batching the inputs together. See Appendix C.1 for a detailed discussion of how we report NFE for DiM models in this paper.

Table 3. Greedy search over  $\{w_n\}_{n=1}^{21} \subseteq [0, 1]$  vs greedy gradient descent over  $[0, 1]$ .

Search Space	MMPMR( $\uparrow$ )		
	AdaFace	ArcFace	ElasticFace
$\{w_n\}_{n=1}^{21} \subseteq [0, 1]$	95.71	93.87	95.3
$[0, 1]$	95.5	94.07	95.09

**Continuous Greedy Search.** Next we replace the simple search over a fixed set of blend values  $\{w_n\}_{n=1}^N$  to an optimization problem over a continuous set of blend values  $[0, 1]$ . Obviously, it is intractable to use the same search strategy from earlier on this uncountably infinite set of blend values. Rather we propose to use gradient descent to find the optimal  $w \in [0, 1]$ . In Table 3 we provide the impact this choice has on MMPMR values. We observe that the continuous search space performs slightly worse than the discrete search space on the AdaFace and ElasticFace FR systems, but slightly stronger on the ArcFace FR system. This mirrors the performance chance between the candidate and greedy search strategy shown in Table 2.

#### Algorithm 1 Greedy-DiM\* Algorithm

```

1: for  $n \leftarrow N, N-1, \dots, 2$  do
2:    $\epsilon \leftarrow \text{stopgrad}(\epsilon_\theta(\mathbf{x}_{t_n}^{(ab)}, \mathbf{z}_{ab}, t_n))$ 
3:    $\epsilon^* \leftarrow \epsilon$ 
4:    $\mathcal{H}^* \leftarrow \mathcal{H}(\hat{\mathbf{x}}_0, \mathbf{x}_0^{(a)}, \mathbf{x}_0^{(b)})$ 
5:   for  $i \leftarrow 1, 2, \dots, n_{opt}$  do
6:      $\hat{\mathbf{x}}_0 \leftarrow \frac{1}{\alpha_{t_n}}(\mathbf{x}_{t_n}^{(ab)} - \sigma_{t_n} \epsilon)$ 
7:      $\mathcal{H}(\epsilon) \leftarrow \mathcal{H}(\hat{\mathbf{x}}_0, \mathbf{x}_0^{(a)}, \mathbf{x}_0^{(b)})$ 
8:     if  $\mathcal{H}(\epsilon) < \mathcal{H}^*$  then
9:        $\mathcal{H}^* \leftarrow \mathcal{H}(\epsilon)$ 
10:       $\epsilon^* \leftarrow \epsilon$ 
11:    end if
12:     $\epsilon \leftarrow \epsilon - \eta \nabla_\epsilon \mathcal{H}(\epsilon)$ 
13:  end for
14:   $\mathbf{x}_{t_{n-1}}^{(ab)} \leftarrow \Phi(\mathbf{x}_{t_n}^{(ab)}, \epsilon^*, t_n)$ 
15: end for
16: return  $\mathbf{x}_0^{(ab)}$ 

```

### 3.2. Greedy Optimization

Rather than searching for the optimal blend  $w$  of noise predictions that minimizes the heuristic function, we propose to directly optimize the noise prediction itself. This is equivalent to optimize the step the ODE solver takes at each timestep  $t$  towards some optimal  $\mathbf{x}_0^*$  as defined by the heuristic function. As we are now directly optimizing the predicted noise, it is no longer necessary to calculate



the twin trajectories, instead the noise prediction network is only called a single time per timestep. In Algorithm 1 we outline this greedy optimization strategy which we call Greedy-DiM\* [ˈɡriːdi dɪm stɑː]. We provide a high-level overview of this algorithm in Figure 2. Algorithm 1 requires the original bona fide images  $\mathbf{x}_0^{(a)}, \mathbf{x}_0^{(b)}$ , the initial noise used in the generative process,  $\mathbf{x}_T^{(ab)}$ , the morphed conditional code  $\mathbf{z}_{ab}$ , the heuristic function  $\mathcal{H}$  which measures how “good” the morphed images is, the number of optimizer steps,  $n_{opt}$ , the learning rate  $\eta$ , PF-ODE solver  $\Phi$ , and time schedule  $\{t_n\}_{n=1}^N$ . We provide a PyTorch implementation of Algorithm 1 with batch processing in Appendix B.4.

Table 4. Comparison of Greedy-DiM-S and Greedy-DiM\*

Algorithm	NFE(↓)	MMPMR(↑)		
		AdaFace	ArcFace	ElasticFace
Greedy-DiM-S	350	95.71	93.87	95.3
Greedy-DiM*	350	<b>99.59</b>	<b>99.18</b>	<b>99.39</b>

In Table 4 we compare Greedy-DiM-S to Greedy-DiM\*. The greedy optimization strategy provides a *significant* uplift in performance over the greedy search strategy while retaining the same number of NFE. We posit that part of this is due to the enhanced size of the search space, allowing for a more optimal solution to be found.

### 3.3. Theoretical Analysis

Motivated by our strong empirical results, we present some theoretical justification for the strong performance of Greedy-DiM\* over Greedy-DiM-S and Morph-PIPE, along with some guarantees on the optimality of Greedy-DiM. A greedy algorithm is one which chooses a locally optimal solution at each timestep; however, it is not automatically guaranteed that this locally optimal solution is globally optimal. Famously, the greedy solution to the traveling salesman problem can actually yield the worst possible route [34]. Motivated by this observation we present Theorem 3.1.

**Theorem 3.1.** *Given a sequence of monotonically descending timesteps,  $\{t_n\}_{n=1}^N$ , from  $T$  to 0, the DDIM solver to the Probability Flow ODE, and a heuristic function  $\mathcal{H}$ , the locally optimal solution admitted by Greedy-DiM\* at time  $t_n$  is globally optimal.*

*Proof.* The proof is rather straightforward and based on the optimal substructure of optimizing the PF-ODE. We provide the full proof in Appendix A.1.  $\square$

Theorem 3.1 shows that Greedy-DiM\* does not just make locally optimal decisions, but globally optimal decisions. Intuitively, as the noise prediction model can be restructured into an image prediction model, see Equation (9),

if the prediction is optimal at time  $t$ , then the same prediction at time  $s < t$  is also optimal. The result from Theorem 3.1 affirms that the choice to solve the optimization problem with a greedy strategy is principled and not merely rooted in experimental success.

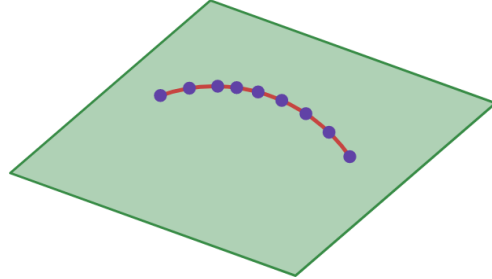


Figure 3. Illustration of the search space in  $\mathbb{R}^2$  of different DiM algorithms at a single step. Purple denotes Morph-PIPE/Greedy-DiM-S, red denotes Greedy-DiM-S continuous, and green denotes Greedy-DiM\*. Note the search spaces of the algorithms other than Greedy-DiM\* lie in a low dimensional manifold.

Next we explore the differences in the search spaces between Morph-PIPE, Greedy-DiM-S, and Greedy-DiM\*. As Morph-PIPE only explores a set of blend values on the inputs into the diffusion model, the search space is a discrete subset of  $\mathcal{X}$  of size  $B$ . The Greedy-DiM-S model has a vastly more expanded search space as there are  $B$  different choices for each timestep, resulting in a finite search space with cardinality  $B^N$ . Lastly, the search space of the Greedy-DiM\* algorithm is  $\mathcal{X}^N$ . We illustrate these differences in Figure 3. The search space of Greedy-DiM\* is significantly larger than that of other algorithms as it takes full dimension rather than lying on a low dimensional manifold. Recognizing the differences in the search spaces between the different algorithms, we consider the probability of the optimal solution being contained within the search space, from which we form Theorem 3.2.

**Theorem 3.2.** *Let  $\mathbb{P}$  be a probability distribution on a compact subset  $\mathcal{X} \subseteq \mathbb{R}^n$  with full support on  $\mathcal{X}$  which models the distribution of the optimal  $\mathbf{x}_0^*$  and is absolutely continuous w.r.t. the  $n$ -dimensional Lebesgue measure  $\lambda^n$  on  $\mathcal{X}$ . Let  $\mathcal{S}_P, \mathcal{S}_S, \mathcal{S}^*$  denote the search spaces of the Morph-PIPE, Greedy-DiM-S, and Greedy-DiM\* algorithms. Then the following statements are true.*

1.  $\mathbb{P}(\mathcal{S}_P) = \mathbb{P}(\mathcal{S}_S) = 0$ .
2.  $\mathbb{P}(\mathcal{S}^*) = 1$ .

*Proof.* The proof is based on classic results from measure theory. The full proof is provided in Appendix A.2.  $\square$

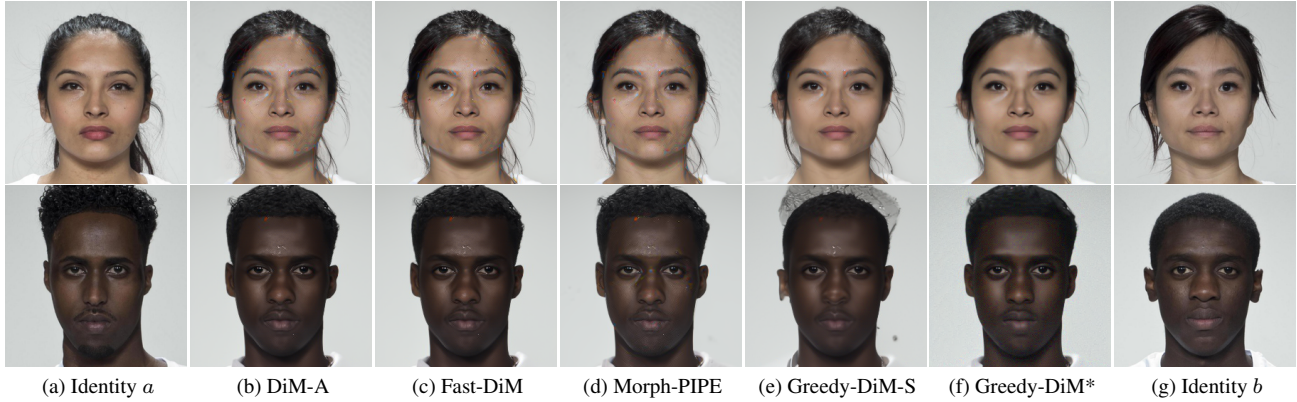


Figure 4. Comparison of DiM morphs on the FRLI dataset.

Theorem 3.2 shows that the search space of all algorithms other than Greedy-DiM\* *almost surely* do not contain the optimal solution. The results from Theorem 3.2 further justifies the superior performance exhibited by Greedy-DiM\* in relation to other search strategies. Whereas Morph-PIPE and Greedy-DiM-S search over a discrete subset of a fixed trajectory between the two bona fide images, the Greedy-DiM\* algorithm searches over the whole image space at each iteration, allowing Greedy-DiM\* to find an optimal solution that deviates from the fixed trajectory.

## 4. Experimental Setup

### 4.1. Datasets

The open-source SYN-MAD 2022 IJCB competition dataset [30] is chosen as the dataset to benchmark the proposed morphing attacks on, as it has been used as a common benchmark for measuring the performance of face morphing attacks and MAD algorithms [21, 35, 36]. The SYN-MAD 2022 dataset is derived from the Face Research Lab London (FRLI) dataset [15]. FRLI is a dataset of high-quality captures of 102 different individuals with frontal images and neutral lighting. There are two images per subject, an image of a “neutral” expression and one of a “smiling” expression. The ElasticFace [31] FR system was used to select the top 250 most similar pairs, in terms of cosine similarity, of bona fide images for both genders, resulting in a total of 500 bona fide image pairs for face morphing [30]. The SYN-MAD 2022 dataset includes morphed images with three landmark-based attacks along with MIPGAN-I and MIPGAN-II attacks. The three landmark-based attacks are the open-source OpenCV attack, the commercial-of-the-shelf (COTS) FaceMorpher, and online-tool Webmorph [30]. *N.B.*, the FaceMorpher attack in the SYN-MAD 2022 is not the same attack used in [16, 20] which was a different open-source landmark-based attack of the same name. The SYN-MAD 2022 dataset includes the OpenCV, FaceMorpher, Webmorph, MIPGAN-I, and MIPGAN-II at-

tacks. In addition to the five attacks provided by the SYN-MAD 2022, we used the same 500 pairs to generate five DiM attacks for comparison purpose. This includes the DiM algorithm using variants A and C from [20], the Morph-PIPE algorithm from [22], and the Fast-DiM and Fast-DiM-ode algorithms from concurrent work [21]. Further details can be found in Appendix C.3.

### 4.2. FR Systems

We evaluate the morphing attacks against three publicly available FR systems, namely the ArcFace [26], AdaFace [32], and ElasticFace [31] models. We chose to evaluate these three models as they represent a combination of widely used and state-of-the-art FR systems which have been used by other works to study the effectiveness of morphing attacks [20–22]. *N.B.*, the ArcFace model used in the identity loss is not the same FR system used during evaluation. Further details on the configuration of the FR systems are found in Appendix C.4.

### 4.3. Metrics

To measure the effectiveness of our proposed morphing attack, we measure the Mated Morph Presentation Match Rate (MMPMR) metric proposed by Scherhag *et al.* [33]. The MMPMR measures the number of morphed images that successfully trick an FR system into falsely matching it with each of the contributing subjects at a given verification threshold. The metric is defined as

$$M(\delta) = \frac{1}{M} \sum_{n=1}^M \left\{ \left[ \min_{n \in \{1, \dots, N_m\}} S_m^n \right] > \delta \right\} \quad (11)$$

where  $\delta$  is the verification threshold,  $S_m^n$  is the similarity score of the  $n$ -th subject of morph  $m$ ,  $N_m$  is the total number of contributing subjects to morph  $m$ , and  $M$  is the total number of morphed images. In our reporting of the MMPMR metric we set the verification threshold such that the False Match Rate (FMR) is 0.1%.

Table 5. Vulnerability of different FR systems across different morphing attacks on the SYN-MAD 2022 dataset. FMR = 0.1%.

Morphing Attack	NFE( $\downarrow$ )	MMPMR( $\uparrow$ )		
		AdaFace	ArcFace	ElasticFace
FaceMorpher [30]	-	89.78	87.73	89.57
Webmorph [30]	-	97.96	96.93	98.36
OpenCV [30]	-	94.48	92.43	94.27
MIPGAN-I [17]	-	72.19	77.51	66.46
MIPGAN-II [17]	-	70.55	72.19	65.24
DiM-A [20]	350	92.23	90.18	93.05
DiM-C [20]	350	89.57	83.23	86.3
Fast-DiM [21]	300	92.02	90.18	93.05
Fast-DiM-ode [21]	150	91.82	88.75	91.21
Morph-PIPE [22]	2350	95.91	92.84	95.5
<b>Greedy-DiM-S</b>	350	95.71	93.87	95.3
<b>Greedy-DiM*</b>	270	<b>100</b>	<b>100</b>	<b>100</b>

We also report the Morphing Attack Potential (MAP), an extension on the MMPMR metric proposed by Ferrara *et al.* [37]. The MAP metric aims to provide a more comprehensive assessment of the risk a morphing attack poses to FR systems. This is accomplished by reporting a matrix such that  $\text{MAP}[r, c]$  denotes the proportion of morphed images that successfully register a false accept against at least  $r$  attempts against each contributing subject of at least  $c$  FR systems. As the SYN-MAD 2022 dataset only has a single probe image per subject, for the other image was used in the creation of the face morph and thus not suitable to be a probe image during evaluation, we simply report  $\text{MAP}[1, c]$ . This can still provide useful insight into the generality of a morphing attack.

## 5. Results

In this section we compare our proposed Greedy-DiM algorithms to existing DiM and concurrent algorithms in addition to several GAN and landmark-based morphing attacks, studying a total of twelve morphing attacks of which ten are from other works. The specific hyperparameter configurations used in these experiments for the Greedy-DiM-S and Greedy-DiM\* algorithms are noted in Appendix B.3. Figure 4 provides some examples of morphed images generated by different DiM algorithms. We observe that all DiM algorithms other than Greedy-DiM\* exhibit strange saturation artefacts, *i.e.*, the generally bright red splotches on the morphed images, a result of clipping the dynamic range of the image to  $[-1, 1]$ . The Greedy-DiM-S algorithm exhibit some blend artefacting outside the core face region unlike the other algorithms.

### 5.1. Vulnerability Study

In Table 5 we measure the MMPMR of the twelve studied morphing attacks on all three FR systems in addition to reporting the NFE if applicable. Our results show that Greedy-DiM\* is unreasonably effective, achieving a

100% MMPMR across the board, boasting superior performance to all other morphing attacks. We observe that Greedy-DiM\* is the *first* representation-based morphing attack to *consistently* outperform landmark-based morphing attacks [17, 20–22, 30]. Remarkably, this performance is achieved with a reduction in NFE, having the lowest NFE out of all proposed DiM models with the exception of Fast-DiM-ode. As expected, Greedy-DiM-S under-performs Greedy-DiM\*, maintaining similar performance to Morph-PIPE, but with significantly lower NFE. Lastly, mirroring the observations in [21, 30] we notice the MIPGAN algorithms tend to perform very poorly and that Webmorph performs quite well.

Table 6. MAP( $\uparrow$ ) metric for all three FR systems on the SYN-MAD 2022 dataset. FMR = 0.1%.

Morphing Attack	NFE( $\downarrow$ )	Number of FR Systems		
		1	2	3
FaceMorpher [30]	-	92.23	89.57	85.28
Webmorph [30]	-	98.77	98.36	96.11
OpenCV [30]	-	97.55	93.87	89.78
MIPGAN-I [17]	-	85.07	72.39	58.69
MIPGAN-II [17]	-	80.37	69.73	57.87
DiM-A [20]	350	96.93	92.43	86.09
DiM-C [20]	350	92.84	87.53	78.73
Fast-DiM [21]	300	97.14	92.43	85.69
Fast-DiM-ode [21]	150	95.91	91.21	84.66
Morph-PIPE [22]	2350	98.16	95.71	90.39
<b>Greedy-DiM-S</b>	350	97.34	95.71	91.82
<b>Greedy-DiM*</b>	270	<b>100</b>	<b>100</b>	<b>100</b>

Similar conclusions are drawn from Table 6 which illustrates the  $\text{MAP}[1, c]$  metric for all twelve studied morphing attacks. Again, Greedy-DiM\* performs unreasonably well having a 100% chance to fool all three FR systems studied in this work. The next closest attack, is Webmorph with a 96.11% chance to fool all three FR systems, and the nearest representation-based attack is Greedy-DiM-S with a 91.82% chance to fool all three FR systems. Likewise, the MIPGAN attacks perform abysmally, with only a mere < 60% chance to fool all three FR systems.

### 5.2. Detectability Study

In this section we study the detectability of Greedy-DiM by implementing a Single-image MAD (S-MAD) detector trained on various morphing attacks. The detectability study follows a similar approach to that of [20, 21]. We employ stratified  $k$ -fold cross validation when performing the study to ensure a consistent balance of morphed and bona fide images in each fold. Further details on the S-MAD detector and other configuration details for this study can be found in Appendix C.5. The S-MAD model is trained in three different scenarios for a potential S-MAD algorithm. In the first scenario we train the S-MAD detector on OpenCV morphs, the second MIPGAN-II morphs, and

Table 7. Detection Study on all training subsets of the SYN-MAD 2022 dataset

Morphing Attack	OpenCV				MIPGAN-II				DiM-A			
	EER(↑)	MACER @ BPCER(↑)			EER(↑)	MACER @ BPCER(↑)			EER(↑)	MACER @ BPCER(↑)		
		0.1%	1.0%	5.0%		0.1%	1.0%	5.0%		0.1%	1.0%	5.0%
FaceMorpher [30]	31.86	99.02	97.06	80.39	<b>82.84</b>	<b>100</b>	<b>100</b>	<b>99.51</b>	44.61	99.02	97.06	87.75
Webmorph [30]	0.49	0.98	0.49	0.49	51.47	<b>100</b>	99.51	93.14	36.27	98.04	90.69	74.02
OpenCV [30]	0.98	2.45	0.49	0.49	50.49	99.51	95.1	90.69	39.71	99.02	85.29	73.04
MIPGAN-I [17]	24.02	83.82	70.1	49.02	8.33	38.24	24.51	14.22	44.61	<b>99.51</b>	95.1	85.78
MIPGAN-II [17]	26.47	73.53	59.8	45.59	3.92	34.31	8.82	2.94	<b>52.94</b>	99.02	<b>98.04</b>	<b>90.69</b>
DiM-A [20]	<b>82.84</b>	<b>100</b>	<b>100</b>	<b>100</b>	54.9	99.51	99.51	93.63	0.98	2.94	0.98	0
DiM-C [20]	63.73	99.51	99.02	98.53	43.63	99.51	93.63	85.29	5.39	42.65	19.61	6.37
Fast-DiM [21]	75	<b>100</b>	<b>100</b>	<b>100</b>	50	99.51	99.51	94.61	2.94	12.25	5.88	1.96
Fast-DiM-ode [21]	75.49	<b>100</b>	<b>100</b>	<b>100</b>	56.37	99.51	99.51	98.53	1.96	7.35	2.94	0.49
Morph-PIPE [22]	82.35	<b>100</b>	<b>100</b>	<b>100</b>	54.9	99.51	99.51	93.63	0.98	5.39	0.98	0
<b>Greedy-DiM-S</b>	52.94	<b>100</b>	<b>100</b>	98.53	36.76	<b>100</b>	92.65	79.41	6.37	63.73	14.22	6.86
<b>Greedy-DiM*</b>	37.75	99.02	93.63	85.29	34.31	96.57	93.14	75	17.16	85.78	56.37	42.65

the third DiM-A morphs.

In order to assess the detectability of morphing attacks by the S-MAD algorithm, we measure the Equal Error Rate (EER), Morphing Attack Presentation Error Rate (MACER), and Bona fide Presentation Classification Error Rate (BPCER). The results of our detectability are shown in Table 7. In general, we notice that in order for the S-MAD algorithm to be able to *consistently* detect morphing attacks it needs to be trained on that general type of morphing attack, *i.e.*, trained on an attack which uses a similar generation process in creating the morphed images. When the S-MAD detector is trained on OpenCV morphs the Greedy-DiM attacks are the most likely to be detected out of all the DiM attacks; however, the MIPGAN and other landmark-based attacks are more easily detected. We notice that the detector trained on MIPGAN-II has a hard time detecting the attack it was trained on, much less detecting the other non-MIPGAN attacks. Interestingly, we observe that for the S-MAD detector trained on DiM-A that the Greedy-DiM attacks are the hardest amongst the DiM attacks to detect with Morph-PIPE being the easiest, outside of the trivial case of DiM-A. This highlights the substantial difference between Greedy-DiM, and Greedy-DiM\* in particular, and all other DiM attacks.

## 6. Conclusions

We propose Greedy-DiM, a novel family of face morphing algorithm that use greedy strategies to enhance the optimization of the PF-ODE solver. Greedy-DiM\* achieves unreasonably effective performance resulting in a 100% MMPMR on all tested FR systems, far surpassing previous efforts. Moreover, this is accomplished with minimal overhead. Additional theoretical analysis corroborates the strong experimental results. While this work has focused primarily on face morphing, the proposed greedy strategies can be used for a variety of heuristic functions and tasks, enabling heuristic guided diffusion for other applications.

In this work we only evaluated our morphs on the SYN-MAD 2022 dataset and used only a single diffusion backbone rather than exploring multiple diffusion architectures. While not tested in this work, it should be a rather straightforward extension to apply these techniques to Latent Diffusion Models. Moreover, modifications to the heuristic function could be explored such as encouraging the morphed image to be less “detectable”. Additionally, the scheduling of the greedy procedure was left as a linear schedule. Such extensions are left for future work.

**Reproducibility Statement.** To ensure the reproducibility and completeness of this work we include the Appendix with five main sections. In Appendix A we provide the complete proofs for all theorems in this paper. In Appendix B we provide the implementation details for the proposed algorithms. Likewise, Appendix C provides further detail on how we performed our evaluations of the proposed algorithms. In Appendix D we provide additional results that were outside the scope of the main paper that the reader may find interesting. Lastly, in Appendix E we provide additional samples created by the proposed algorithms.

**Ethics Statement.** Face morphs can be used for a variety of malicious purposes from creating misleading content to attacking FR systems. Our advances in heuristic guided generation of diffusion models can be harnessed to produce deceptive face morphs or “deepfakes” among other attacks. We hope our colleagues will incorporate this work into future research on MAD algorithms to mitigate the large threat these morphs pose to unprotected FR systems.

## Acknowledgements

This material is based upon work supported by the Center for Identification Technology Research and National Science Foundation under Grant #1650503.



## References

- [1] L. J. Spreeuwiers, A. J. Hendrikse, and K. J. Gerritsen, "Evaluation of automatic face recognition for automatic border control on actual data recorded of travellers at schiphol airport," in *Proc. of the Int'l Conf. of Biometrics Special Interest Group (BIOSIG)*, 2012, pp. 1–6. [1](#)
- [2] M. Ngan, P. Grother, K. Hanaoka, and J. Kuo, "Face recognition vendor test (frvt) part 4: Morph - performance of automated face morph detection," 2020-03-06 2020. [1](#)
- [3] M. Ferrara, A. Franco, and D. Maltoni, *On the Effects of Image Alterations on Face Recognition Accuracy*. Cham: Springer International Publishing, 2016, pp. 195–222. [Online]. Available: [https://doi.org/10.1007/978-3-319-28501-6\\_9](https://doi.org/10.1007/978-3-319-28501-6_9) [1](#)
- [4] D. J. Robertson, R. S. S. Kramer, and A. M. Burton, "Fraudulent id using face morphs: Experiments on human and automatic recognition," *PLOS ONE*, vol. 12, no. 3, pp. 1–12, 03 2017. [Online]. Available: <https://doi.org/10.1371/journal.pone.0173319> [1](#)
- [5] R. Raghavendra, K. B. Raja, and C. Busch, "Detecting morphed face images," in *IEEE 8th Int'l Conf. on Biometrics Theory, Applications and Systems (BTAS)*, 2016, pp. 1–7. [1](#)
- [6] L. Colbois and S. Marcel, "On the detection of morphing attacks generated by gans," in *2022 International Conference of the Biometrics Special Interest Group (BIOSIG)*, 2022, pp. 1–5. [1](#)
- [7] Z. Blasingame and C. Liu, "Leveraging adversarial learning for the detection of morphing attacks," *2021 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–8, 2021. [1](#)
- [8] S. Venkatesh, R. Ramachandra, K. Raja, and C. Busch, "Face morphing attack generation and detection: A comprehensive survey," *IEEE Transactions on Technology and Society*, vol. 2, no. 3, pp. 128–145, 2021. [1](#)
- [9] M. Ferrara, A. Franco, and D. Maltoni, "The magic passport," *IEEE International Joint Conference on Biometrics*, pp. 1–7, 2014. [1](#)
- [10] K. Raja et al., "Morphing attack detection - database, evaluation platform and benchmarking," *IEEE Transactions on Information Forensics and Security*, pp. 1–1, 2020. [1](#)
- [11] R. Raghavendra, K. B. Raja, S. Venkatesh, and C. Busch, "Transferable deep-cnn features for detecting digital and print-scanned morphed face images," in *IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1822–1830. [1](#)
- [12] U. Scherhag, C. Rathgeb, and C. Busch, "Morph detection from single face image: A multi-algorithm fusion approach," in *Proceedings of the 2018 2nd International Conference on Biometric Engineering and Applications*, ser. ICBEA '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 6–12. [Online]. Available: <https://doi.org/10.1145/3230820.3230822> [1](#)
- [13] K. O'Haire, S. Soleymani, B. Chaudhary, P. Aghdaie, J. Dawson, and N. M. Nasrabadi, "Adversarially perturbed wavelet-based morphed face generation," in *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, 2021, pp. 01–05. [1](#)
- [14] S. Venkatesh, R. Ramachandra, K. Raja, L. Spreeuwiers, R. Veldhuis, and C. Busch, "Detecting morphed face attacks using residual noise from deep multi-scale context aggregation network," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020. [1](#)
- [15] L. DeBruine and B. Jones, "Face Research Lab London Set," 5 2017. [Online]. Available: [https://figshare.com/articles/dataset/Face\\_Research\\_Lab\\_London\\_Set/5047666](https://figshare.com/articles/dataset/Face_Research_Lab_London_Set/5047666) [1](#), [6](#)
- [16] E. Sarkar, P. Korshunov, L. Colbois, and S. Marcel, "Are gan-based morphs threatening face recognition?" in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 2959–2963. [1](#), [2](#), [6](#), [17](#)
- [17] H. Zhang, S. Venkatesh, R. Ramachandra, K. Raja, N. Damer, and C. Busch, "Mipgan—generating strong and high quality morphing attacks using identity prior driven gan," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 3, no. 3, pp. 365–383, 2021. [1](#), [2](#), [7](#), [8](#), [17](#)
- [18] S. Venkatesh, H. Zhang, R. Ramachandra, K. Raja, N. Damer, and C. Busch, "Can gan generated morphs threaten face recognition systems equally as landmark based morphs? - vulnerability and detection," in *2020 8th International Workshop on Biometrics and Forensics (IWBF)*, 2020, pp. 1–6. [1](#)
- [19] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 8780–8794. [Online]. Available: <https://proceedings.neurips.cc/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf> [1](#)
- [20] Z. W. Blasingame and C. Liu, "Leveraging diffusion for strong and high quality face morphing attacks," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 6, no. 1, pp. 118–131, 2024. [1](#), [2](#), [3](#), [4](#), [6](#), [7](#), [8](#), [14](#), [15](#), [17](#), [18](#), [20](#)
- [21] —, "Fast-dim: Towards fast diffusion morphs," *IEEE Security & Privacy*, pp. 2–13, 2024. [1](#), [2](#), [3](#), [6](#), [7](#), [8](#), [17](#), [18](#), [19](#), [20](#)
- [22] H. Zhang, R. Ramachandra, K. Raja, and B. Christoph, "Morph-pipe: Plugging in identity prior to enhance face morphing attack based on diffusion model," in *Norwegian Information Security Conference (NISK)*, 2023. [1](#), [2](#), [3](#), [4](#), [6](#), [7](#), [8](#), [15](#), [17](#), [20](#)
- [23] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwanajakorn, "Diffusion autoencoders: Toward a meaningful and decodable representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 619–10 629. [1](#), [3](#), [15](#), [19](#)

- [24] M. Ferrara, A. Franco, and D. Maltoni, “Decoupling texture blending and shape warping in face morphing,” in *2019 International Conference of the Biometrics Special Interest Group (BIOSIG)*, 2019, pp. 1–5. [2](#)
- [25] N. Damer, A. M. Saladié, A. Braun, and A. Kuijper, “Morgan: Recognition vulnerability and attack detectability of face morphing attacks created by generative adversarial network,” in *2018 IEEE 9th Int’l Conf. on Biometrics Theory, Applications and Systems (BTAS)*, 2018, pp. 1–10. [2](#)
- [26] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699. [2](#), [3](#), [6](#), [17](#)
- [27] J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=St1giarCHLP> [2](#)
- [28] T. Karras, M. Aittala, T. Aila, and S. Laine, “Elucidating the design space of diffusion-based generative models,” in *Proc. NeurIPS*, 2022. [2](#)
- [29] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=PXTIG12RRHS> [2](#)
- [30] M. Huber, F. Boutros, A. T. Luu, K. Raja, R. Ramachandra, N. Damer, P. C. Neto, T. Gonçalves, A. F. Sequeira, J. S. Cardoso, J. Tremoço, M. Lourenço, S. Serra, E. Cermeño, M. Ivanovska, B. Batagelj, A. Kronovšek, P. Peer, and V. Štruc, “Syn-mad 2022: Competition on face morphing attack detection based on privacy-aware synthetic training data,” in *2022 IEEE International Joint Conference on Biometrics (IJCB)*, 2022, pp. 1–10. [3](#), [6](#), [7](#), [8](#)
- [31] F. Boutros, N. Damer, F. Kirchbuchner, and A. Kuijper, “Elasticface: Elastic margin loss for deep face recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 1578–1587. [3](#), [6](#), [17](#)
- [32] M. Kim, A. K. Jain, and X. Liu, “Adaface: Quality adaptive margin for face recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. [3](#), [6](#), [17](#)
- [33] U. Scherhag, A. Nautsch, C. Rathgeb, M. Gomez-Barrero, R. N. J. Veldhuis, L. Spreeuwiers, M. Schils, D. Maltoni, P. Grother, S. Marcel, R. Breithaupt, R. Ramachandra, and C. Busch, “Biometric systems under morphing attacks: Assessment of morphing techniques and vulnerability reporting,” in *2017 International Conference of the Biometrics Special Interest Group (BIOSIG)*, 2017, pp. 1–7. [3](#), [6](#)
- [34] G. Gutin, A. Yeo, and A. Zverovich, “Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp,” *Discrete Applied Mathematics*, vol. 117, no. 1, pp. 81–86, 2002. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0166218X01001950> [5](#)
- [35] M. Long, Q. Yao, L.-B. Zhang, and F. Peng, “Face demorphing based on diffusion autoencoders,” *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3051–3063, 2024. [6](#)
- [36] H. Rachalwar, M. Fang, N. Damer, and A. Das, “Depth-guided robust face morphing attack detection,” in *2023 IEEE International Joint Conference on Biometrics (IJCB)*, 2023, pp. 1–9. [6](#)
- [37] M. Ferrara, A. Franco, D. Maltoni, and C. Busch, “Morphing attack potential,” in *2022 International Workshop on Biometrics and Forensics (IWF)*, 2022, pp. 1–6. [7](#)
- [38] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, “Consistency models,” *arXiv preprint arXiv:2303.01469*, 2023. [15](#)
- [39] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, “On the variance of the adaptive learning rate and beyond,” in *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, April 2020. [15](#), [20](#)
- [40] Y. Gu, X. Wang, Y. Ge, Y. Shan, X. Qie, and M. Z. Shou, “Rethinking the Objectives of Vector-Quantized Tokenizers for Image Synthesis,” p. arXiv:2212.03185, Dec. 2022. [15](#), [20](#)
- [41] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4396–4405. [17](#)
- [42] I. C. Duta, L. Liu, F. Zhu, and L. Shao, “Improved residual networks for image and video recognition,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 9415–9422. [17](#)
- [43] X. An, X. Zhu, Y. Gao, Y. Xiao, Y. Zhao, Z. Feng, L. Wu, B. Qin, M. Zhang, D. Zhang, and Y. Fu, “Partial fc: Training 10 million identities on a single machine,” in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 1445–1449. [17](#)
- [44] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2017. [17](#)
- [45] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141. [17](#)
- [46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. [17](#)
- [47] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, “Alias-free generative adversarial networks,” in *Proc. NeurIPS*, 2021. [18](#)
- [48] D. Busbridge, J. Ramapuram, P. Ablin, T. Likhomanenko, E. Gunesh Dhekane, X. Suau, and R. Webb, “How to Scale Your EMA,” *arXiv e-prints*, p. arXiv:2307.13813, Jul. 2023. [18](#)

- [49] U. M. Kelly, M. Nauta, L. Liu, L. J. Spreeuwers, and R. N. J. Veldhuis, “Worst-Case Morphs using Wasserstein ALI and Improved MIPGAN,” *arXiv e-prints*, p. arXiv:2310.08371, Oct. 2023. [19](#)
- [50] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 586–595. [19](#)
- [51] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *Proceedings of the third International Conference on Learning Representations (ICLR 2015)*, 2015. [19](#)
- [52] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, “Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models,” 2023. [19](#)
- [53] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the third International Conference on Learning Representations (ICLR 2015)*, 2015. [20](#)
- [54] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, “Universal adversarial perturbations,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 86–94. [22](#)

# Contents

<b>1. Introduction</b>	<b>1</b>
<b>2. Prior Work</b>	<b>2</b>
2.1. DiM	2
<b>3. Greedy-DiM</b>	<b>3</b>
3.1. Greedy Search	3
3.2. Greedy Optimization	4
3.3. Theoretical Analysis	5
<b>4. Experimental Setup</b>	<b>6</b>
4.1. Datasets	6
4.2. FR Systems	6
4.3. Metrics	6
<b>5. Results</b>	<b>7</b>
5.1. Vulnerability Study	7
5.2. Detectability Study	7
<b>6. Conclusions</b>	<b>8</b>
<b>A Proofs</b>	<b>13</b>
A.1. Greedy-DiM is Globally Optimal	13
A.2. Probability of Finding the Optimal Solution	13
<b>B Implementation Details</b>	<b>14</b>
B.1. DiM Algorithm	14
B.2. Repositories Used	14
B.3. Hyperparameters	15
B.4. PyTorch Implementation of Greedy Optimization	16
<b>C Evaluation Details</b>	<b>16</b>
C.1. NFE	16
C.2. Hardware	17
C.3. Datasets	17
C.4. FR Systems	17
C.5. Detectability Study	17
<b>D Additional Results</b>	<b>18</b>
D.1. Relative Strength Metric	18
D.2. Comparison of all Design Choices	18
D.3. Empirical Time Analysis	20
<b>E Additional Images</b>	<b>22</b>



## A. Proofs

### A.1. Greedy-DiM is Globally Optimal

**Theorem A.1.** *Given a sequence of monotonically descending timesteps,  $\{t_n\}_{n=1}^N$ , from  $T$  to 0 and the DDIM solver to the Probability Flow ODE in the Variance-Preserving formulation and a heuristic function  $\mathcal{H}$ , the locally optimal solution  $\epsilon_{t_n}$  which minimizes  $\mathcal{H}$  at timestep  $t_n$  is globally optimal.*

*Proof.* Let  $\mathbf{x}'_0$  minimize  $\mathcal{H}$ . Consider an arbitrary timestep  $t$  with a locally optimal  $\epsilon'_t$  which is related to  $\mathbf{x}'_0$  by

$$\epsilon'_t = \frac{\mathbf{x}_t - \alpha_t \mathbf{x}'_0}{\sigma_t} \quad (12)$$

i.e.,  $\epsilon_t$  minimizes  $\mathcal{H}$  at time  $t$ . We will show that for any  $s < t$ ,  $\epsilon_t = \epsilon_s$ . Let  $s$  be the next timestep in the sequence. The DDIM update equation shows that  $\mathbf{x}_s$  is defined as

$$\mathbf{x}_s = \frac{\alpha_s}{\alpha_t} (\mathbf{x}_t - \sigma_t \epsilon'_t) + \sigma_s \epsilon'_t \quad (13)$$

Plugging Equation (12) into Equation (13) we have

$$\mathbf{x}_s = \alpha_s \mathbf{x}'_0 + \frac{\sigma_s}{\sigma_t} (\mathbf{x}_t - \alpha_t \mathbf{x}'_0) \quad (14)$$

Now we write Equation (12) at time  $s$  in terms of Equation (14) which yields

$$\epsilon'_s = \frac{\alpha_s \mathbf{x}'_0 + \frac{\sigma_s}{\sigma_t} (\mathbf{x}_t - \alpha_t \mathbf{x}'_0) - \alpha_s \mathbf{x}'_0}{\sigma_s} \quad (15)$$

Simplifying, Equation (15) we find

$$\epsilon'_s = \frac{\mathbf{x}_t - \alpha_t \mathbf{x}'_0}{\sigma_t} = \epsilon'_t \quad (16)$$

Therefore for any timestep  $s < t$  the locally optimal solution  $\epsilon'_t$  is locally optimal at time  $s$ . Thus starting at time  $T$  with locally optimal solution  $\epsilon'_T$  the locally optimal solutions for all  $t \in [0, T]$  are  $\epsilon'_T$ . Therefore, for any timestep  $t_n$  the locally optimal solution  $\epsilon_{t_n}$  is globally optimal.  $\square$

### A.2. Probability of Finding the Optimal Solution

**Theorem A.2.** *Let  $\mathbb{P}$  be a probability distribution on a compact subset  $\mathcal{X} \subseteq \mathbb{R}^n$  with full support on  $\mathcal{X}$  which models the distribution of the optimal  $\mathbf{x}_0^*$  and is absolutely continuous w.r.t. the  $n$ -dimensional Lebesgue measure  $\lambda^n$  on  $\mathcal{X}$ . Let  $\mathcal{S}_P, \mathcal{S}_S, \mathcal{S}^*$  denote the search spaces of the Morph-PIPE, Greedy-DiM-S, and Greedy-DiM\* algorithms. Then the following statements are true.*

1.  $\mathbb{P}(\mathcal{S}_P) = \mathbb{P}(\mathcal{S}_S) = 0$ .
2.  $\mathbb{P}(\mathcal{S}^*) = 1$ .

*Proof.* Let  $\mathbf{x}_T^{(a)}, \mathbf{x}_T^{(b)}$  denote the initial noise of the bona fide images and let  $\mathbf{z}_a, \mathbf{z}_b$  denote the conditional representations of the bona fide images. Remark a measure  $\mu$  is said to be absolutely continuous w.r.t. to  $\nu$  if and only if for all measurable sets  $A$ ,  $\nu(A) = 0 \implies \mu(A) = 0$ . We consider the search space of the Morph-PIPE algorithm  $\mathcal{S}_P$ . We can construct  $\mathcal{S}_P$  as

$$\mathcal{S}_P = \left\{ \Phi^N(\text{slerp}(\mathbf{x}_T^{(a)}, \mathbf{x}_T^{(b)}; w), \text{lerp}(\mathbf{z}_a, \mathbf{z}_b; w)) \mid w \in \{w_n\}_{n=1}^B \right\} \quad (17)$$

where  $\Phi^N(\mathbf{x}_T, \mathbf{z})$  denotes the output of the diffusion model with  $N$  sampling steps and  $\{w_n\}_{n=1}^B \subseteq [0, 1]$  is set of blend values. By construction it follows that  $|\mathcal{S}_P| = B$ , i.e., the  $B$  candidates generated by the Morph-PIPE algorithm. Clearly, search space is a null set w.r.t. to the Lebesgue measure on  $\mathcal{X}$  and therefore  $\lambda^n(\mathcal{S}_P) = 0$  which by the definition of absolute continuity implies  $\mathbb{P}(\mathcal{S}_P) = 0$ .

Next we consider the search space of the Greedy-DiM-S algorithm  $\mathcal{S}_S$ . Clearly the search space at a timestep  $t$ , is a set of a size of the  $B$  blend values explored. Over the  $N$  sampling steps of the diffusion model the greedy algorithm can explore a

maximum of  $B^N$  models. It follows the total search space  $\mathcal{S}_S$  is still discrete and thus the Lebesgue measure of the search space is zero which implies that  $\mathbb{P}(\mathcal{S}_S) = 0$ . We have now shown that Statement 1 is true.

Lastly, we consider the search space of the Greedy-DiM\* algorithm  $\mathcal{S}^*$ . By definition Greedy-DiM\* performs gradient descent on  $\mathcal{X}$  at each timestep and therefore the search space of the whole algorithm is  $\mathcal{S}^* = \mathcal{X}$ . By definition  $\mathbb{P}(\mathcal{X}) = \mathbb{P}(\mathcal{S}^*) = 1$  as  $\mathbb{P}$  takes full support on  $\mathcal{X}$ . We have now shown that Statement 2 is true finishing the proof.  $\square$

Note it can be shown that search space of Greedy-DiM-S continuous lies on a low-dimensional manifold of  $\mathcal{X}$  and therefore takes probability 0, but the proof is more technical so we opt to omit it as it is not important to our analysis.

## B. Implementation Details

### B.1. DiM Algorithm

For completeness we provide the DiM algorithm from [20] in our own notation in Algorithm 2. The original bona fide images are denoted  $\mathbf{x}_0^{(a)}$  and  $\mathbf{x}_0^{(b)}$ . The conditional encoder is  $\mathcal{E} : \mathcal{X} \rightarrow \mathcal{Z}$ ,  $\Phi$  is the numerical PF-ODE solver,  $\Phi^+$  is the numerical ODE solver of the PF-ODE as time runs forwards from 0 to  $T$ .

---

**Algorithm 2** DiM Framework.

---

**Require:** Blend parameter  $w = 0.5$ . Time schedule  $\{t_i\}_{i=1}^N \subseteq [0, T], t_i < t_{i+1}$ .

```

1:  $\mathbf{z}_a \leftarrow \mathcal{E}(\mathbf{x}_0^{(a)})$  ▷ Encoding bona fides into conditionals.
2:  $\mathbf{z}_b \leftarrow \mathcal{E}(\mathbf{x}_0^{(b)})$ 
3: for  $i \leftarrow 1, 2, \dots, N - 1$  do
4:    $\mathbf{x}_{t_{i+1}}^{(a)} \leftarrow \Phi^+(\mathbf{x}_{t_i}^{(a)}, \epsilon_\theta(\mathbf{x}_{t_i}^{(a)}, \mathbf{z}_a, t_i), t_i)$  ▷ Solving the PF-ODE as time runs from 0 to  $T$ .
5:    $\mathbf{x}_{t_{i+1}}^{(b)} \leftarrow \Phi^+(\mathbf{x}_{t_i}^{(b)}, \epsilon_\theta(\mathbf{x}_{t_i}^{(b)}, \mathbf{z}_b, t_i), t_i)$ 
6: end for
7:  $\mathbf{x}_T^{(ab)} \leftarrow \text{slerp}(\mathbf{x}_T^{(a)}, \mathbf{x}_T^{(b)}; w)$  ▷ Morph initial noise.
8:  $\mathbf{z}_{ab} \leftarrow \text{lerp}(\mathbf{z}_a, \mathbf{z}_b; w)$  ▷ Morph conditionals.
9: for  $i \leftarrow N, N - 1, \dots, 2$  do
10:   $\mathbf{x}_{t_{i-1}}^{(ab)} \leftarrow \Phi(\mathbf{x}_{t_i}^{(ab)}, \epsilon_\theta(\mathbf{x}_{t_i}^{(ab)}, \mathbf{z}_{ab}, t_i), t_i)$  ▷ Solving the PF-ODE as time runs from  $T$  to 0.
11: end for
12: return  $\mathbf{x}_0^{(ab)}$ 

```

---

Note for a vector space  $V$  and two vectors  $u, v \in V$ , the spherical interpolation by a factor of  $\gamma$  is given as

$$\text{slerp}(u, v; \gamma) = \frac{\sin((1 - \gamma)\theta)}{\sin \theta} u + \frac{\sin(\gamma\theta)}{\sin \theta} v \quad (18)$$

where

$$\theta = \frac{\arccos(u \cdot v)}{\|u\| \|v\|} \quad (19)$$

### B.2. Repositories Used

For reproducibility purposes we provide a list of links to the official repositories of other works used in this paper.

1. The SYN-MAD 2022 dataset used in this paper can be found at <https://github.com/marcohuber/SYN-MAD-2022>.
2. The implementation for OpenCV morphs can be found at <https://learnopencv.com/face-morph-using-opencv-cpp-python>.
3. The COTS FaceMorpher tool can be found at <https://www.luxand.com/facemorpher>.
4. The Webmorph online tool can be found at <https://webmorph.org>.
5. The ArcFace models, MS1M-RetinaFace dataset, and MS1M-ArcFace dataset can be found at <https://github.com/deepinsight/insightface>.

6. The ElasticFace model can be found at <https://github.com/fdbtrs/ElasticFace>.
7. The AdaFace model can be found at <https://github.com/mk-minchul/AdaFace>.
8. The official Diffusion Autoencoders repository can be found at <https://github.com/phizaz/diffae>.
9. The official MIPGAN repository can be found at <https://github.com/ZHYYYYYYYYYYYYY/MIPGAN-face-morphing-algorithm>.
10. The SE-ResNeXt101-32x4d can be found at [https://catalog.ngc.nvidia.com/orgs/nvidia/resources/se\\_resnext\\_for\\_pytorch](https://catalog.ngc.nvidia.com/orgs/nvidia/resources/se_resnext_for_pytorch).

### B.3. Hyperparameters

In Table 8 we enumerate the hyperparameters used in the main paper for the Greedy-DiM algorithms and provide some justification for our choices. For Greedy-DiM-S we used the recommended  $N = 100$  steps for the DDIM solver from [20,23]. In our experiments we found  $N = 20$  to perform just as well as  $N = 100$  for Greedy-DiM\*, see Table 12. We use the recommend  $N_F = 250$  steps with the DiffAE forward ODE solver [20,23]. For the Greedy-DiM-S algorithm we use  $B = 21$  as that is what was used in the Morph-PIPE piper [22]. Additionally, we use slerp as our interpolation function for the predicted noise which is a common choice when morphing in the image domain [20,23]. At the suggestion of Song *et al.* [38] we use the Rectified Adam optimizer [39], with no learning rate decay or warm-up. In we performed an initial parameter study and found a learning rate of 0.01 to work well and chose to use that for the remainder of our experiments. We performed a small parameter search on the momentum parameters of the optimizer and found the values  $\beta_0 = 0.5$  and  $\beta_1 = 0.9$  recommended by Gu *et al.* [40] to work quite well, see Table 13. We varied the optimizer stride, *i.e.*, how many steps to skip the greedy search on, and found it to negatively impact results so we opted to leave it 1, thereby disabling optimizer striding. In our initial experiments we found  $n_{opt} = 50$  to work quite well. We performed an extensive study on the heuristic function and found the ArcFace identity loss to work the best. See Table 9 for the full results of this study.

Table 8. Hyperparamters used for the main paper.

Hyperparameter	Greedy-DiM-S	Greedy-DiM*
<b>ODE Solvers</b>		
$N$	100	20
ODE Solver	DDIM	DDIM
$N_F$	250	250
Forward ODE Solver	DiffAE	DiffAE
<b>Search</b>		
Number of blends	21	-
Interpolation function	slerp	-
<b>Optimization</b>		
Optimizer	-	RAdam
Learning rate	-	0.01
$\beta_0$	-	0.5
$\beta_1$	-	0.9
Optimizer stride	-	1
$n_{opt}$	-	50
Heuristic function	ArcFace Identity Loss	ArcFace Identity Loss

## B.4. PyTorch Implementation of Greedy Optimization

We provide a minimalist example of the greedy optimization procedure for diffusion models in PyTorch. The code expects noise prediction network  $\text{model}(x_t, z, t)$  that takes the noisy input image  $x_t$ , conditional  $z$ , and timestep  $t$  and a scheduler, like the DPM-Solver as seen in [https://huggingface.co/docs/diffusers/api/schedulers/multistep\\_dpm\\_solver](https://huggingface.co/docs/diffusers/api/schedulers/multistep_dpm_solver). Additionally, the code allows for an optimal parameter `noise_level` if the starting sample timestep  $t_N < T$  and optimizer striding with `opt_stride`.

---

**Algorithm 3** Pytorch Code for Greedy Optimization w.r.t. to a Heuristic function.

---

```
def greedy_optimization(model, scheduler, xt, z, loss_fn, n_opt_steps=50, opt_stride=1,
→ noise_level=1.0, opt_kwargs={}, device=None):
    device = device if device is not None else model.device

    timesteps = scheduler.timesteps

    if noise_level < 1.0:
        timesteps = timesteps[int((1. - noise_level) * len(timesteps)):]

    for i, t in enumerate(timesteps):
        with torch.no_grad():
            out = model(xt, z, t)

            if (i % opt_stride) == 0:
                out = out.detach().clone().requires_grad_(True)
                opt = torch.optim.RAdam([out], **opt_kwargs)

                x0_pred = convert_eps_to_x0(out, t, xt.detach())
                best_loss = loss_fn(x0_pred)
                best_out = out

            for _ in range(n_opt_steps):
                opt.zero_grad()

                x0_pred = convert_eps_to_x0(out, t, xt.detach())
                loss = loss_fn(x0_pred)

                loss.mean().backward()
                opt.step()

                do_update = (loss < best_loss).float()
                best_loss = do_update * loss + (1. - do_update) * best_loss
                best_out = do_update[:, None, None, None] * out \
                    + (1. - do_update)[:, None, None, None] * best_out

            out = best_out

    xt = scheduler.step(out, t, xt)

    return xt
```

---

## C. Evaluation Details

### C.1. NFE

In our reporting of the NFE we record the number of times the diffusion noise prediction U-Net is evaluated both during the encoding phase,  $N_E$ , and solving of the PF-ODE,  $N$ . We chose to report  $N + N_E$  over  $N + 2N_E$  as even though two bona fide images are encoded resulting in  $2N_E$  NFE during encoding, this process can simply be batched together, reducing the NFE down to  $N_E$ . When reporting the NFE for the Morph-PIPE model, we report  $N_E + BN$  where  $B$  is the number of blends. While a similar argument can be made that the morphed candidates could be generated in a large batch of size  $B$ , reducing the NFE of the sampling process down to  $N$ , we chose to report  $BN$  as the number of blends,  $B = 21$ , used in the



Morph-PIPE is quite large, potentially resulting in Out Of Memory (OOM) errors, especially if trying to process a mini-batch of morphs. Using  $N_E + N$  reporting over  $N_E + BN$ , the NFE of Morph-PIPE is 350, which is comparable to DiM. The NFE for the Greedy-DiM-S algorithm is reported as  $N_E + N$  rather than  $N_E + 2N$ . The justification being that even though we calculate twin trajectories while solving the PF-ODE, this operation can simply be batched, as it only doubles the number of samples.

## C.2. Hardware

All experiments were done on a single NVIDIA V100 32GB GPU.

## C.3. Datasets

In the SYN-MAD 2022 dataset the OpenCV morphs only consist of 489 morphed images rather than the full 500 due to technical issues with the remaining 11 morphs. Due to this we elect to only use the 489 pairs of bona fide images for our experiments to ensure a fair comparison across all morphing attacks. All the bona fide images from the FRLI dataset are aligned and cropped using the dlib library based on the pre-processing used to align and crop the FFHQ dataset [41]. The landmark-based morphs are also aligned in a post-processing step using this strategy. Since the MIPGAN- and DiM-based morphs already use the alignment script in pre-processing the bona fide images for face morphing, it is unnecessary to run it a second time for these morphing attacks. All experiments on DiM models used a pre-trained Diffusion Autoencoder trained on the FFHQ dataset as was used in the original papers [20–22].

## C.4. FR Systems

All three FR systems use the Improved ResNet (IResNet-100) architecture [42] as the neural net backbone for the FR system. The ArcFace model is widely used FR system [16, 17, 20, 22]. It employs an additive angular margin loss to enforce intra-class compactness and inter-class distance, which can enhance the discriminative ability of the feature embeddings [26]. ElasticFace builds upon the ArcFace model by using an elastic penalty margin over the fixed penalty margin used by ArcFace. This change results in an FR system with state-of-the-art performance [31]. Lastly, the AdaFace model employs an adaptive margin loss by weighting the loss relative to an approximation of the image quality [32]. The image quality is approximated via feature norms and is used to give less weight to misclassified images, reducing the impact of “low” quality images on training. This improvement allows the AdaFace model to achieve state-of-the-art performance in FR tasks.

The AdaFace and ElasticFace models are trained on the MS1M-ArcFace dataset, whereas the ArcFace model is trained on the MS1M-RetinaFace dataset. *N.B.*, the ArcFace model used in the identity loss is not the same ArcFace model used during evaluation. The model used in the identity loss is an IResNet-100 trained on the Glint360k dataset [43] with the ArcFace loss. We use the cosine distance to measure the distance between embeddings from the FR models. All three FR systems require images of  $112 \times 112$  pixels. We resize every images, post alignment from dlib which ensures the images are square, to  $112 \times 112$  using bilinear down-sampling. The image tensors are then normalized such that they take value in  $[-1, 1]$ . Lastly, the AdaFace FR system was trained on BGR images so the image tensor is shuffled from the RGB format to the BGR format.

## C.5. Detectability Study

Following the approach taken in [20, 21] we design a detectability study of the morphing attacks using a pre-trained SE-ResNeXt101-32x4d network developed by NVIDIA. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model [44] with added Squeeze-and-Excitation layers [45] pre-trained on the ImageNet [46] dataset. We employ  $k$ -fold stratified cross validation to ensure that the class balance between morphed and bona fide images is preserved across each fold. We opt to use  $k = 5$  for our study. The powerful SE-ResNeXt101-32x4d is used as the backbone for the S-MAD detector. We introduce replace the last layer with a fully connected layer with two outputs which denote the log probabilities of the bona fide and morphed classes. For each training dataset the backbone is fine-tuned for 3 epochs with an exponential learning rate scheduler and differential learning rates to combat any potential overfitting during training. We begin with a learning rate of 0.001 on the fully connected layer and reduce the learning rate exponentially to a learning rate of  $10^{-7}$  for each layer further away from the fully connected layer. *N.B.*, the learning rate scheduler has step size of 3 and an exponential rate  $\gamma = 0.1$ . To further combat overfitting, the cross entropy loss used for training is used with label smoothing parameter 0.15. Additionally, we opt to scale the EMA decay  $\beta_{ema}$ , with the batch size  $M$  in accordance to

$$\beta_{ema} = \left(\frac{1}{2}\right)^{\frac{M}{1000}} \quad (20)$$

in a manner similar to that of the scaling rule used to update the generator in the Alias-Free GAN [47] and recent work has shown the benefit of scaling the EMA decay with batch size [48]. In our experiments we use a batch size  $M = 128$ , and therefore  $\beta_{ema} \approx 0.915$ . *N.B.*, none of the bona fide images and morph pairs used in the training fold are present for the validation fold and the morph pairs in the validation fold are identical for each morphing attack to ensure a fair comparison.

## D. Additional Results

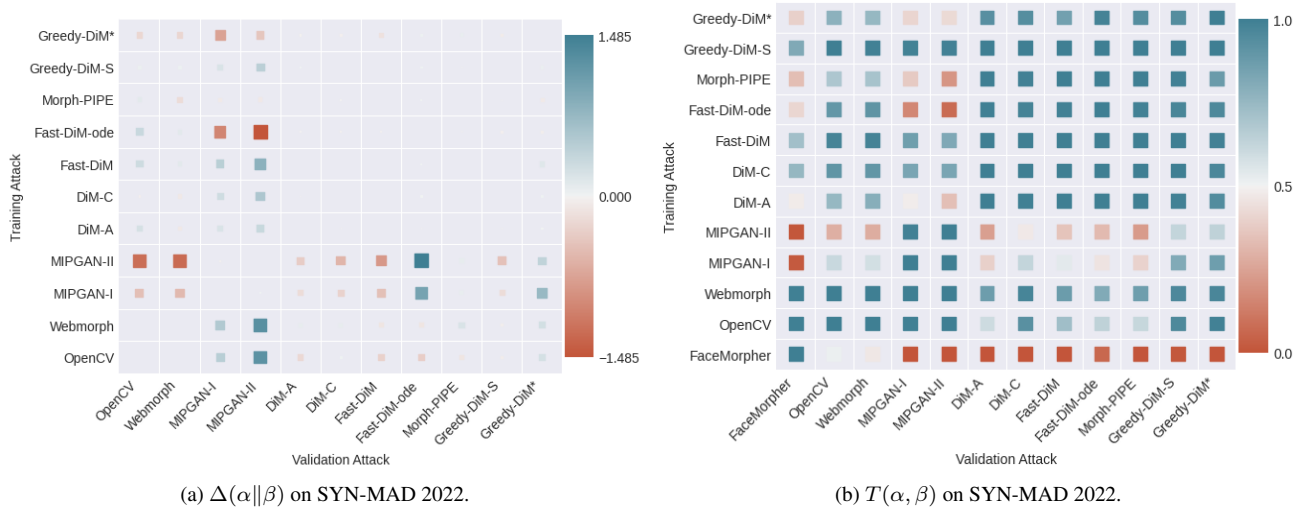


Figure 5. The RSM and Transferability metrics.

### D.1. Relative Strength Metric

We also report the Relative Strength Metric (RSM) proposed by Blasingame *et al.* [20]. The RSM measures how “strong” one morphing attack is to another attack, *i.e.*, if a MAD model is trained to detect one attack, how well can it detect the other morphing attack? For two morphing attacks  $\alpha, \beta$ , the RSM from  $\alpha$  to  $\beta$  is defined as

$$\Delta(\alpha||\beta) = \log \left( \frac{T(\alpha, \beta)}{T(\beta, \alpha)} \right) \quad (21)$$

where  $T(\alpha, \beta)$  denotes the probability of a MAD model trained on  $\alpha$ ,  $f^\alpha : \mathcal{X} \rightarrow \{0, 1\}$  is able to detect the morphed attack  $\beta$  given  $f^\alpha$  detects  $\alpha$ , *i.e.*,

$$T(\alpha, \beta) = P(f^\alpha(X^\beta) = 1 \mid f^\alpha(X^\alpha) = 1) \quad (22)$$

where  $X^\alpha, X^\beta$  denotes morphed images generated by  $\alpha$  and  $\beta$ , respectively.

In Figure 5 the RSM and transferability metrics are plotted. Note, in Figure 5a we opt to omit the FaceMorpher entry as it was an outlier with an RSM of approximately  $-20$  relative to many of the other morphing attacks, causing scaling issues with the other entries of the plot. Corroborating our results from Table 7 the Greedy-DiM\* algorithm is slightly “weaker” than the other DiM morphs relative to the landmark and GAN-based morphs. Interestingly we also note that Fast-DiM-ode is “weaker” relative to MIPGAN attacks. This is a strange observation as the only difference between Fast-DiM and Fast-DiM-ode [21] is the forward ODE solver and in terms of MMPMR Fast-DiM performs similarly to DiM-C. We leave further investigation into this matter as future work. We note that the FaceMorpher attack is especially “weak” to other morphs with extremely low transferability to all representation-based morphing attacks, see Figure 5b. Likewise, the MIPGAN attacks are generally quite “weak” as well.

### D.2. Comparison of all Design Choices

In this section we presented additional experiments on the impact of different design choices on Greedy-DiM algorithms that were not able to be included in the main paper. We begin by exploring the impact the choice  $\mathcal{H}$  plays on the performance

Table 9. Impact of heuristic function on Greedy-DiM\*.

Heuristic	Network	MMPMR( $\uparrow$ )		
		AdaFace	ArcFace	ElasticFace
Identity	ArcFace	<b>99.59</b>	98.77	<b>99.59</b>
Worst-Case L2	ArcFace	98.77	98.16	98.98
Worst-Case Cosine	ArcFace	83.84	73.21	80.37
Identity	LPIPS	93.05	89.37	93.66
Worst-Case L2	LPIPS	93.25	89.78	93.46
Identity + Perceptual Loss	ArcFace and LPIPS	99.18	<b>99.18</b>	99.39

of Greedy-DiM. We define the worst case loss as the distance between the worst-case morph and the true morph as

$$v_W = \frac{F(\mathbf{x}_0^{(a)}) + F(\mathbf{x}_0^{(b)})}{2} \quad (23)$$

$$\mathcal{L}_W = d(F(\mathbf{x}_0^{(ab)}), v_W) \quad (24)$$

In this construction  $v_W$  is the worst-case morph as it is the morphed code which minimizes L2 distance between both embeddings of the bona fide images. Alternatively, for the cosine distance the worst-case morph is defined as

$$v_W = \frac{F(\mathbf{x}_0^{(a)}) + F(\mathbf{x}_0^{(b)})}{\|F(\mathbf{x}_0^{(a)}) + F(\mathbf{x}_0^{(b)})\|} \quad (25)$$

Recent work [49] has explored a similar idea of using the worst-case morphs to guide the generation process of GANs. We also examine the use of an LPIPS [50] trained VGG [51] network. In Table 9 we outline the impact the choice of heuristic function. We observe that using a worst-case morph formulation with the ArcFace FR system drops the performance severely over using the identity loss. Additionally, using the worst-case morph in terms of cosine distance further drops the performance of the Greedy-DiM\* algorithm. We observed that using LPIPS over ArcFace as the neural net backbone with the identity or worst-case losses also decreased performance. Lastly, we noticed only a marginal change in performance, positive and negative, when incorporating the LPIPS perceptual loss in addition to the ArcFace identity loss.

Table 10. Impact of ODE solver on Greedy-DiM\*.

ODE Solver	MMPMR( $\uparrow$ )		
	AdaFace	ArcFace	ElasticFace
DDIM	99.59	99.18	99.39
DPM++ 2M	98.98	99.18	99.39

As suggested in [21] we explore using the DPM++ 2M solver proposed by Lu *et al.* [52] over the DDIM solver. The DPM++ 2M solver is a second-order multi-step ODE solver designed for the PF-ODE. The drop in slight drop in performance is unsurprisingly, as the multi-step solver requires previous iterations; however, in the Greedy-DiM\* algorithm we update the iteration at each step breaking the theoretical assumptions underpinning the derivations in [52]. Now as Greedy-DiM\* is globally optimal this should lessen the impact, but there is still the variability of gradient descent which could play a role.

Table 11. Impact of initial noise variable.

Algorithm	Initial $\mathbf{x}_T$	MMPMR( $\uparrow$ )		
		AdaFace	ArcFace	ElasticFace
Fast-DiM [21]	$\mathbf{x}_T = \text{slerp}(\mathbf{x}_T^{(a)}, \mathbf{x}_T^{(b)}, 0.5)$	92.02	90.18	93.25
Fast-DiM [21]	$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$	4.5	3.48	2.04
<b>Greedy-DiM*</b>	$\mathbf{x}_T = \text{slerp}(\mathbf{x}_T^{(a)}, \mathbf{x}_T^{(b)}, 0.5)$	<b>99.59</b>	<b>98.77</b>	<b>99.59</b>
<b>Greedy-DiM*</b>	$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$	70.14	75.05	63.8

Next we examine if it is even necessary to encode the bona fide images into their noisy representations or if we can simply start the Greedy-DiM\* algorithm with random noise. Preechakul *et al.* [23] posit that the semantic information is encoded in  $\mathbf{z}$  and that  $\mathbf{x}_T$  only encodes the “stochastic” information, *i.e.*, information which doesn’t impact semantic meaning, like

the randomness of hair strands, but not texture or color. Concurrent work [21] has called this assertion into question showing that the stochastic information  $\mathbf{x}_T$  is essential to creating high quality morphs. However, since the Greedy-DiM\* algorithm searches over the whole image space with a greedy search we explore if it is still necessary to start from the morphed initial noise or if random noise would suffice. In Table 11 we found the morphed initial noise still plays a large role in helping the gradient descent algorithm find an optimal solution. We did notice, however, that the Greedy-DiM\* algorithm fared much better than the Fast-DiM algorithm when starting with random noise resulting in a roughly 70% increase in MMPMR.

Table 12. Impact of the number of sampling steps on Greedy-DiM\*.

$N$	MMPMR( $\uparrow$ )		
	AdaFace	ArcFace	ElasticFace
20	99.59	98.77	99.59
50	99.18	99.18	99.39
100	99.59	99.18	99.39

We perform a small study on the impact of the number of sampling steps on the MMPMR of the resulting morphed image the results of which are shown in Table 12. We observe that the number of sampling steps has a negligible impact on performance and so we opt to use  $N = 20$  to conserve compute resources.

Table 13. Impact of optimizer momentum parameters on Greedy-DiM\*.

$\beta_0$ $\beta_1$		MMPMR( $\uparrow$ )		
		AdaFace	ArcFace	ElasticFace
0.9	0.999	99.59	98.77	99.59
0.5	0.9	100	100	100

Lastly, in Table 13 we explore the impact of the optimizer momentum parameters on the Greedy-DiM\* algorithm. The Adam [53] optimizer, and its derivatives like RAdam [39], recommend the use of the parameters  $\beta_0 = 0.9$  and  $\beta_1 = 0.999$  by default. We explore alternative parameters that were used by Gu *et al.* [40] with  $\beta_0 = 0.5$  and  $\beta_1 = 0.9$ . We find that these parameters are the final missing piece in maximizing the MMPMR performance on the studied FR systems resulting in a 100% MMPMR at FMR = 0.1%. We posit that because the parameters,  $\beta_0, \beta_1$  control the exponential decay rates of the Adam optimizer, while the initial recommended parameter settings are appropriate for training a neural network or gradient descent on, the  $\epsilon$  space needs a faster update to converge quickly, resulting in the lower decay rates. Moreover, as our search space is considerably simpler than that of the parameter space of a large neural network, we can reduce the amount of smoothing used by the optimizer.

Table 14. Time needed to create 500 morphed images on a single NVIDIA Tesla V100 32GB PCIe GPU.  $\dagger$  denotes the batched implementation of Morph-PIPE.

Model	NFE	Batch Size	Time (HH:MM:SS)
Fast-DiM-ode [21]	150	32	01:10:14
Fast-DiM [21]	300	48	01:30:01
DiM-C [20]	350	48	01:43:16
DiM-A [20]	350	48	01:43:17
<b>Greedy-DiM*</b>	270	40	02:28:57
<b>Greedy-DiM-S</b>	350	48	02:54:47
Morph-PIPE [22]	2350	48	07:25:16
Morph-PIPE $^\dagger$ [22]	350	4	07:35:22

### D.3. Empirical Time Analysis

While we chose to report NFE as our measure of efficiency as it is a standard method of reporting the computational cost of diffusion models and is invariant to the underlying hardware, for completeness we report the time to generate the morphed images used in this paper using our hardware. We recorded the time it took to generate all 500 morphed images from the SYN-MAD 2022 dataset and reported our results in Table 14. For each model we selected the largest batch size that would fit on our GPU without running into Out Of Memory errors during the creation of the morphs. *N.B.*, for increased efficiency of GEMM kernel calls we used an integer multiple of 8 whenever possible. Table 14 shows that while the search



and optimization step of the Greedy-DiM algorithm adds some overhead compared to the original DiM morphs, or especially the recently proposed Fast-DiM morphs, in comparison with the only other identity guided DiM model, Morph-PIPE, our added overhead is far less. Combined with the superior performance of Greedy-DiM\*, we believe that this represents a significant contribution as it is far faster than Morph-PIPE with a small amount of overhead compared to the original DiM morphs.

For completeness, we also implement an alternative implementation for Morph-PIPE, wherein the candidate morphs are generated in a batch of size 21 blends instead of being calculated sequentially; however, due to the computationally demanding requirements of generating all 21 blends at once, we had to significantly reduce the batch size in order to fit into the memory of our GPU. We denote this implementation as Morph-PIPE<sup>†</sup> in Table 14. This approach is therefore limited only to groups with significant amounts of computing resources. Conversely, our method can be used by groups with more modest hardware, providing greater accessibility to researchers. Interestingly, the batched implementation of Morph-PIPE takes slightly longer than the regular implementation, however, this difference is quite small and unlikely to be significant. We believe the experimental results from Table 14 justify the reasoning used to make our choices in Appendix C.1.

The overhead introduced by Greedy-DiM is rather minimal, as the search/optimization step of the Greedy-DiM models is *relatively* cheap as it involves only performing a search over the possible  $\epsilon = \epsilon_\theta(\mathbf{x}_t, \mathbf{z}, t)$ . As such, the gradient calculation is fairly cheap, as we do *not* backpropagate through the U-Net  $\epsilon_\theta$ . It can be shown that the gradient of the heuristic function w.r.t.  $\epsilon$  involves only a single call of an automatic differentiation algorithm, apart from any additional calls to backpropagate through the heuristic function, to evaluate  $\nabla_{\mathbf{x}}\mathcal{H}$ . Remember that  $\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sigma_t \epsilon}{\alpha_t}$  then the gradient of interest is  $\nabla_{\epsilon} \mathcal{H} = -\frac{\sigma_t}{\alpha_t} \nabla_{\mathbf{x}} \mathcal{H}$  which is only as expensive as the image size and heuristic function.

## E. Additional Images

In this section we provide additional samples produced by the Greedy-DiM-S algorithm, Figure 6, and Greedy-DiM\* algorithm, Figure 7. In Figure 7 we observe consistent high frequency noise artefacts that looks akin to film grain whereas the Greedy-DiM-S, and other DiM variants for that matter, lack this quality. We posit this a by product of the optimization process and this noise encodes information to trick the FR system similar to the technique of adversarial perturbations [54].



Figure 6. Morphed images generated via Greedy-DiM-S.



Figure 7. Morphed images generated via Greedy-DiM\*.