Source Driver Output

Latent Space Control in Autoencoders for Synthetic Face Generation in Driver Monitoring System Validation

Master's thesis in Complex Adaptive Systems MPCAS

Jakob Nilsson
Johan Philis

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Latent Space Control in Autoencoders for Synthetic Face Generation in Driver Monitoring System Validation

Jakob Nilsson
Johan Philis

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Latent Space Control in Autoencoders for Synthetic Face Generation in Driver Monitoring System Validation

Supervisors: John Dahl, Zenseact
Examiner: Jonas Fredriksson, Department of Electrical Engineering

Master's Thesis 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Generated images by our model (right) of a manipulated source face (left) controlled by rotations from driving frames (center).

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Latent Space Control in Autoencoders for Synthetic Face Generation in Driver Monitoring System Validation
Jakob Nilsson, Johan Philis
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Driver errors are the primary cause of road traffic accidents, often resulting from inattention or distraction. Most modern cars are equipped with camera-based driver monitoring systems (DMS) to estimate the driver's state, helping to minimize the risk of such accidents. Validation of the DMS requires large amounts of expensive data of driver faces to cover common driving scenarios. By simulating these scenarios with synthetic data, one could potentially improve the validation process. The investigated idea is to use various setups of autoencoders to generate synthetic data, with the possibility to control latent variables such as head position and rotation. The controllability is achieved through a proposed training step where the latent variables are swapped, enabling the autoencoders to have a structured latent space containing a steerable position or rotation representation. The results are benchmarked against a generative model called LivePortrait, and the compatibility of the synthetic data with existing open-source tracking software is investigated. The results demonstrate that the proposed model is capable of generating synthetic videos that are compatible with Google's head rotation tracking algorithm from the MediaPipe framework. To enhance the practical value of these models, future work should focus on evaluating the synthetic videos using tracking algorithms from a real DMS and extending the model to allow for controlling eye gaze direction.

Keywords: driver monitoring systems, autoencoder, structured latent space, synthetic faces, head rotation, deep learning

Acknowledgements

We would like to thank Zenseact, and especially our supervisor John Dahl for the opportunity to pursue this project and for his valuable assistance. We would also like to thank Jonas Fredriksson, our examiner at Chalmers, for his continuous support and insightful suggestions.

Jakob Nilsson, Johan Philis, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADAS	Advanced driver-assistance systems
ACC	Adaptive Cruise Control
BLIS	Blind Spot Information Systems
CAS	Collision Avoidance Systems
DMS	Driver Monitoring Systems
Euro NCAP	European New Car Assessment Program
GAN	Generative Adversarial Network
LKA	Lane Keep Assistance
RPE model	Reconstruction-Position-Encoding model
SPE model	Swap-Position-Encoding model
SD model	Swap-Direct model
VAE	Variational Autoencoder

Contents

List of Figures

1
Introduction

In 2022, there were 5.93 million police-reported motor vehicle accidents, causing 42514 deaths in the USA [18]. Furthermore, according to the World Health Organization, traffic injury is the leading cause of death for kids and young adults aged 5-29 years [23]. To make roads safer, modern vehicles are equipped with both passive and active technologies aimed at improving safety and enhancing the driving experience. During the last decades, the focus has moved more towards active safety [3], where advanced driver assistance systems (ADAS), including driver monitoring systems (DMS), are among the most promising technologies [13].

1.1 Advanced Driver Assistance Systems

Advanced driver assistance systems (ADAS) is the term for a range of systems that aid the driver in safe driving, including systems like collision avoidance systems (CAS), lane-keeping assistance (LKA), driver monitoring systems (DMS), adaptive cruise control (ACC), parking aid and blind spot information systems (BLIS) [7][3]. The National Safety Council, a nonprofit safety advocate in the USA, predicts that ADAS systems can prevent 62% of total traffic deaths [19]. However, for ADAS to function reliably and avoid inappropriate interventions, such as unwanted lane corrections, it is essential to assess the driver's state. This is where DMS comes into play.

1.2 Driver Monitoring Systems

A study by the National Motor Vehicle Crash Causation Survey assigns 94% of accidents to driver errors [4]. Among these, 41% are attributed to recognition errors, such as driver inattention or distraction. Driver monitoring systems are becoming a primary safety feature in modern cars (Euro NCAP) [7], and can help reduce accidents caused by such recognition errors. DMS refers to a variety of tools that collect driver data to assess driving capability [7]. By providing notifications regarding the driver's state, referring to the physical and mental condition of the driver, the driver can be alerted, and the rest of the ADAS systems can become more sensitive to prevent accidents earlier. Another problem that arises in modern driving with increased automation is an increase in driver disengagement due to preoccupation with secondary tasks, such as using a smartphone or the car's infotainment system [14]. DMS can help reduce the risk of accidents rooted in human error, which often results from drivers relying too heavily on autonomous driving features [16].

There are three common types of DMS: kinematic-, biology-, and image-based systems [1].

Kinematic-based systems use measurements like steering variability, lane positioning, or braking patterns to assess the driver's state [9] indirectly. The systems compare the collected data with the standard driving performance, and when the difference is significant, ADAS features can be triggered. A weakness of these systems is their susceptibility to external factors, such as weather or traffic conditions, which can compromise even the performance of a perfectly focused driver. Additionally, when ADAS features like ACC or lane centering are active, measuring the driver's state through braking patterns or lane positioning is prevented.

Biology-based systems rely on measuring units mounted on the human body, such as electrocardiograms. These measurements have the potential to infer the driver's state accurately, but they are impractical for passenger vehicles and are therefore mainly used for research purposes [7] [22].

Image-based systems utilize cameras to analyze the facial features of the driver, including head, mouth, and eye movements, to detect problematic driver states. Eye movement includes features like eyelid movement, pupil dilation, and gaze direction [7]. Eye tracking has undergone significant improvements in recent decades, thanks to advancements in computer vision and deep learning methods [10]. These models employ a learning-based approach and can identify key points, such as pupils, eyelids, or the boundaries of the sclera [20]. There are multiple deep learning frameworks for detecting such key points, for example MediaPipe [5], which can be used to detect head position, rotation, and eye gaze direction.

1.3 Problem

Due to the susceptibility of kinematic-based systems to external factors and the impracticality of biology-based systems, image-based systems are the most promising form of DMS. For image-based systems to be effective, they must function correctly across a wide range of typical driving scenarios such as different driver behaviors, lighting conditions and driver appearances. However, collecting real-world driving data for training and validating the image-based systems is both expensive and complicated due to privacy concerns. Consequently, there is a growing interest in generating synthetic data for validation. The synthetic data needs to be of sufficient quality to be compatible with the head and eye tracking algorithms of DMS. This may now be feasible thanks to recent advancements in deep learning, enabling the generation of hyper-realistic synthetic videos of human faces. This project aims to investigate the generation and compatibility of synthetic videos with tracking algorithms to validate DMS features. Existing models for synthetic video generation are computationally demanding and lack the direct controllability necessary to accurately generate driving scenarios and driver head motions. Therefore, we train and evaluate our model to generate synthetic faces with control of head position and rotation.

1.4 Project Outline

In the first part of the project, an autoencoder is trained to utilize a controllable latent space for generating frames of a cut-out face at desired positions. The key finding that enables control of position in a high-dimensional latent space is to swap the position representation between latent vectors in a batch, inspired by the transfer loss in VASA-1 [24]. Using the same core idea, the second part of the project explores training an autoencoder capable of directly controlling head rotation, in terms of pitch, yaw, and roll, and train the model on realistic frames of a face with random head rotations. Using this model, frames are generated with arbitrary head rotation controlled by a driving video, and the results are evaluated against a portrait animation framework called LivePortait [6]. The results are interpreted both visually and quantitatively by tracking the head rotation of the generated videos and the driving video using the MediaPipe framework [5], to investigate the compatibility of the fake videos with tracking algorithms.

1.5 Limitations

The autoencoders are only trained on one face in one setting. This is a limitation since it would require training a new model for each test person when used for validation. The model could still be useful, although it might be more efficient to train a larger model on multiple faces. Furthermore, the final model is only aimed at controlling head rotation, even though controlling eye gaze direction would be crucial for a proper validation of DMS. Finally, the proposed model is only evaluated and benchmarked against LivePortrait's videos using rotation tracking, although LivePortrait's videos could potentially be compatible with eye gaze tracking as well.

2
Theory

In this chapter, related work regarding synthetic video generation is discussed. Furthermore, autoencoders and the idea of interpretable latent spaces are described.

2.1 Synthetic Video Generation

The generation of deepfake visual content is experiencing rapid advancement, transitioning from text-to-image models to text-to-video models in just one year, between 2021 and 2022 [26]. Fake videos that are almost indistinguishable from reality can now be created using generative models, such as generative adversarial networks (GANs), variational autoencoders (VAEs), autoregressive models, and diffusion models. Although text-to-video models have gained most of the mainstream attention, video generation is a widely researched topic with successful models using all kinds of input modes.

Synthetic video generation can be categorized into video prediction, editing, and synthesis [26]. Video prediction involves predicting the next frames of a video, where RaMViD [8] successfully predicts future frames and fills in the gaps between frames (infilling) using diffusion models and random masking. Deep learning methods for video editing enable users to customize scenes while maintaining temporal consistency. DiffVideoAE [11] is a model that fine-tunes face-based speech videos by modifying facial attributes. Pix2Video [2] incorporates self-attention features across frames to improve video smoothness. Models for video synthesis are often controlled by text, audio, an image, or a combination of them.

Portrait animation aims to generate realistic videos from a single source image, by applying control signals from a driving video, audio, text, or other media [6]. Wang et al. introduced vid2vid [21], a video-driven framework for portrait animation that can generate videos from a single image and a sequence of 3D facial keypoints. LivePortrait [6] improves upon vid2vid [21] by scaling up the training data and introducing a stitching module for precise controllability of mouth- and eyelid openness. Talking face animation is a widely researched topic within portrait animation, which involves generating videos of a talking face with lip movements and facial expressions that match an audio clip. Some recent advancements in talking face animation are VAE-based models (SadTalker [25]) and diffusion-based models (DreamTalk [15], VASA-1 [24]). VASA-1 utilizes a latent representation of images

that disentangles facial structure, identity, facial dynamics, and head pose, allowing for the transfer of these features between videos. A key idea for ensuring disentanglement between facial dynamics and head pose is to swap latent variables between different images during training. In VASA-1, the latent representation of a frame consists of an appearance volume, an id of the person, the head pose and the facial dynamics. Given two randomly sampled frames i and j of the same person, with similar appearance volume and id, the head pose from frame i is transferred to frame j, and the facial dynamics from frame j is transferred to frame i. This creates two pairs of latent representations with identical head pose and facial dynamics, which are then both decoded into fake frames. VASA-1 shows that introducing a transfer loss between pairs of such decoded frames significantly increases disentanglement.

2.2 Autoencoders

A common approach in synthetic video generation is to first encode frames into latent representations of lower dimensionality, then process these representations for improved efficiency, and finally decode them back into frames. The latent representation is usually a vector in the so-called latent space. Ideally, the latent space is interpretable, so that changes to the latent vector produce desirable changes in the output. Autoencoders are a common deep learning architecture for this, consisting of an encoder implemented as a neural network, with as many output neurons as the dimensionality of the latent space, and a decoder also implemented as a neural network, taking the output neurons of the encoder as its input.

Figure 2.1 shows an autoencoder in its simplest form, where an input X_i in a batch is passed through the encoder, producing an intermediate latent representation. This representation is then passed through a decoder to generate the output

$$\hat{X}_i = A(X_i), \tag{2.1}$$

where A is the autoencoder, with the goal of reconstructing the input as accurately as possible. A loss function $L_i(\hat{X}_i, X_i)$ computes the reconstruction error between the input X_i and the output $\hat{X}_i$. A common reconstruction loss function is the mean squared error (MSE) over each pixel, defined as

$$L_i(\hat{X}_i, X_i) = \text{MSE}(\hat{X}_i, X_i) \tag{2.2}$$

$$= \frac{1}{rc} \sum_{j,k \in \mathcal{R} \times \mathcal{C}} [\hat{X}_{i,j,k} - X_{i,j,k}]^2, \tag{2.3}$$

where $\mathcal{R} = \{j\}_0^{r-1}$ and $\mathcal{C} = \{k\}_0^{c-1}$ are the sets of indices of rows and columns. The total loss for a batch becomes

$$L(\hat{X}, X) = \frac{1}{brc} \sum_{i,j,k \in \mathcal{B} \times \mathcal{R} \times \mathcal{C}} [A(X_i)_{j,k} - X_{i,j,k}]^2, \tag{2.4}$$

where $\mathcal{B} = \{i\}_0^{b-1}$ is the set of batch indices.

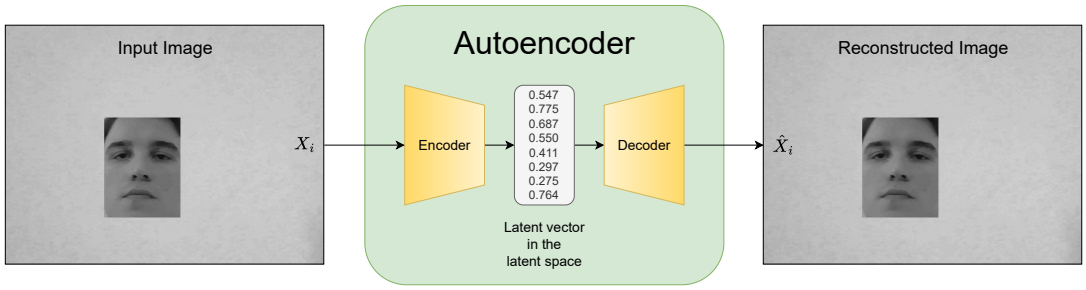


Figure 2.1: Schematic view of an autoencoder. The figure illustrates how an input image, X_i is encoded into the latent space, in this case as an 8-dimensional latent vector of seemingly random numbers, which are produced by the learned network weights of the autoencoder. The latent vector is then passed to the decoder, which produces an output image $\hat{X}_i$ that is supposed to mimic the input. The figure illustrates the compressive nature of autoencoders, showing how high-dimensional inputs can be encoded as low-dimensional vectors.

The loss function is minimized using gradient descent during training to improve the reconstruction. Since the training process does not require external labels, as the target outputs are identical to the inputs themselves, it is considered a form of self-supervised learning. Overall, the autoencoder learns to reconstruct its inputs, and the latent space it forms provides opportunities for further analysis or manipulation, such as exploring learned representations or generating variations. [17]

3

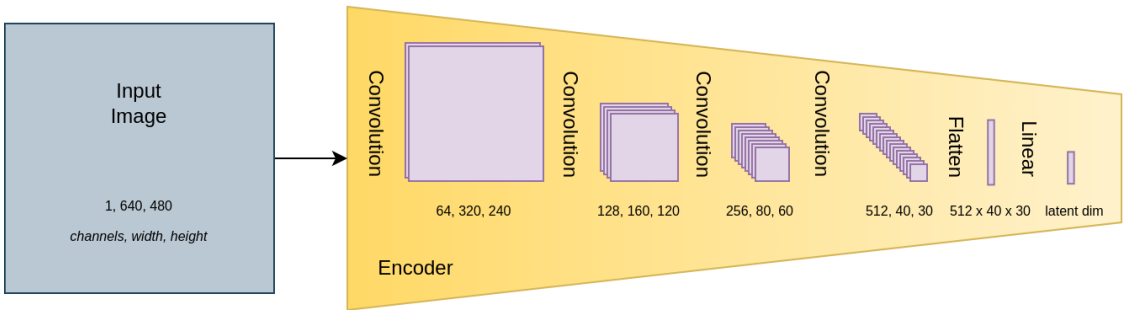
Controlling the Position of a Cut-Out Face

This chapter introduces an autoencoder for reconstructing simple images of faces, with a focus on the controllability of the reconstruction. We use a dataset containing a cut-out face in different positions, and the aim is to control the position (x, y) of the face. Experiments with different models and latent dimensionalities are performed to learn how to generate high-quality images while controlling the position. A key finding that enables controllability is to *swap* a relevant part of the latent representations between samples within a batch.

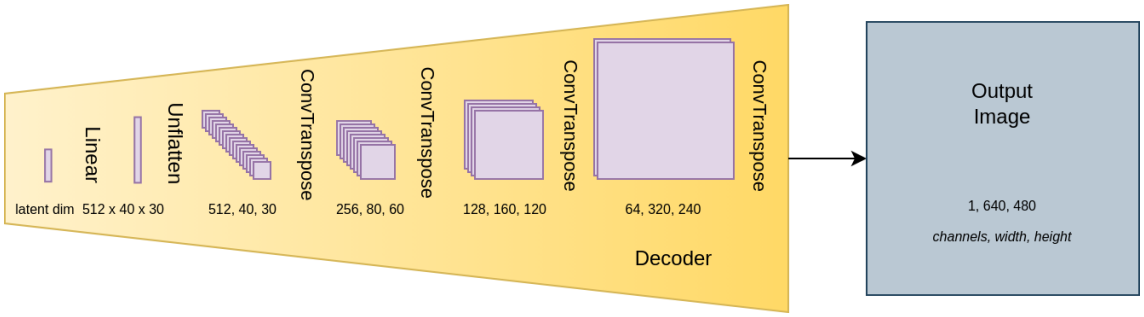
3.1 Method

All of the models utilize a convolutional autoencoder architecture, consisting of an encoder and a decoder, as illustrated in Figure 3.1. The encoder consists of four 2D convolutional layers. The first convolutional layer scales the greyscale image to 64 feature maps and reduces the width and height of the image by half. The following three layers double the number of feature maps and continue to halve the image width and height. Finally, the 2D feature maps are flattened to a vector, and a final linear layer maps it to a latent vector. The decoder mirrors the architecture of the encoder but uses convolutional transpose layers instead of convolutional layers to reconstruct the image back to its original dimensions. The decoder starts with a linear layer, followed by an unflattening operation that reshapes the latent representation into feature maps. The feature maps are then passed through four convolutional transpose layers that double the resolution at each step, ultimately reconstructing the image.

The first model (Section 3.1.3) focuses solely on reconstruction, while the second model (Section 3.1.4) is additionally trained to encode position in the latent space. The third model (Section 3.1.5) introduces the swapping of encoded positions between latent vectors in a batch, a key step that enables control of position for more than two latent dimensions. In the fourth model (Section 3.1.6), the model no longer encodes position; instead, the true position is injected straight into the latent space.



(a) Convolutional Encoder.



(b) Convolutional Decoder.

Figure 3.1: Convolutional encoder and decoder. The encoder consists of four convolutional layers and a final linear layer that projects the input down to a latent representation of varying dimensionality. The decoder architecture is the reverse of the encoder, utilizing convolutional transpose layers instead of convolutional layers. The purple squares represent feature maps, and the numbers underneath them represent the number of feature maps, as well as their width and height.

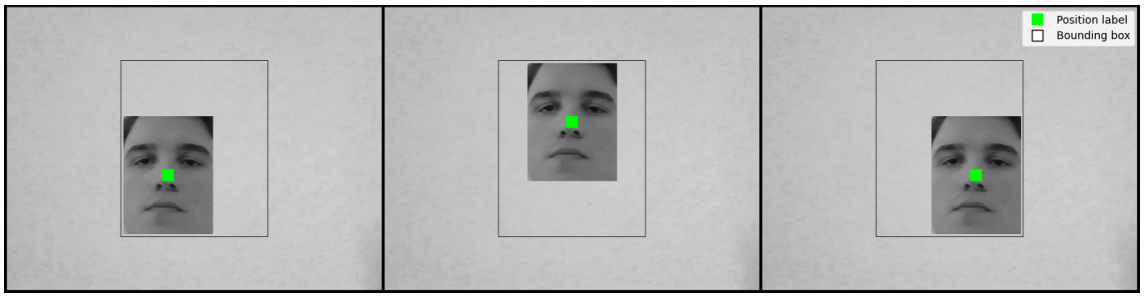


Figure 3.2: Sample from the position dataset. Images of a cut-out face in different positions on a static background. The position labels are marked in green, and the range of face positions is marked with a black bounding box.

3.1.1 Position Dataset

The position dataset comprises images of a cut-out face of a single person in various positions (x, y) against a fixed background, specifically a white wall. The dataset contains 1000 such images, each with a resolution of 640×480 pixels. In each direction, the face can move 50 pixels from the center and has a size of 152×200 pixels, creating a bounding box with dimensions $(2 \cdot 50 + 152) \times (2 \cdot 50 + 200)$, as shown in Figure 3.2. The bounding box marks the rectangle in which the face is allowed to move. This setup allows for a total of 10,000 possible image variations. The coordinate system adheres to a standard image convention: the origin $(0, 0)$ is located in the top-left corner, with the x-axis increasing to the right and the y-axis decreasing downward. The face positions are sampled uniformly within the bounding box.

3.1.2 Latent Dimensionality Experiment

The latent dimension experiment is designed to investigate the effect of varying the latent dimensionality on image generation performance. Four different autoencoder models are trained according to the experiment, with the position dataset. The ADAM [12] optimizer is used with a fixed learning rate of 0.0005, a batch size of four images, and latent dimensionalities in the set $\{2, 4, 8, 16, 32, 64, 128, 256\}$. For each latent dimensionality, three separate runs are performed using randomly initialized network weights and different random training and validation splits, each with 800 training images and 200 validation images. The four models include the Reconstruction model, the Reconstruction-Position-Encoding (RPE) model, the Swap-Position-Encoding (SPE) model and the Swap-Direct (SD) model. All pixel values and positional coordinates are normalized to values between 0 and 1 when used in the models.

3.1.3 The Reconstruction Model

The reconstruction model is a convolutional autoencoder trained for image reconstruction, and no attempts are made to control the face position with this model. It serves as a base model for the more advanced models, which are targeted at

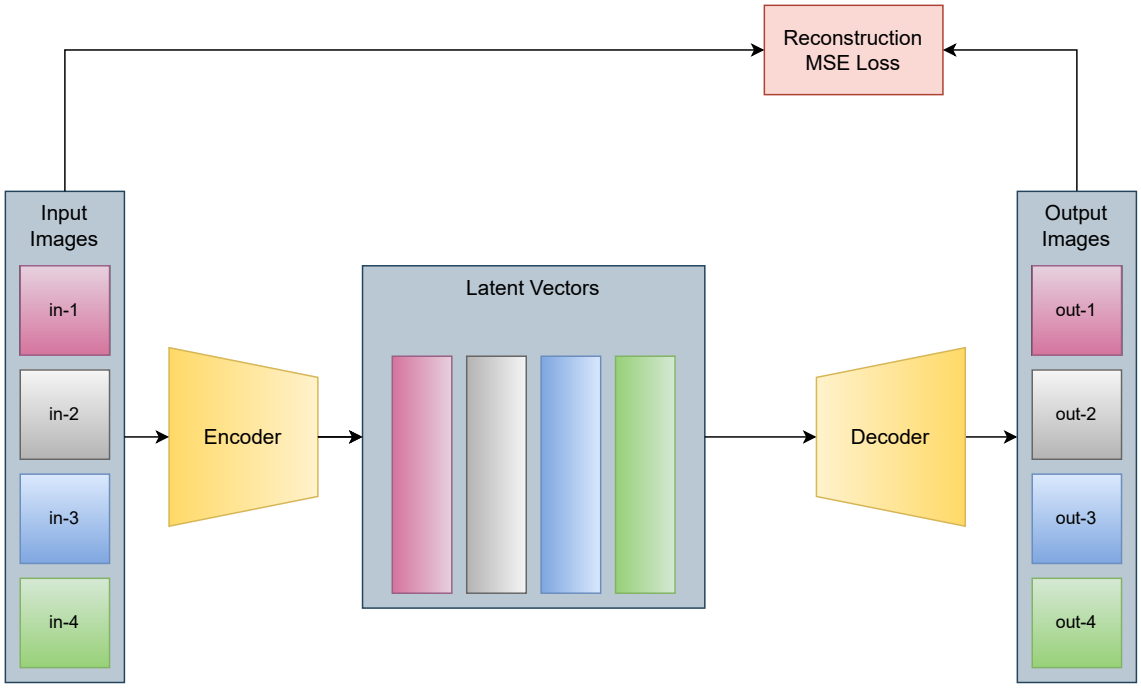


Figure 3.3: Training pipeline for the reconstruction model, utilizing the encoder and decoder from Figure 3.1. Batches of input images are encoded into latent vectors, and decoded into output images. An MSE loss, referred to as the reconstruction loss, is computed between the input and output images.

controlling position. Figure 3.3 shows its training pipeline, where input images are encoded into latent vectors of significantly lower dimensionality than the original, which are then decoded into output images. The objective of the model is to minimize the MSE loss, (2.2), also referred to as the reconstruction loss, between the input and output images. For each run and latent dimension of the experiment in Section 3.1.2, the model is trained and evaluated for 100 epochs.

3.1.4 The Reconstruction-Position-Encoding Model

Figure 3.4 shows the training pipeline for the RPE model, an extension of the Reconstruction model that is, apart from image reconstruction, trained to encode position in the latent space. The model learns to encode position in the first two dimensions of the latent vectors thanks to an MSE loss against the actual position labels. We refer to this loss as the position loss. The objective is to minimize the sum of the reconstruction and the position loss. The model is trained and evaluated multiple times for 100 epochs, using different latent dimensionalities, as described in Section 3.1.2. The idea is to investigate whether the position encoding can be modified in the inference stage, allowing for the generation of images at arbitrary positions within the bounding box.

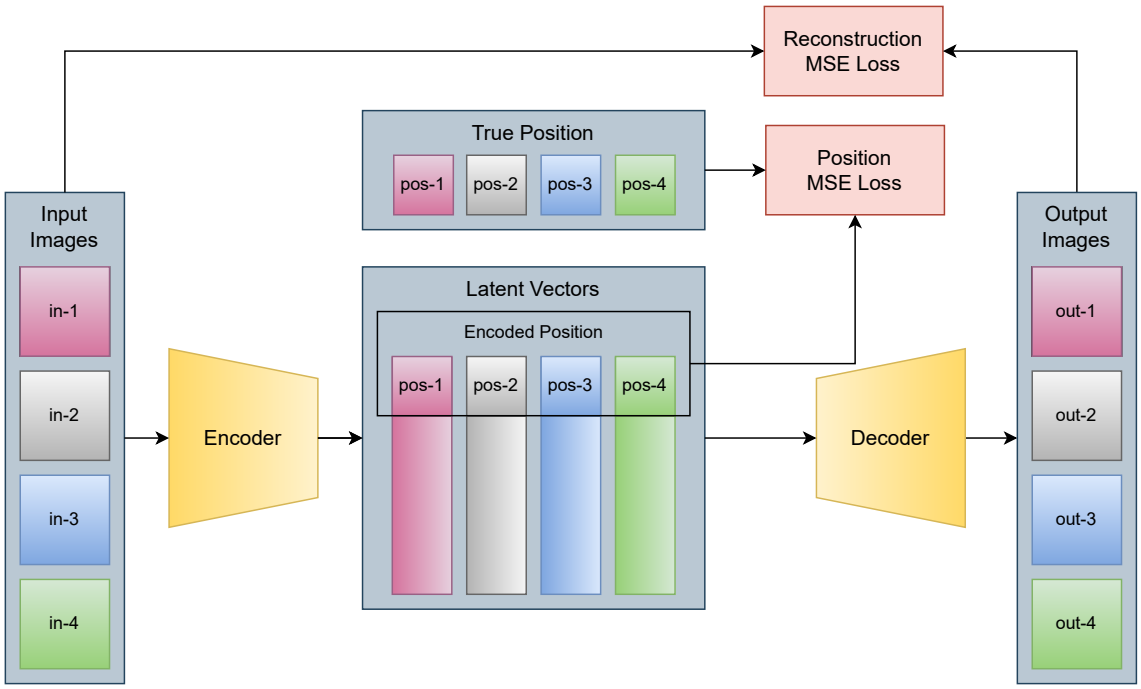


Figure 3.4: Training pipeline for the reconstruction-position-encoding model. As in the reconstruction model in Figure 3.3, the batch of input images is encoded into latent vectors, and decoded into output images, and an MSE loss, referred to as the reconstruction loss, is computed between the input and output images. In this model, an additional MSE loss, referred to as the position loss, is computed between the top 2 dimensions of the latent vectors and the true position. The model utilizes the encoder and decoder from Figure 3.1.

3.1.5 The Swap-Position-Encoding Model

Early experiments with the RPE model revealed difficulties in controlling position when the latent dimensionality exceeded 2. Meanwhile, models with a higher latent dimensionality tended to generate images of higher quality. To obtain the resolution benefits of a high-dimensional latent space with the steering abilities of a small latent space, we introduce *swapping* in the latent space. The swapping is inspired by the transfer loss in VASA-1, [24], between facial dynamics and head rotation. The difference from VASA-1 is that we only swap one component, in this case the position, and compute the loss against the ground truth input image rather than between two fake images.

The training pipeline for the SPE model, including the swapping mechanism, is illustrated in Figure 3.5. The parts of the latent vectors that encode position are pairwisely swapped before the latent vectors are decoded into output images. For our case with a batch size of four inputs, this corresponds to swapping the first and second, and the third and fourth positional encoding. The same pairwise swapping is applied to the input images before they are used as labels in the MSE loss, referred to as the *swap* loss, against the output images. The objective is to minimize the sum of the swap loss and the position loss. The model is trained and evaluated for 250 epochs, for each run and latent dimensions according to the experiment described in Section 3.1.2.

3.1.6 The Swap-Direct Model

With the SD model, we want to see if it is possible to bypass the use of position encoding in the model and instead inject the true position directly into the latent space, as shown in Figure 3.6. The position loss is removed, and the injected true positions are swapped pairwise; the rest of the training pipeline remains the same as for the SPE model. It should be noted that for just two latent dimensions, the encoder is removed and the decoder is trained to generate images directly from the true position, guided by the swap loss. The model is trained and evaluated multiple times for 250 epochs, using different latent dimensions as described in Section 3.1.2.

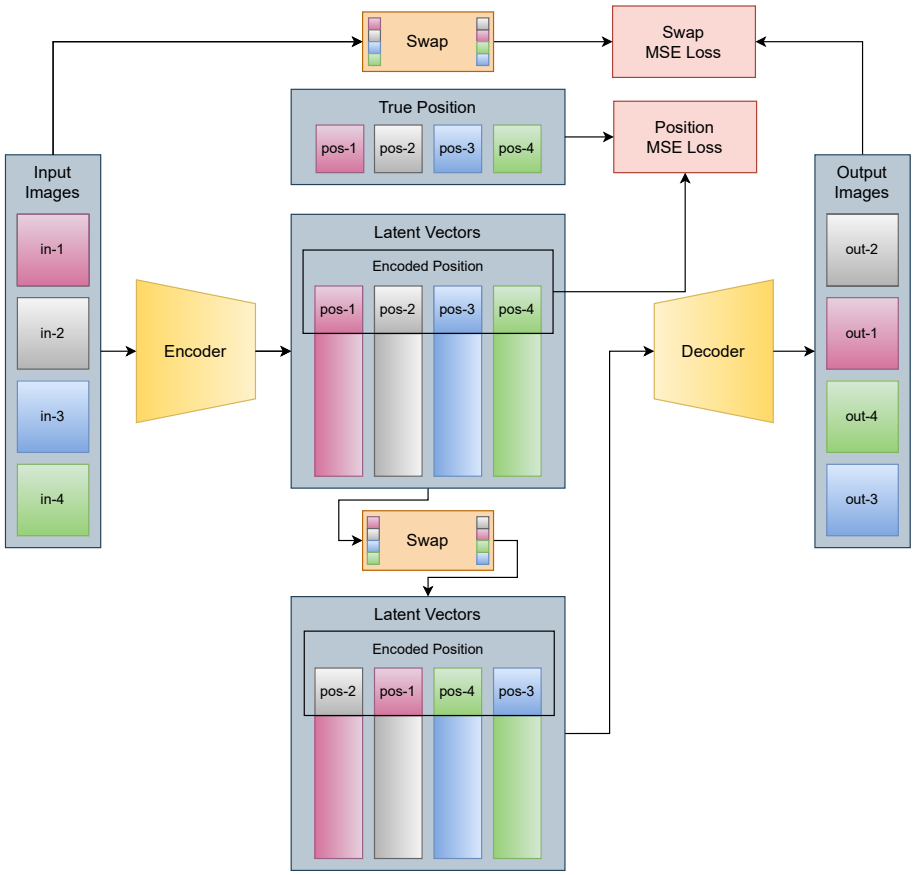


Figure 3.5: Training pipeline for the swap-position-encoding model. As for the reconstruction-position-encoding (RPE) model in Figure 3.3, the batch of input images are encoded into latent vectors, and a position loss is applied to ensure that the first two dimensions of the latent vectors learn to encode position. The difference in this model compared to the RPE model is the pairwise swapping step. In the swapping step, the parts of the latent vectors that encode position are swapped between vector 1 and 2 and between vector 3 and 4, into modified latent vectors. The input images are reordered in a similar fashion before they are used as labels to an MSE loss that we call the *swap* loss. The swap loss is computed between the decoded modified latent vectors and the reordered input images. The model utilizes the encoder and decoder from Figure 3.1.

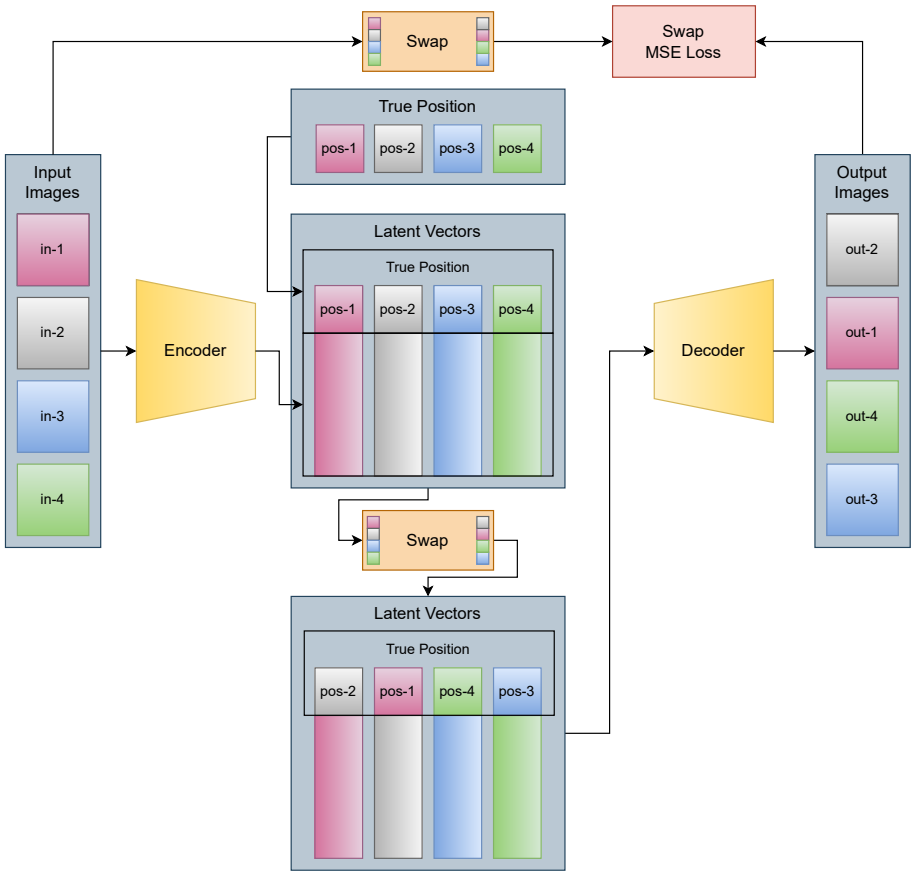


Figure 3.6: Training pipeline for the swap-direct model. As for the swap-position-encoding (SPE) model in Figure 3.5, the batch of input images is encoded into latent vectors, and the first two dimensions of the latent vectors are swapped pairwise. Also, the input images are reordered with a pairwise swap before being used as labels for the swap loss. The difference in this model compared to the SPE model is that the true position is injected straight into the first two dimensions of the latent vectors, and no position loss is necessary. The model utilizes the encoder and decoder from Figure 3.1.

3.2 Results and Discussion

This section presents and discusses the results from the experiments with the Reconstruction model, RPE model, SPE model, and the SD model. The general trend is that higher latent dimensionality improves the quality of the output images. However, at some point, the improvement becomes no longer significant. The most important finding is that the swapping mechanism in the SPE model and the SD model allows for controlling the position with any of the investigated latent dimensionalities.

3.2.1 Reconstruction Model

Figure 3.7 shows the reconstruction losses from the best training epoch, for all latent dimensionalities in the experiment. Higher latent dimensionality provides better reconstruction, and the difference between dimensionality 2 and 4 is the greatest. Figure 3.8 shows the reconstruction losses for all of the 100 epochs. This figure shows that the validation loss is still improving, although very slowly, indicating that the model is not overfitting to the data. The learning curve is noisier in the validation loss, possibly because the validation set is smaller than the training set.

Figure 3.9 shows three samples of input images from the dataset and reconstructed images using models with latent dimensionalities 2, 4, 8, 16, and 32. The significant improvement between a 2-dimensional and a 4-dimensional latent space is also visible in the samples; however, for higher adjacent dimensionalities, it is harder to notice a difference in the images. However, compared with Figure 3.10, the improvement between a 4-dimensional and a 256-dimensional latent space is clearly visible in the samples.

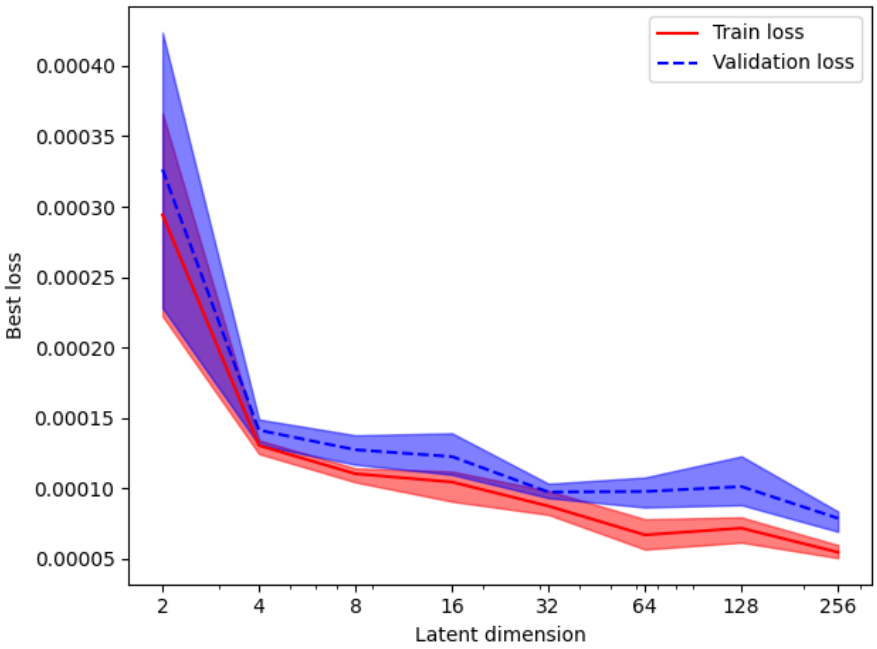


Figure 3.7: Experiment using the position dataset and the reconstruction model, with varying latent dimensionality. The plot shows the *reconstruction* losses from the epoch with the lowest validation loss. The line plots show mean loss, and the shadowed regions span from minimum to maximum loss for 3 different training/validation splits.

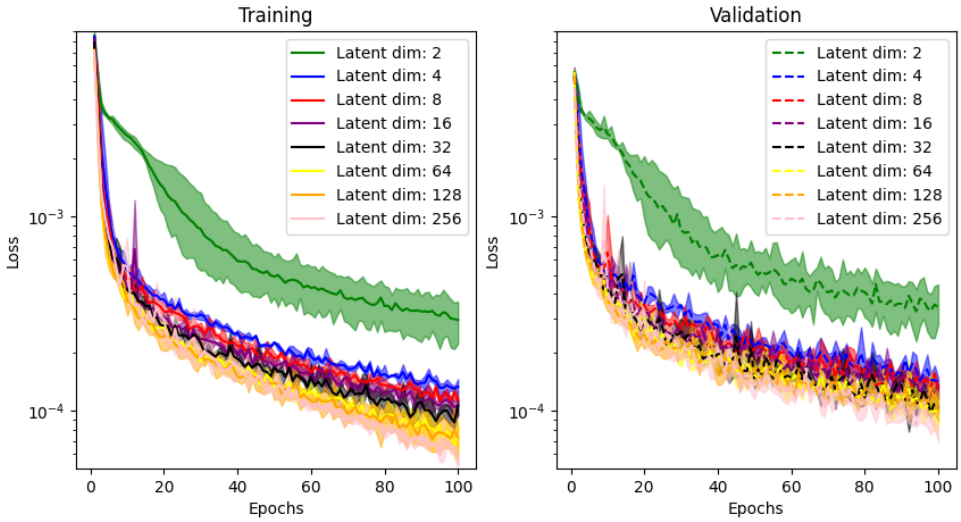


Figure 3.8: Experiment using the position dataset and the reconstruction model showing *reconstruction* losses over 100 epochs for varying latent dimensionality. The line plots show mean loss, and the shadowed regions span from minimum to maximum loss for 3 different train/validation splits.

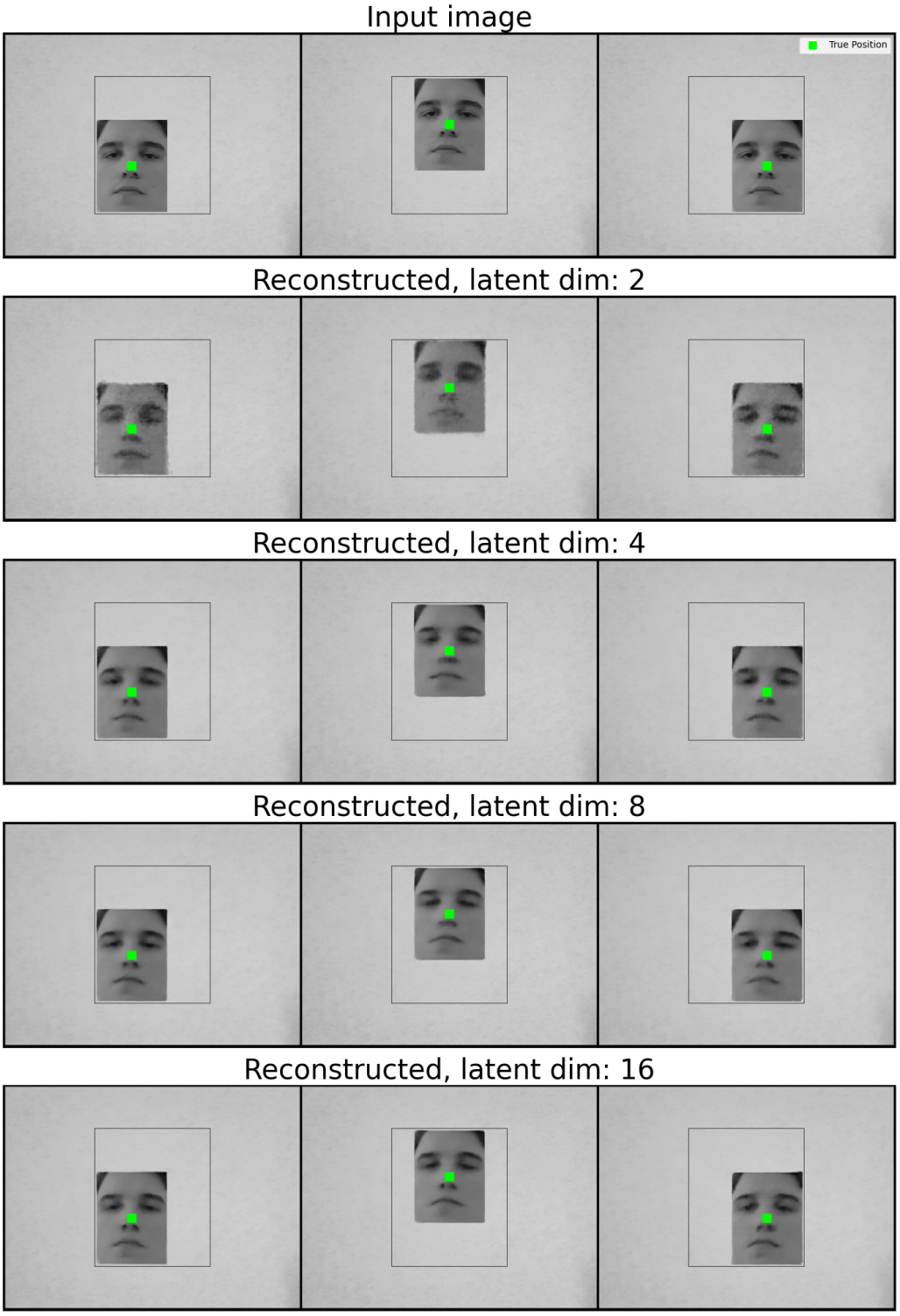


Figure 3.9: Input images from the position dataset and reconstructed images using the reconstruction model with 2, 4, 8, and 16 latent dimensions. The green square marks the center of the face in the input image.

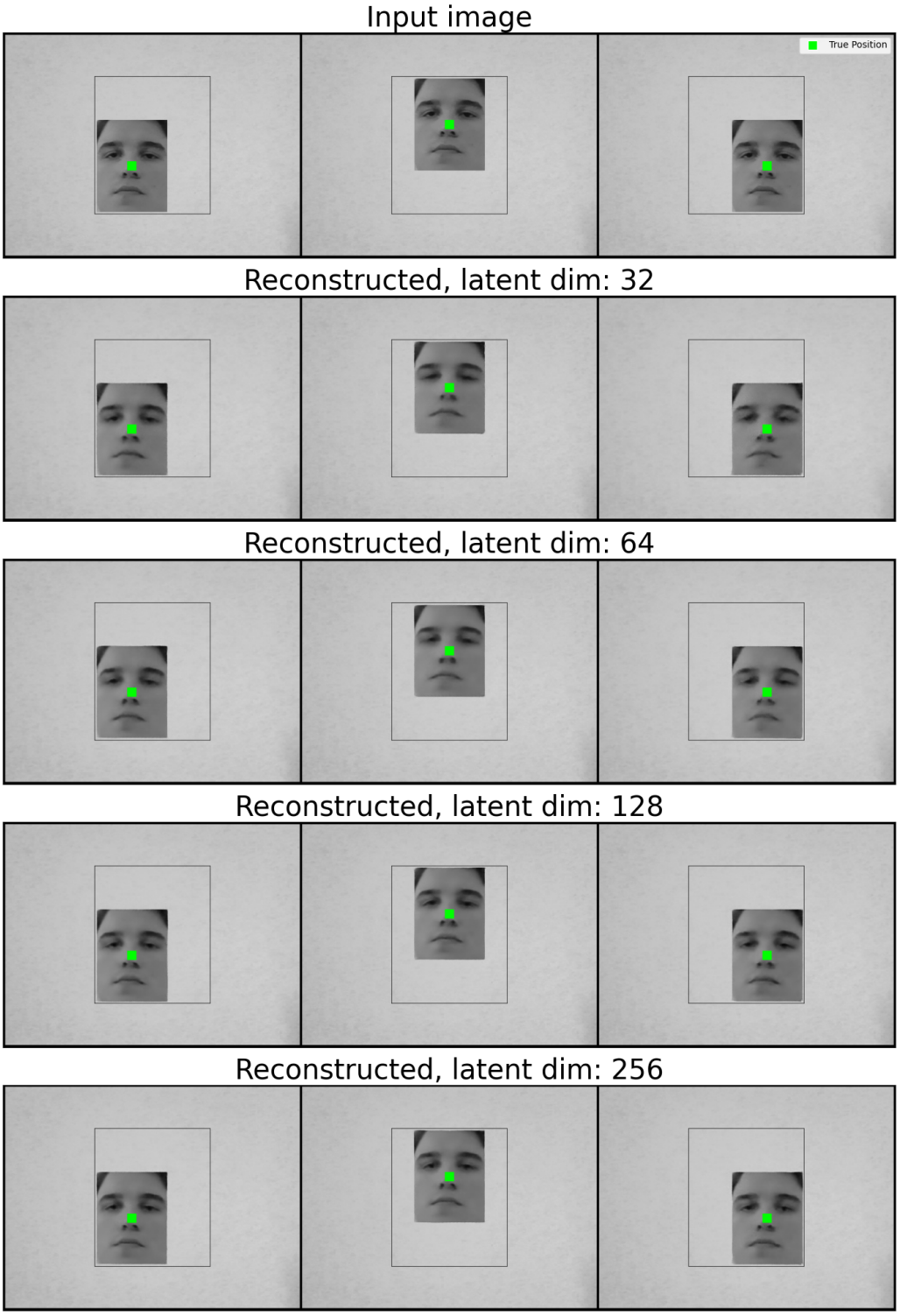
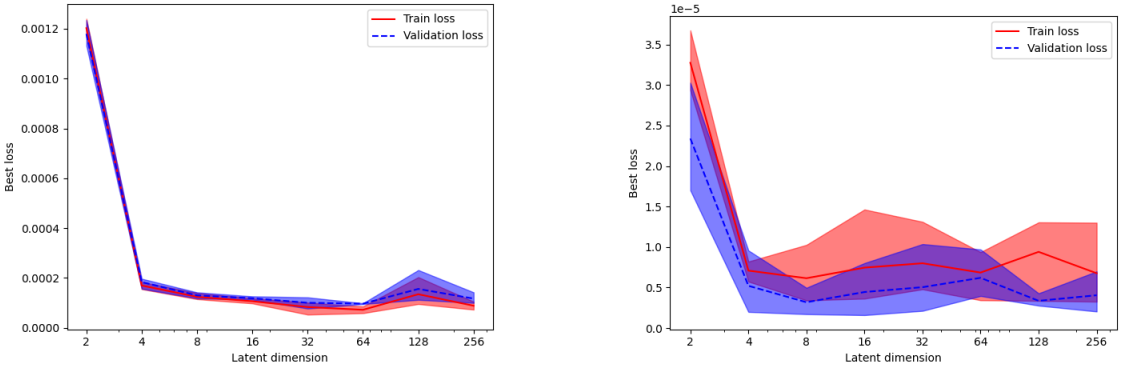


Figure 3.10: Input images from the position dataset and reconstructed images using the reconstruction model with 32, 64, 128, and 256 latent dimensions. The green square marks the center of the face in the input image.

3.2.2 Reconstruction-Position-Encoding (RPE) Model

Figure 3.11 shows the reconstruction and position losses for the best epoch, for each latent dimension. The results show that the most significant improvement in both reconstruction and position loss is between 2 and 4 latent dimensions. When increasing to 4 dimensions and beyond, both losses barely change.



(a) *Reconstruction* loss.

(b) *Position* loss.

Figure 3.11: Experiment with the position dataset and the reconstruction-position-encoding model. For varying latent dimension, the plot is showing the *reconstruction* (a) and *position* (b) losses from the epochs with the lowest validation losses. The bold line plots show mean loss and the shadowed regions span from minimum to maximum loss for 3 different train/validation splits.

Figure 3.14 shows samples of input images and reconstructed images for 2, 4, 8, and 16 latent dimensions. The reconstructed images for dimensions 32, 64, 128, and 256 are found in Appendix A in Figure A.1. The difference in reconstruction loss between 2 dimensions and higher-dimensional settings is even more pronounced than in the R-model, with the 2-dimensional result appearing significantly blurrier. However, the reconstructed faces are still centered very close to their original position. The difference in position loss between 2 and higher dimensions is also visible as the encoded position square (red) is further away from the true position square (green), although the position encoding is decent for all dimensions.

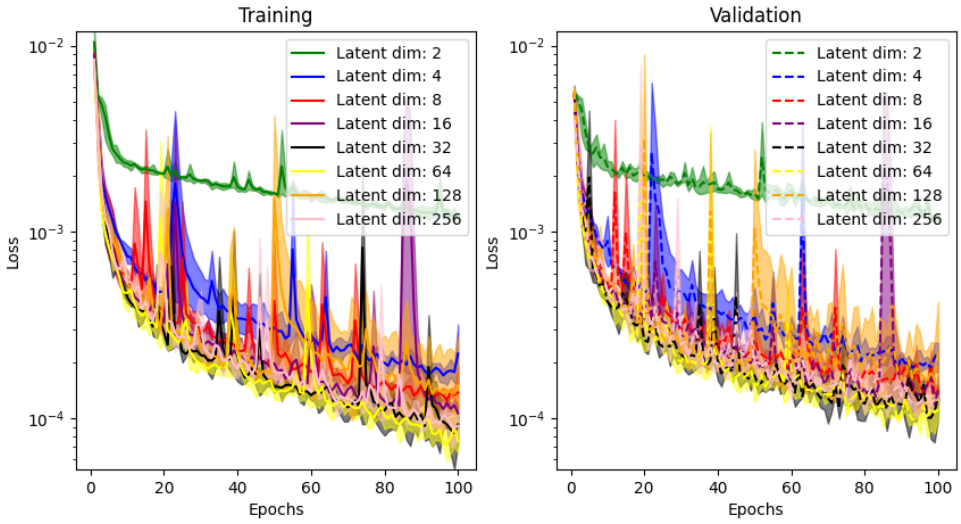
Figure 3.15 shows samples of input images and reconstructed images from modified latent vectors for 2, 4, 8, and 16 latent dimensions. The reconstructed images for dimensions 32, 64, 128, and 256 are found in Appendix A in Figure A.1. When modifying the position part of the latent vectors from the position marked in red to the position marked in blue, the image moves in 2 dimensions, but nothing visible happens for higher dimensions. This means that generating images at an arbitrary position works for 2 latent dimensions but not for higher dimensions.

We have seen in figure 3.14 that the model can reconstruct faces centered very close to the the true position. For two dimensions, the output image is generated directly from the encoded position. We have also seen that the encoded position is very close

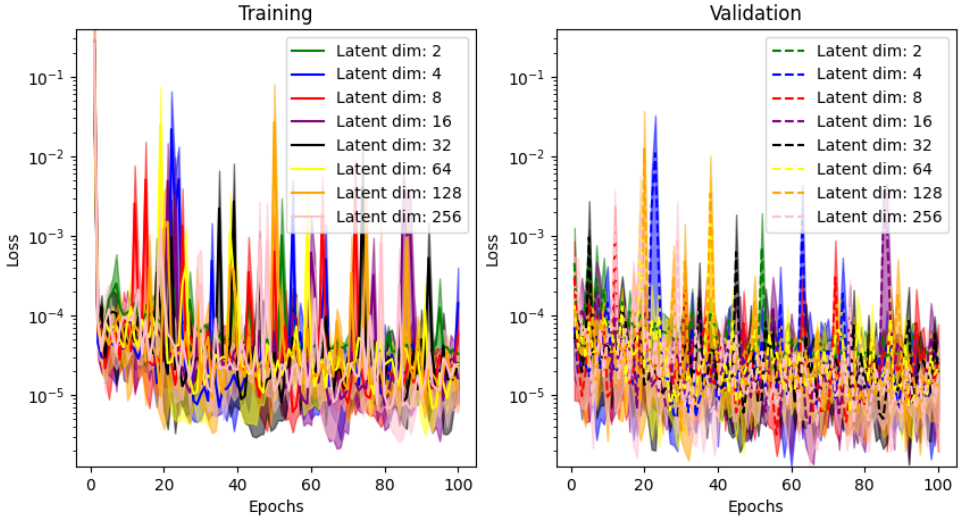
to the true position. The validation loss is also low for faces with positions inside the bounding box, showing that the decoder can interpolate and generate images from unseen position encodings. Therefore, it is possible to generate an output image very close to the input image by passing a novel position inside the bounding box to the decoder.

For more than two dimensions, the output is not generated directly from the encoded position, since the decoder also takes the remaining part of the latent vector as input. In the previous setup, the decoder received input either from the encoder it was jointly trained with or from positions within the bounding box, in which it had demonstrated the ability to generalize. However, when modifying the position part of the larger latent vector, the decoder will receive input from a latent vector outside of the manifold that it was taught to generalize over.

Looking at the losses over epochs in Figure 3.12, we see that there are large spikes in both losses where the model becomes a lot worse for one epoch. The spikes are correlated between the reconstruction and the position loss, for example, there are blue and orange spikes next to each other at approximately epoch 20 for both validation losses. To investigate what causes the large spikes, we train a model on position only, and the loss curves from three different runs are shown in Figure 3.13. These loss curves, when isolating position encoding, also contain large spikes, although not as large as those observed in the experiments with the RPE model. Looking back at the loss curves from the R-model in Figure 3.8 with reconstruction isolated, there were no large spikes, indicating that the irregularities in the position loss might be causing the spikes in Figure 3.13. A possible reason for why the spikes might be even larger when training with both losses is that when the position loss spikes high after a gradient update, the reconstruction performance might be thrown far off since it uses the encoded position, which could cause the position loss to spike even higher until both losses start decreasing again.



(a) *Reconstruction* loss.



(b) *Position* loss.

Figure 3.12: Experiment with the position dataset and the reconstruction-position-encoding model showing (a) *reconstruction* and (b) *position* losses over 100 epochs for varying latent dimension. The bold line plots show mean loss and the shadowed regions span from minimum to maximum loss for 3 different train/validation splits.

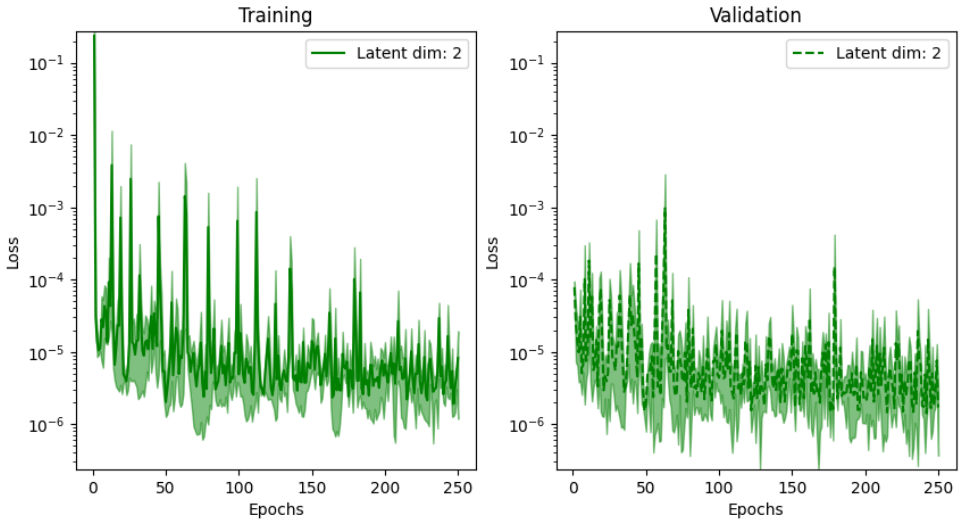


Figure 3.13: Experiment with the position dataset and the reconstruction-position-encoding showing *position* loss over 250 epochs. The bold line plots show mean loss and the shadowed regions span from minimum to maximum loss for 3 different train/validation splits.

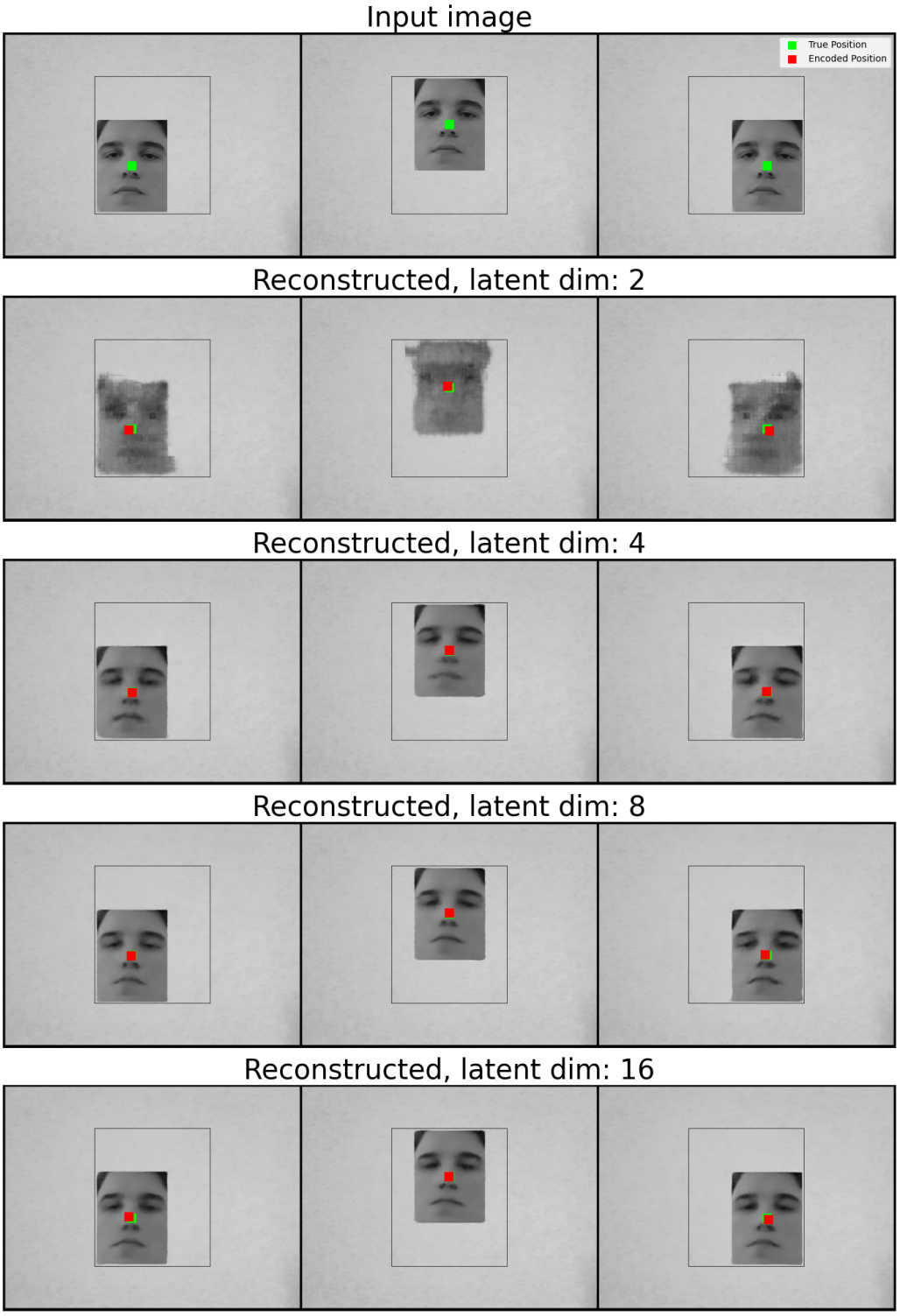


Figure 3.14: Input images from the position dataset and reconstructed images using the reconstruction-position-encoding model with 2, 4, 8, and 16 latent dimensions. The green square marks the true center and the red square marks the encoded center of the face in the input image.

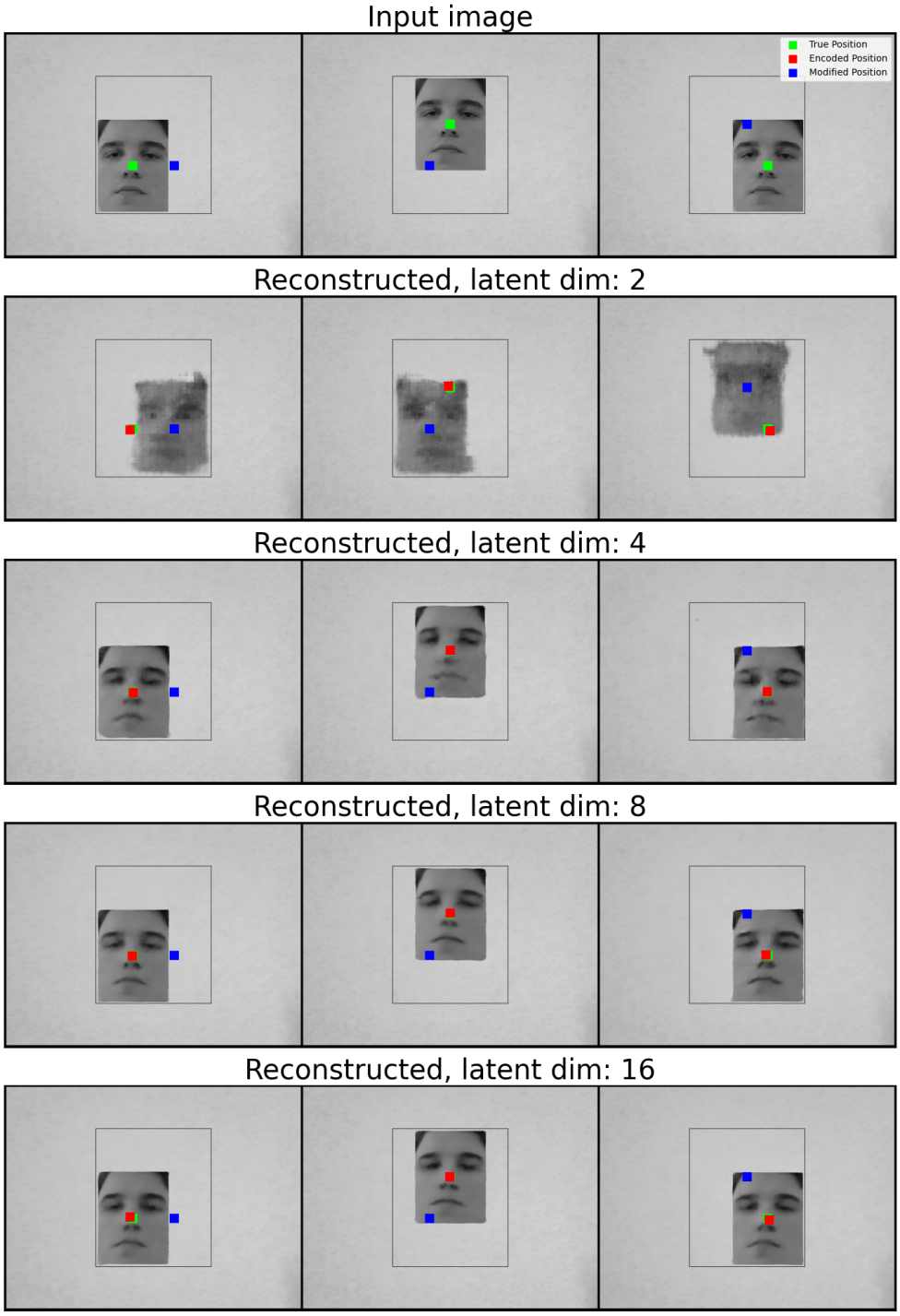
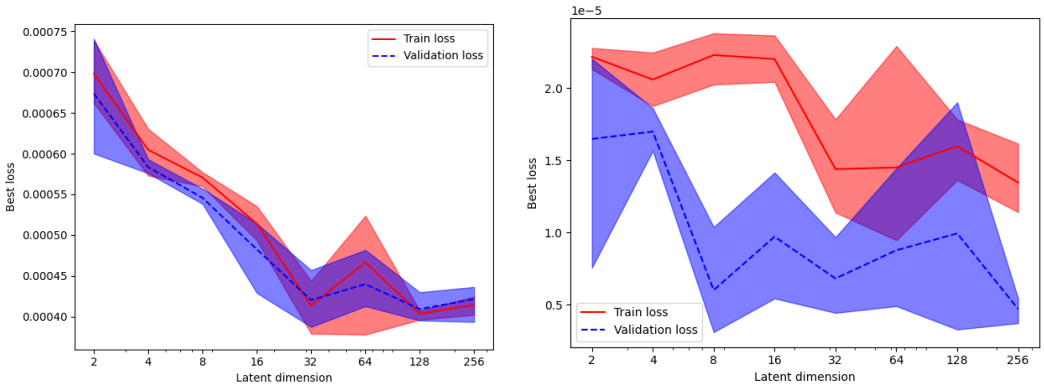


Figure 3.15: Input images from the position dataset and reconstructed images from a modified latent vector using the reconstruction-position-encoding model with 2, 4, 8 and 16 latent dimensions. The green square marks the true center and the red square marks the encoded center of the face in the input image. The objective is to move the images from the red square to the blue square by updating the first 2 dimensions of the latent vector representing position (x, y).

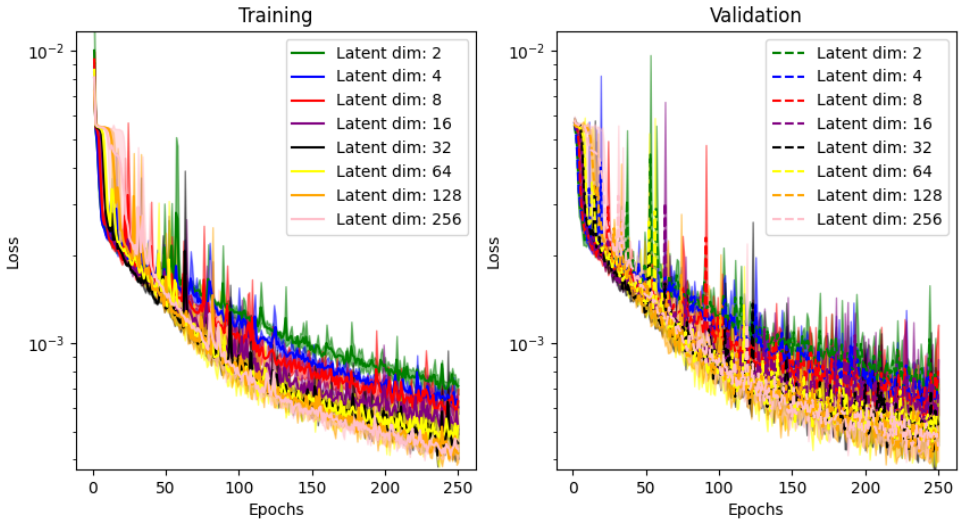
3.2.3 Swap-Position-Encoding (SPE) Model

As seen in Figure 3.16, the swap loss decreases up to 32 dimensions, resulting in output images of increasing quality. Above 32 dimensions, the swap loss does not decrease further. An explanation for this could be that the encoded position information gets lost in the ever-larger latent space. Furthermore, we can observe in Figure 3.16a and Figure 3.17a, that the improvement between 2 and 4 latent dimensions is less dramatic compared to the RPE model. The change in the swap loss between dimensions seems to be multiplicative for each doubling in latent dimensionality. In contrast, the position loss quickly reaches low values and remains relatively flat on average when changing the latent dimension, as shown in Figure 3.17b, but exhibits problematic spikes throughout training. These spikes seem to spread to the swap loss as seen in the comparison in Figure 3.17, as in Section 3.1.4.

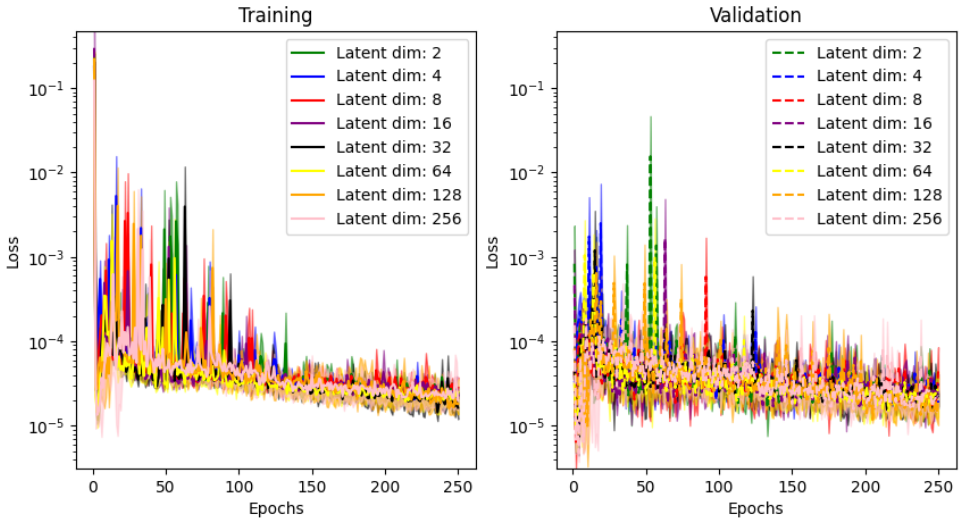


(a) *Swap* loss. **(b)** *Position* loss.

Figure 3.16: Experiment with the position dataset and the swap-position-encoding model. For varying latent dimensionalities, the plot shows the *swap* (a) and *position* (b) losses from the epochs with the lowest validation losses. The line plots show mean loss, and the shadowed regions span from minimum to maximum loss for 3 different train/validation splits.



(a) *Swap* loss.



(b) *Position* loss.

Figure 3.17: Experiment with the position dataset and the swap-position-encoding model showing (a) *swap* and (b) *position* losses over 250 epochs for varying latent dimensionality. The line plots show mean loss, and the shadowed regions span from minimum to maximum loss for 3 different train/validation splits.

The qualitative results of the SPE model are shown in Figure 3.18, showing a steady improvement when increasing the latent dimensionality. The figure also shows that the movement of the face when swapping is effective, but the image quality is inferior to that of the Reconstruction model and the RPE model. The output images for 32, 64, 128, and 256 dimensions are found in Appendix A in Figure A.3.

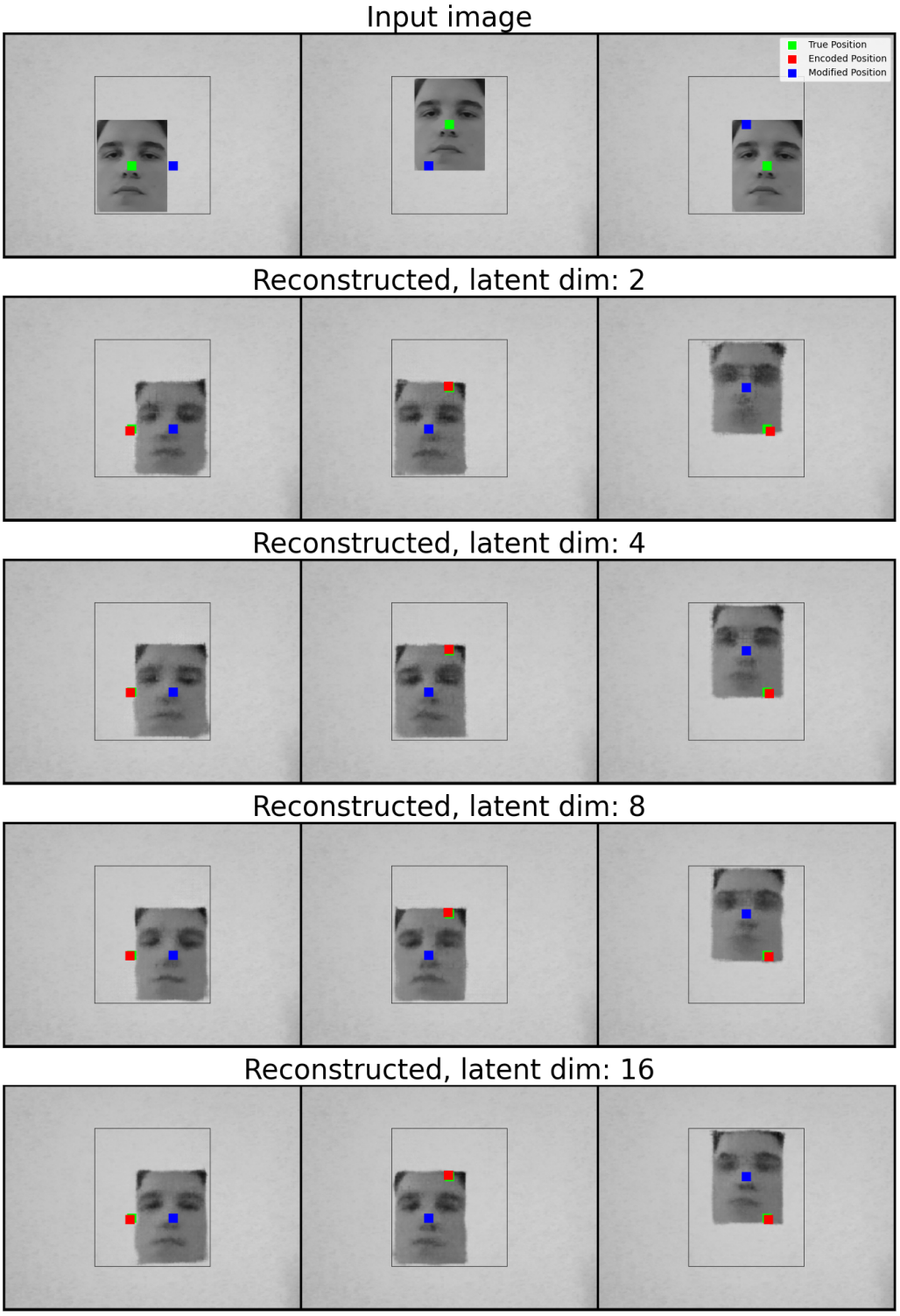


Figure 3.18: Input images from the position dataset and reconstructed images from a modified latent vector using the swap-position-encoding model with 2, 4, 8, and 16 latent dimensions. The green square marks the true center, and the red square marks the encoded center of the face in the input image. The objective is to move the images from the red square to the blue square by updating the first 2 dimensions of the latent vector representing position (x, y).

3.2.4 Swap-Direct (SD) Model

Figure 3.19 shows the swap losses from the best training epoch for all latent dimensionalities in the experiment. The loss curve is very similar to the one for the SPE model, in that the loss stops decreasing after 32 latent dimensions. The swap loss for all latent dimensionalities is slightly lower compared to the results from the SPE model in Figure 3.16. The improvement might be caused by a more stable loss curve over the training as seen in Figure 3.20, with smaller spikes compared to the losses from the SPE model in Figure 3.17. Removing the position loss, that probably caused the spikes as proposed in Section 3.1.4, might be the reason for the more stable loss curve. The slight improvement in loss achieved by the SPE model is not visually perceptible in the generated images. Sample outputs are provided in Appendix A, Figures A.4 and A.5.

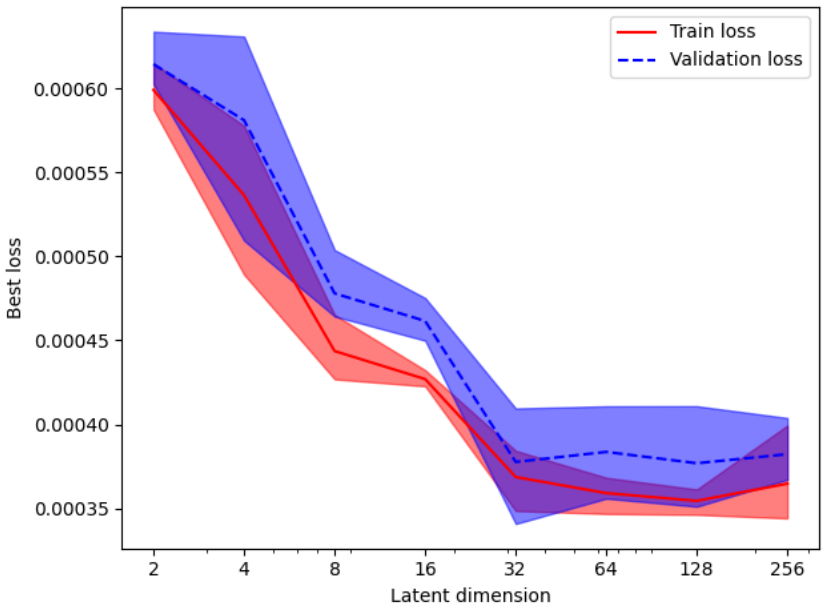


Figure 3.19: Experiment with the position dataset and the swap-direct model. For varying latent dimensionality, the plot is showing the *swap* loss from the epoch with the lowest validation loss. The bold line plots show mean loss and the shadowed regions span from minimum to maximum loss for 3 different train/validation splits.

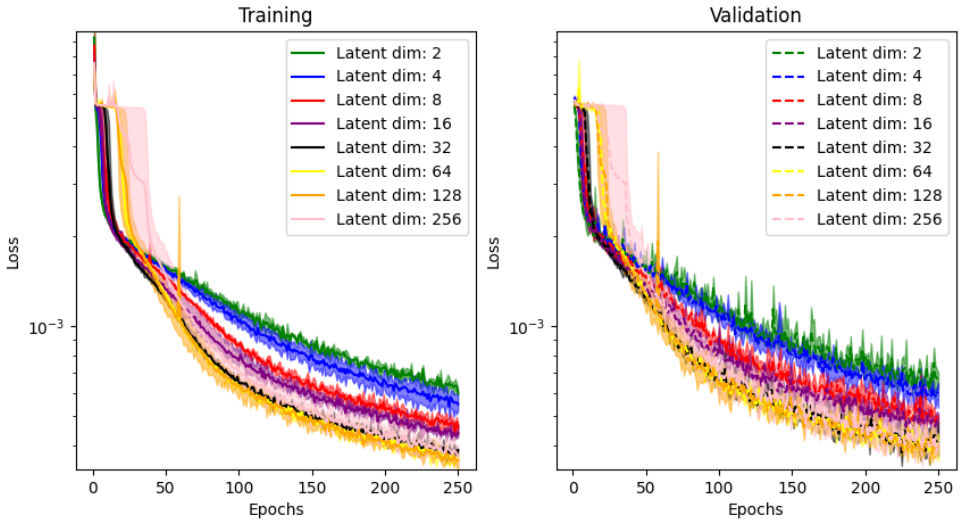


Figure 3.20: Experiment with the position dataset and the swap-direct model showing losses over 250 epochs for varying latent dimension. The bold line plots show mean loss and the shadowed regions span from minimum to maximum loss for 3 different train/validation splits.

3.2.5 Summary

This chapter investigated techniques for structured control of the latent space of autoencoders, specifically steering the position of faces. Four models were examined:

- **The Reconstruction Model**: This simple model achieved near-perfect reconstruction quality for higher latent dimensions, although it did not allow for control of the latent space.
- **The Reconstruction-Position-Encoding Model**: This model combined reconstruction with encoding of the position in the first two latent dimensions. Controlling position was only possible with a two dimensional latent space, and the position loss caused instability during training.
- **The Swap-Position-Encoding Model**: Building on the RPE approach, this model introduced pairwise swapping of latent vectors. Key findings include:
  - successful position control for any dimensionality of the latent space,
  - performance improvements plateauing after 32 dimensions,
  - difficulty achieving high-quality outputs.
- **The Swap-Direct Model**: In this model, the true position was injected directly into the latent space rather than encoding it, resulting in:
  - slightly better performance than SPE,
  - more stable loss curves during training,
  - similar performance plateau after 32 dimensions.

Our experiments demonstrated that controlling position through latent space modification is feasible, and that the swapping mechanism is crucial for controlling position for a latent dimension higher than 2. A consistent challenge was the unstable training behavior when combining multiple losses, likely caused by an unstable position loss. Furthermore, achieving high-quality outputs remained a challenge when

moving away from the simple reconstruction model.

4

Controlling the Rotation of a Realistic Face

The experiments with the cut-out face concluded that it is possible to learn how to generate faces at new positions, however, with imperfect details. In this section, the same core idea is applied, but using real images with different head rotations, rather than positions. The aim is to investigate how well the idea of swapping parts of the latent vectors works for a more realistic problem.

4.1 Rotation Dataset

The dataset consists of images from a video recording in which the subject attempts to rotate their head while maintaining a static body and facial expression. The dataset contains 5000 training images and 5000 test images, each with a resolution of 640 $\times$ 480 pixels. Samples from the rotation dataset are shown in Figure 4.1. Even with a large dataset, it is impossible to capture all possible head rotations, and two images with the same rotation could contain slightly different facial expressions.

Head rotation labels, pitch, yaw, and roll, are computed using facial landmarks from the framework MediaPipe [5]. The marginal distributions for pitch, yaw, and roll are shown in Figure 4.2. These distributions are typically centered around 0 degrees, since the subject was facing directly at the camera in more frames than any other direction.

Figure 4.1: Sample from the rotation dataset. The rotation labels for each image, computed using the framework MediaPipe [5], is displayed in green.

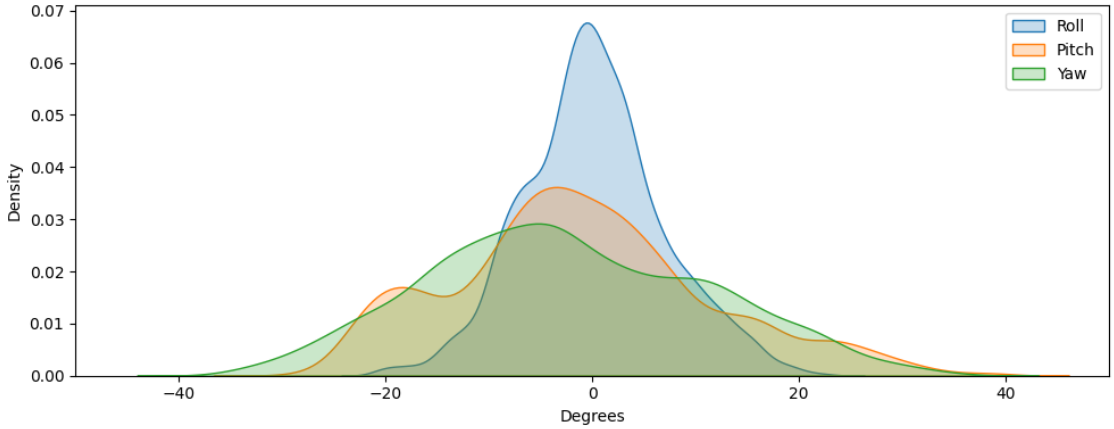


Figure 4.2: Marginal distributions of pitch, yaw, and roll of the rotation dataset.

4.2 Method

With a new model, referred to as the rotation model, the aim is to generate images that resemble the ones from the rotation dataset, but with unseen head rotations. The training pipeline for the rotation model is the same as for the SPE-model in Figure 3.5, but instead of encoding positional coordinates (x, y) in two dimensions, rotation in terms of pitch, yaw, and roll is encoded and swapped in three dimensions of the latent vectors. Like the models discussed above, the rotation model utilizes the encoder and decoder architectures described in figure 3.1. As for the latent dimension experiment (Section 3.1.2), we use a batch size of four images and the ADAM [12] optimizer with fixed learning rate 0.0005. The size of the latent representations is 8 dimensions, including three dimensions encoding rotation, and the model is trained for 1000 epochs, which took 2 days on 1 NVIDIA RTX4000 GPU.

In the inference stage, a source image from the rotation test dataset is encoded into a latent vector, and driving rotations are extracted from a set of 250 unseen images of an arbitrary face using the facial landmarks from MediaPipe. For each driving rotation, the encoded rotation in the latent vector is replaced by the driving rotation and the updated latent vector is decoded into an output image. The inference results from our rotation model are evaluated both qualitatively and quantitatively against a state of the art model called LivePortrait [6]. The inference time for the task was a few seconds for the rotation model, compared to a few minutes for LivePortrait.

LivePortrait is a portrait-animation framework whose objective is to bring portraits to life by transferring head pose, eye-gaze and facial expressions from a driver video onto a source portrait. It works by first extracting key facial landmarks and motion features from the driver video, then using deep neural networks to map these dynamic cues onto the static portrait. Through this process, the system synthesizes realistic animations that preserve the identity of the source portrait while mimicking the expressions and movements of the driver. LivePortrait was trained on 69 million

frames for 12 days in total on 8 NVIDIA A100 GPUs. [6]

4.2.1 Results and Discussion

Overall, the inference results show that the rotation of the output images follows the driving rotation well, however with some compromises in sharpness in the facial region. Figure 4.3 shows some of our output images, along with the images generated by LivePortrait. It is clearly visible for both models that the head rotation from the source images has adapted closely to the driving rotations.

Compared to LivePortrait, the rotation model generates blurry details within the face. The blurriness could be caused by imperfections in the model or a lack of training, as in the position control section. However, the rotation dataset differs due to variability in the images, such as varying facial expressions, changes in eye gaze direction, and small body movements, that the rotation labels cannot explain. This means that, in theory, there may be two images in the dataset with the same rotation label that appear slightly different, making it ambiguous for the decoder to determine which image to generate for that rotation. Therefore, the rotation model has a theoretical limit in performance, and a possible solution could be to also encode the eye gaze direction and swap those together with the rotation. Since the DMS tracking algorithms focus on the driver's eye gaze direction, such a model would hopefully suffice for the problem.

Figure 4.4 shows the MediaPipe tracking results from the driving inputs, the rotation model, and LivePortrait's outputs. The tracking results agree with the visual results since the rotation trajectories from the rotation model and LivePortrait reflect the driving rotation trajectory closely. Comparing the models, the rotation model has the advantage of being a lot leaner with a significant reduction in inference speed, training time and resources. The rotation model is also directly steerable using rotation inputs, as opposed to LivePortrait's required driving videos. For both the rotation model and LivePortrait, the tracking is most accurate for roll, followed by pitch and yaw. The fact that the tracking works even though there are defects in the face shows that the MediaPipe tracking algorithm is relatively robust to noise. Finally, the results conclude that the synthetic videos are good enough to be compatible with head rotation tracking, which is promising for the application of DMS validation.

Figure 4.3: From left to right: source images, driving rotations/images, outputs from the rotation model, and lastly outputs from LivePortrait. The source images are sampled from the rotation dataset (Section 4.1). For the rotation model, the objective is to transfer the driving rotation to the source image. The output of the model is then compared in the last column to the output of the LivePortrait model, which transfers holistic facial information from the driving image rather than just the rotation.

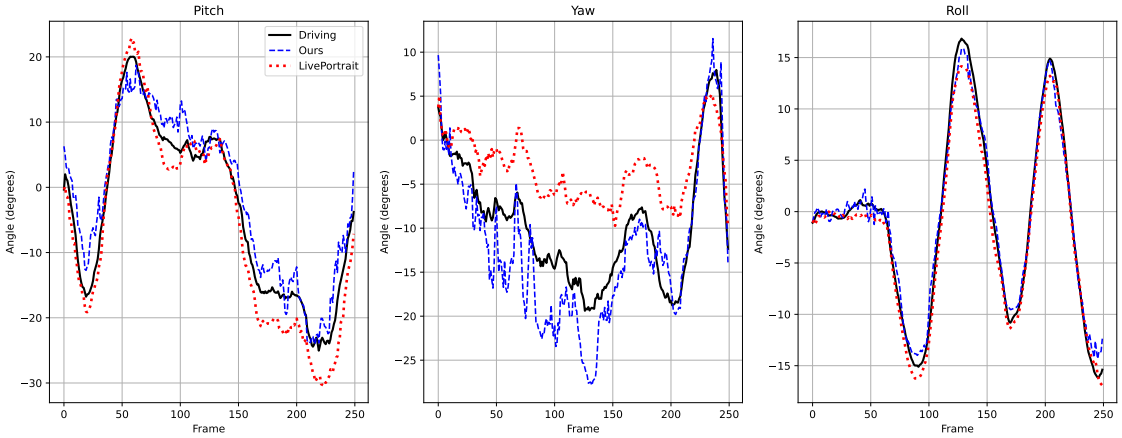


Figure 4.4: Tracking results of pitch, yaw, and roll using the MediaPipe framework [5], on driving input images, and fake images generated by the rotation model and LivePortrait [6].

5
Discussion and Future Work

It has been demonstrated that the rotation model is capable of generating synthetic videos with controlled head rotation, which appear realistic enough to be compatible with MediaPipe tracking algorithms. However, the rotation model in its current state is not yet ready for generating synthetic DMS videos, which are required for validating the DMS. To solve this, the rotation model needs to be trained on real DMS videos to learn to generate videos from that distribution. Given that the model can interpolate to unseen head rotations, it is possible to generate significantly more synthetic data than the size of the dataset, allowing for the customization of DMS scenarios for validation.

A typical validation scenario could be to let a synthetic driver move the head in a target trajectory from -45 degrees to $+45$ degrees yaw, and compare the output rotation sequence from the DMS to the target trajectory. While this comparison would serve as a form of validation for the DMS, there is a concern that the model may be biased due to the MediaPipe tracker used during training. However, the method could still be beneficial for testing how different versions of a DMS compare against each other in relation to the target trajectory. Such tests could help support claims stating that a new DMS version has improved by a given amount compared to a previous version. Other cases for validation could involve a scenario in which a car overtakes a long-haul truck on a snowy road. Assume that in the original recording, the driver remains focused and no issues arise. By applying the rotation model, the driver's behavior can be synthetically altered to simulate inattention, for example, by making the driver look out the side window for too long during the passing. This modified data can then be processed through the DMS to evaluate its response and assess the system's effectiveness in detecting distracted driving in potentially dangerous situations.

Another contribution to the project is demonstrating that LivePortrait is capable of generating synthetic videos that are compatible with tracking algorithms. Since LivePortrait is trained on a wide range of videos, it can generalize well enough to generate DMS videos. However, for validation of DMS, LivePortrait's videos also need to be compatible with the tracking algorithms used in a real DMS, which is a topic for future work.

Future work should primarily focus on refining the rotation model to achieve improved image quality, stabilizing the training process, and extending these techniques to control additional image attributes, particularly eye gaze direction, thereby en-

abling a more comprehensive validation of DMS. Training the model on real frames from a DMS is essential for its usefulness. Another approach is to modify LivePortrait to make it directly steerable, omitting the need for a driving video. Finally, it would be interesting to evaluate the synthetic videos using head rotation tracking algorithms from a real driver monitoring system, and test the DMS on typical validation scenarios.

6
Conclusion

The contributions are twofold. Firstly, a relatively light-weight autoencoder for directly controlling the head position of a face is developed through experiments with different model setups and parameter choices. It has been demonstrated that swapping parts of the latent space of an autoencoder enables the control of the output using a subset of the latent variables in a high-dimensional latent space. The swapping delivers the output quality of a high-dimensional latent space, with the controllability of a low-dimensional latent space. During experiments with the architecture and parameter choices of the autoencoder, it was concluded that a high-dimensional latent space yields better output quality, but only up to a certain point. Since adding dimensions in the latent space introduces more parameters to the model, resulting in longer training and inference times, there is a tradeoff when choosing the latent dimensionality. It was also discovered that a spiky loss, such as the position loss, can transfer its undesirable characteristics to other losses when they are summed into a shared learning objective. These findings contribute to our understanding of disentangled representations in autoencoders and highlight important trade-offs between control precision, reconstruction quality, and training stability.

Secondly, the rotation model is utilized to generate synthetic videos with control of head rotation, building on insights from the position control experiments, particularly the swapping method. The model is capable of generating frames in which the head appears to rotate in accordance with the specified driving rotations, although the facial details in the output frames tend to be blurry. When applying tracking algorithms to the synthetic frames, the estimated rotations closely resemble the driving rotations. For evaluation, the performance of the rotation model is compared to LivePortrait, which serves as a benchmark due to its status as a state-of-the-art method for facial reenactment. The evaluation shows that the rotation model and LivePortrait perform very similarly when comparing the output rotation sequences with the driving rotation sequence. LivePortrait, however, generates frames of higher quality in the facial details. These results indicate that synthetic frames produced by both the rotation model and LivePortrait are compatible with existing open-source head rotation tracking tools, showing progress toward the generation of DMS videos for validation purposes. Compared to LivePortrait, the rotation model offers the advantages of being more lightweight and enabling direct control over head rotation, without requiring a driving video as input.

Bibliography

[1] Y. Albadawi, M. Takruri, and M. Awad. A review of recent developments in driver drowsiness detection systems. *Sensors*, 22(5), 2022. URL: https://www.mdpi.com/1424-8220/22/5/2069, doi:10.3390/s22052069.

[2] D. Ceylan, C.-H. P. Huang, and N. J. Mitra. Pix2video: Video editing using image diffusion, 2023. URL: https://arxiv.org/abs/2303.12688, arXiv:2303.12688.

[3] J. Dahl. *Machine Learning-Based Prediction Models and Threat Detection for Lane-Keeping Assistance*. PhD thesis, Chalmers University of Technology, 2023. Accessed on: 2025-01-29. URL: https://research.chalmers.se/en/publication/535461.

[4] N. C. for Statistics and i. b. Analysis (U.S.). Critical reasons for crashes investigated in the national motor vehicle crash causation survey., 2015.

[5] Google. Mediapipe face landmarker. https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker, 2025. Accessed: 2025-06-09.

[6] J. Guo, D. Zhang, X. Liu, Z. Zhong, Y. Zhang, P. Wan, and D. Zhang. Liveportrait: Efficient portrait animation with stitching and retargeting control, 2025. URL: https://arxiv.org/abs/2407.03168, arXiv:2407.03168.

[7] A. C. Hayley, B. Shiferaw, B. Aitken, F. Vinckenbosch, T. L. Brown, and L. A. Downey. Driver monitoring systems (dms): The future of impaired driving management? *Traffic Injury Prevention*, 22(4):313–317, 2021. PMID: 33829941. arXiv:https://doi.org/10.1080/15389588.2021.1899164, doi:10.1080/15389588.2021.1899164.

[8] T. Höppe, A. Mehrjou, S. Bauer, D. Nielsen, and A. Dittadi. Diffusion models for video prediction and infilling, 2022. URL: https://arxiv.org/abs/2206.07696, arXiv:2206.07696.

[9] H.-B. Kang. Various approaches for driver and driving behavior monitoring: A review. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 616–623, 2013. doi:10.1109/ICCVW.2013.85.

[10] M. Q. Khan and S. Lee. Gaze and eye tracking: Techniques and applications in adas. *Sensors (Basel)*, 19(24):5540, December 2019. Epub ahead of print. URL: https://www.mdpi.com/1424-8220/19/24/5540, doi:10.3390/s19245540.

[11] G. Kim, H. Shim, H. Kim, Y. Choi, J. Kim, and E. Yang. Diffusion video autoencoders: Toward temporally consistent face video editing via disentangled video encoding. In *Proceedings - 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,

pages 6091–6100. IEEE Computer Society, 2023. Publisher Copyright: © 2023 IEEE.; 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023 ; Conference date: 18-06-2023 Through 22-06-2023. doi:10.1109/CVPR52729.2023.00590.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017. URL: https://arxiv.org/abs/1412.6980, arXiv:1412.6980.

[13] X. Li, K.-Y. Lin, M. Meng, X. Li, L. Li, Y. Hong, and J. Chen. A survey of adas perceptions with development in china. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14188–14203, 2022. doi:10.1109/TITS.2022.3149763.

[14] R. Llaneras, J. Salinger, and C. Green. Human factors issues associated with limited ability autonomous driving systems: Drivers' allocation of visual attention to the forward roadway. *Proc., 7th International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design*, pages 92–98, 01 2013.

[15] Y. Ma, S. Zhang, J. Wang, X. Wang, Y. Zhang, and Z. Deng. Dreamtalk: When emotional talking head generation meets diffusion probabilistic models, 2024. URL: https://arxiv.org/abs/2312.09767, arXiv:2312.09767.

[16] V. Melnicuk, S. Birrell, E. Crundall, and P. Jennings. Towards hybrid driver state monitoring: Review, future perspectives and the role of consumer electronics. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 1392–1397, 2016. doi:10.1109/IVS.2016.7535572.

[17] U. Michelucci. An introduction to autoencoders, 2022. URL: https://arxiv.org/abs/2201.03898, arXiv:2201.03898.

[18] National Highway Traffic Safety Administration. Overview of motor vehicle traffic crashes in 2022, 2022. Accessed: 2025-01-29. URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813560.

[19] Advanced driver assistance systems, 2023. Accessed: 2025-01-30. URL: https://injuryfacts.nsc.org/motor-vehicle/occupant-protection/advanced-driver-assistance-systems/data-details/.

[20] I. Rakhmatulin and A. T. Duchowski. Deep neural networks for low-cost eye tracking. *Procedia Computer Science*, 176:685–694, 2020. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020. URL: https://www.sciencedirect.com/science/article/pii/S1877050920319360, doi:10.1016/j.procs.2020.09.041.

[21] T.-C. Wang, A. Mallya, and M.-Y. Liu. One-shot free-view neural talking-head synthesis for video conferencing, 2021. URL: https://arxiv.org/abs/2011.15126, arXiv:2011.15126.

[22] A. P. Wolkow, S. M. W. Rajaratnam, V. Wilkinson, D. Shee, A. Baker, T. Lillington, P. Roest, B. Marx, C. Chew, A. Tucker, S. Haque, A. Schaefer, and M. E. Howard. The impact of heart rate-based drowsiness monitoring on adverse driving events in heavy vehicle drivers under naturalistic conditions. *Sleep Health*, 6(3):366–373, June 2020. Epub 2020 Apr 25. doi:10.1016/j.sleh.2020.03.005.

[23] World Health Organization. Global status report on road safety 2018, 2018. Accessed: 2025-01-29. URL: https://www.who.int/publications/i/item/9789241565684.

[24] S. Xu, G. Chen, Y.-X. Guo, J. Yang, C. Li, Z. Zang, Y. Zhang, X. Tong, and B. Guo. Vasa-1: Lifelike audio-driven talking faces generated in real time, 2024. URL: https://arxiv.org/abs/2404.10667, arXiv:2404.10667.

[25] W. Zhang, X. Cun, X. Wang, Y. Zhang, X. Shen, Y. Guo, Y. Shan, and F. Wang. Sadtalker: Learning realistic 3d motion coefficients for stylized audio-driven single image talking face animation, 2023. URL: https://arxiv.org/abs/2211.12194, arXiv:2211.12194.

[26] P. Zhou, L. Wang, Z. Liu, Y. Hao, P. Hui, S. Tarkoma, and J. Kangasharju. A survey on generative ai and llm for video generation, understanding, and streaming, 2024. URL: https://arxiv.org/abs/2404.16038, arXiv:2404.16038.

A

Appendix 1

Figure A.1, A.2, A.3, A.4, and A.5 are additional results from the models for position control.

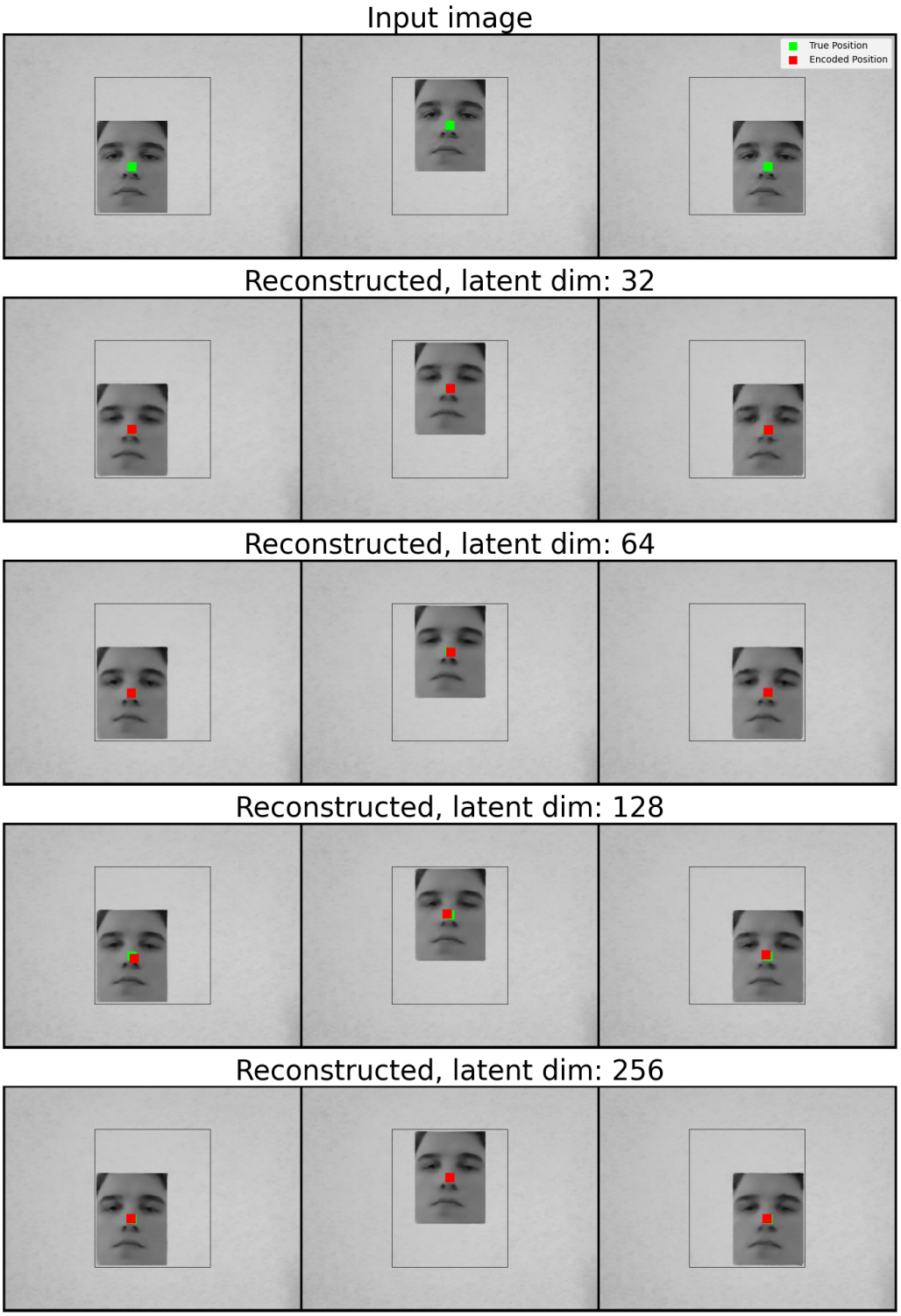


Figure A.1: Input images from the position dataset and reconstructed images using the reconstruction-position-encoding model with 32, 64, 128, and 256 latent dimensions. The green square marks the true center and the red square marks the encoded center of the face in the input image.

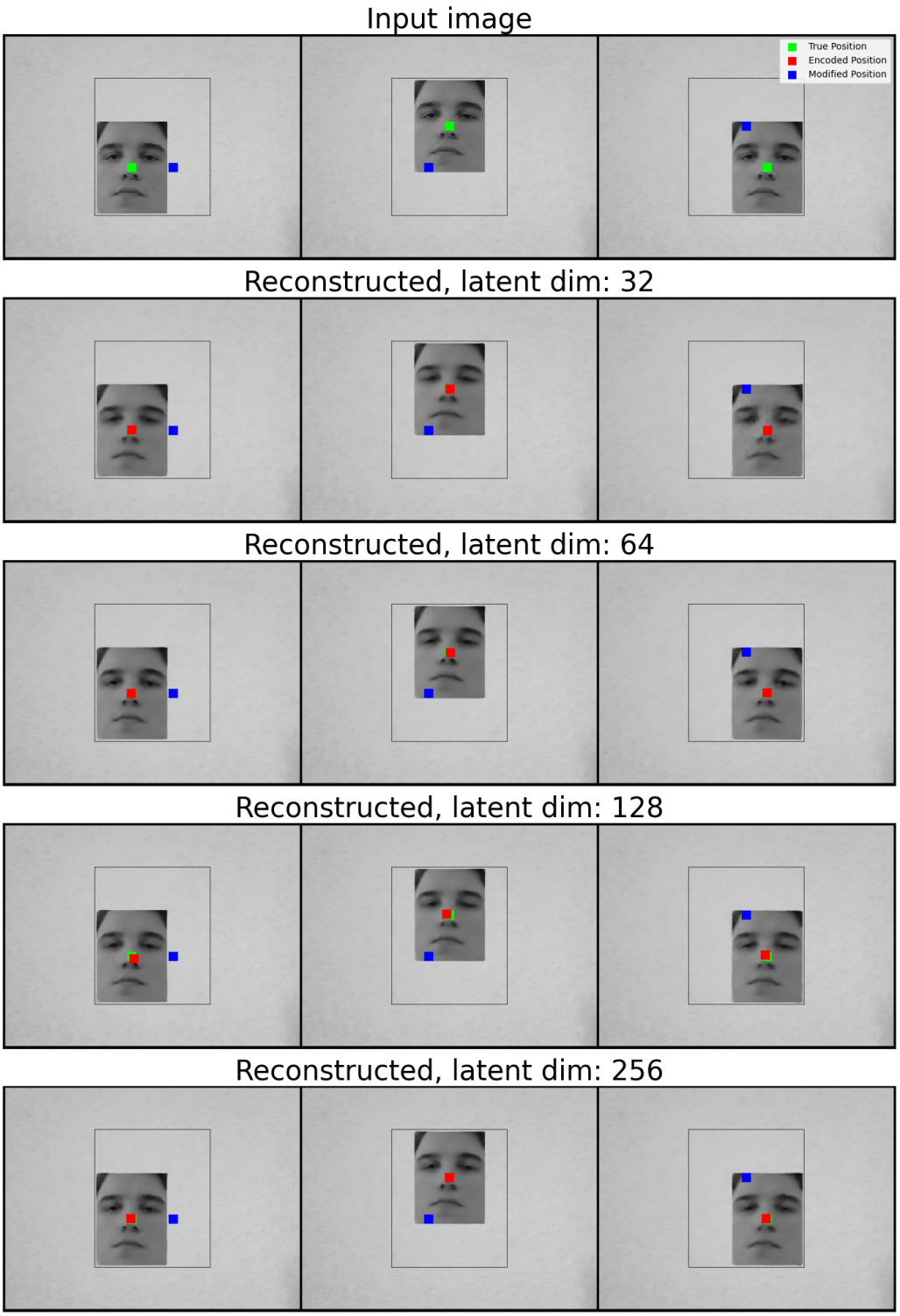


Figure A.2: Input images from the position dataset and reconstructed images using the reconstruction-position-encoding model with 32, 64, 128, and 256 latent dimensions. The green square marks the true center and the red square marks the encoded center of the face in the input image. The objective is to move the images from the red square to the blue square by updating the first 2 dimensions of the latent vector representing position (x, y).

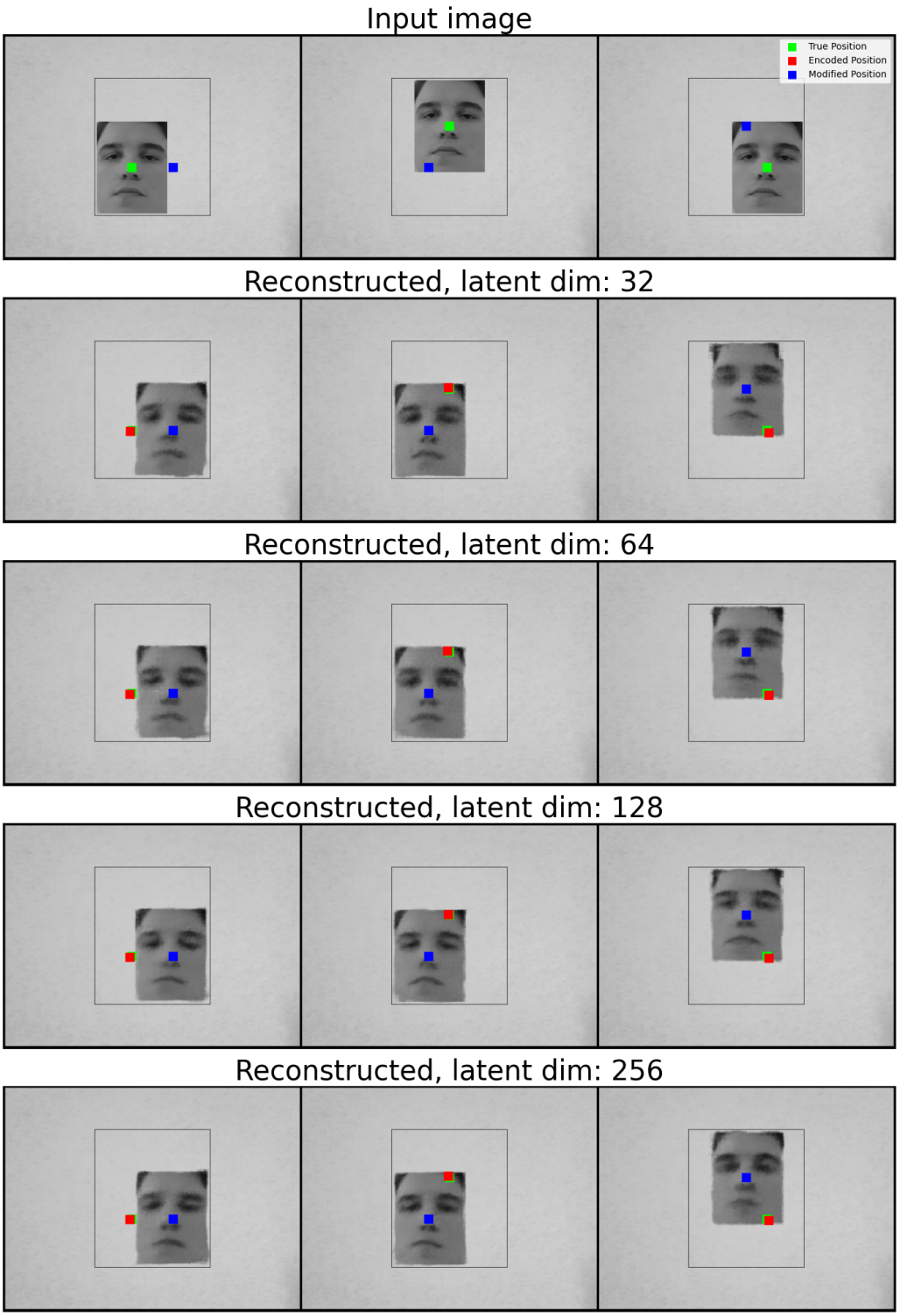


Figure A.3: Input images from the position dataset and reconstructed images from a modified latent vector using the swap-position-encoding with 32, 64, 128, and 256 latent dimensions. The green square marks the true center and the red square marks the encoded center of the face in the input image. The objective is to move the images from the red square to the blue square by updating the first 2 dimensions of the latent vector representing position (x, y).

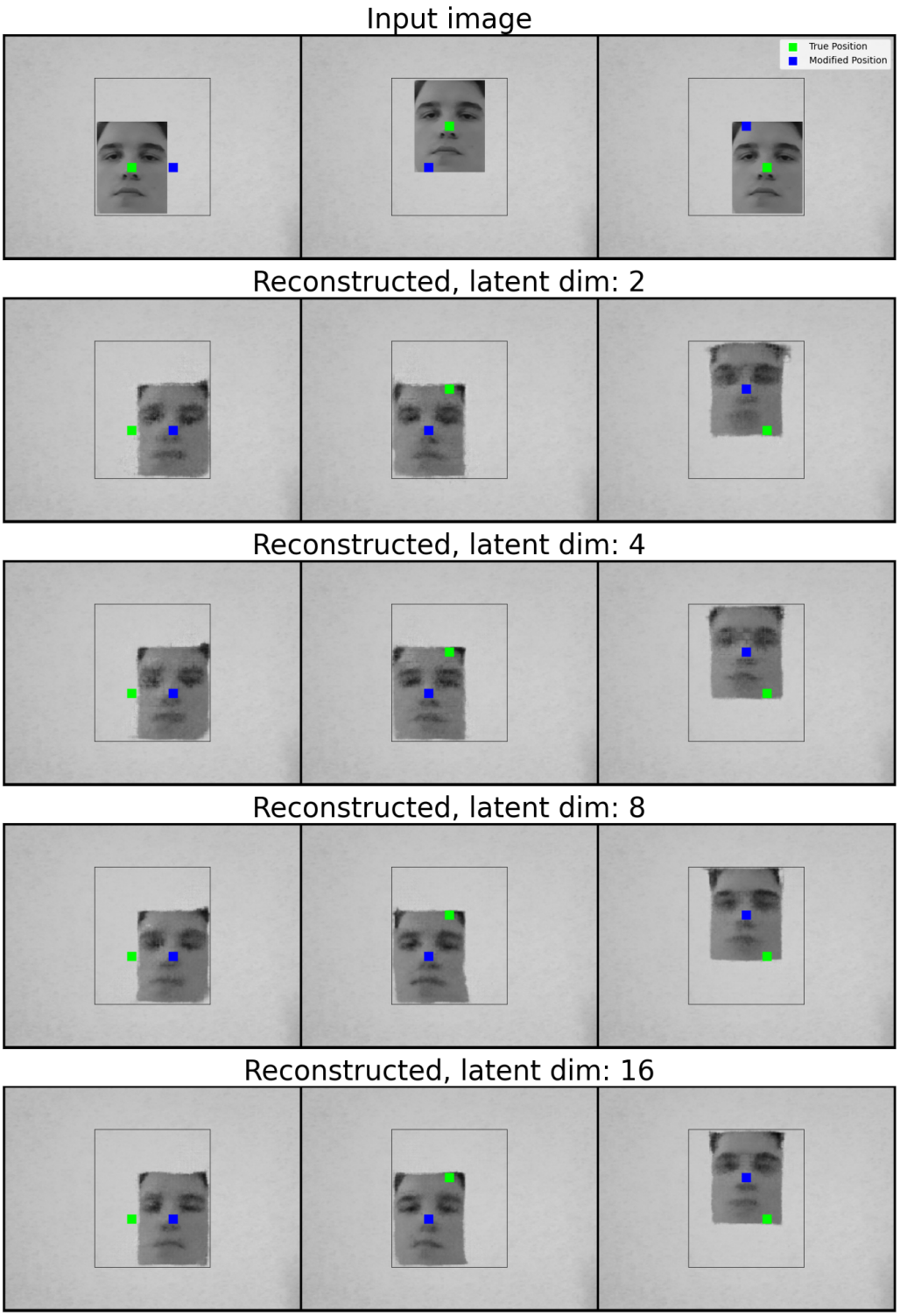


Figure A.4: Input images from the position dataset and reconstructed images from a modified latent vector using the swap-direct model with 2, 4, 8 and 16 latent dimensions. The green square marks the true center of the face in the input image. The objective is to move the images from the green square to the blue square by updating the first 2 dimensions of the latent vector representing position (x, y).

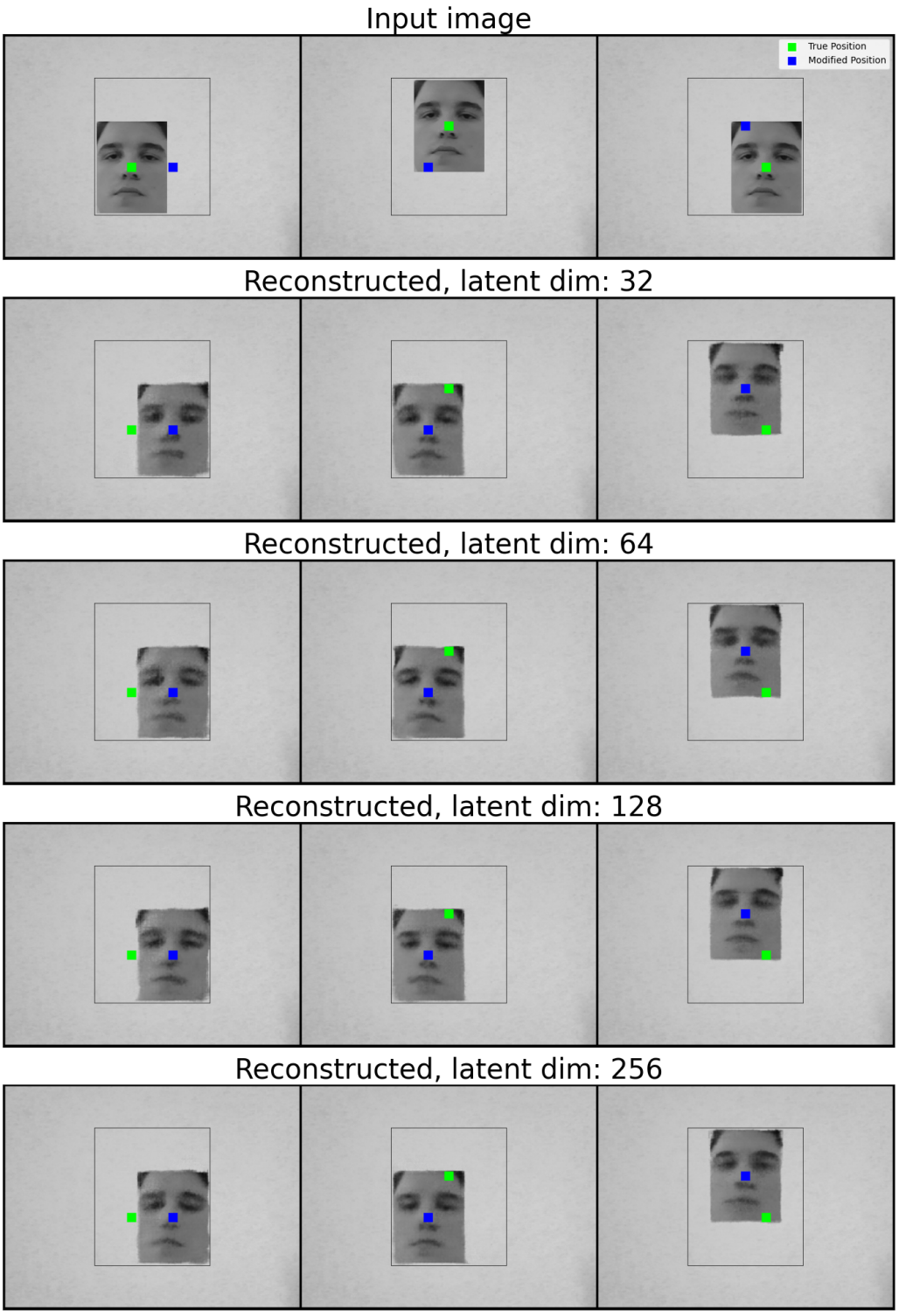


Figure A.5: Input images from the position dataset and reconstructed images from a modified latent vector using the swap-direct model with 32, 64, 128, and 256 latent dimensions. The green square marks the true center of the face in the input image. The objective is to move the images from the green square to the blue square by updating the first 2 dimensions of the latent vector representing position (x, y).

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY