Variational AutoEncoders for Image Super-Sampling

Chad Weatherly
COSC 6372 Computer Graphics – Final Project
Department of Computer Science
University of Houston
Houston, TX
cdweathe@central.uh.edu

***Abstract*— The use of Variational Autoencoders for image super-resolution has been a novel method that has gained more attention in recent years. One notable example is NVIDIA's DLSS software, which uses a VAE network to improve the performance of high-quality graphics rendering in real-time. VD-VAE, a very deep VAE from OpenAI, is one of the most prevalent models for image super-resolution in Academia. Researchers have attempted to improve upon VD-VAE's structure by conditioning the generator not only on the inference parameters but also on a low-resolution version of the image. In this paper, we propose a convolutional variational autoencoder model for single image super-resolution, a task of increasing the resolution of a low-resolution image to a high-resolution image. We used the Fashion MNIST dataset and updated the loss function to create a latent space that was also similar to the low-resolution image. We were able to get our model to find overall trends in images but lacked the ability to sufficiently pick up on more minute details.**

Keywords*— *Generative modeling, Variational autoencoders, Convolutional Layers, Image Super-Sampling, Deep Learning Super-Sampling

I. Introduction

Super sampling of images is a challenging problem in computer vision and image processing. The goal is to generate a high-resolution version of a low-resolution input image. This problem arises in various applications, such as medical imaging, video conferencing, surveillance, and more recently in video game graphics rendering. A high-resolution image contains more details and is visually more appealing than a low-resolution image. However, increasing the resolution of an image is not a trivial task, and various approaches have been proposed to tackle it.

One of the most common approaches to super sampling is interpolation, which involves estimating a value between two known values. In the context of image super sampling, interpolation is used by scaling the known pixels and then leveraging known pixels to fill in information about unknown values to nearby pixels. However, these methods have limitations. Interpolation usually results in blocky or blurry images, which are not much of an improvement over the low-res image.

Recently, deep learning-based methods have shown promising results in image super sampling. These methods learn a mapping function between the low-resolution and high-resolution images using deep neural networks. The most used deep learning-based methods for image super sampling are convolutional neural networks (CNNs) and generative adversarial networks (GANs). In the past few years, super sampling based on variational autoencoders (VAEs) have been introduced [1], with NVIDIA being one of the biggest implementers of this new model architecture. Video game graphical computations have exponentially increased as gamers want higher refresh rates, higher resolutions, higher framerates, and included ray-tracing in their game rendering. NVIDIA's state-of-the-art Deep Learning Super-Sampling (DLSS) software uses Deep Learning to render low-resolution graphics to higher resolutions, with up to 120 frames per second (FPS) and ray tracing implemented [2], [3].

II. Preliminaries

A. Variational AutoEncoders

VAEs are an updated version of the traditional autoencoder structure. An input vector $\boldsymbol{x}$ is fed to a neural network with two components, an encoder, and a decoder. The encoder takes $\boldsymbol{x}$ and transforms it to a lower dimension (latent space). This transformation is represented by the vector $\boldsymbol{z}$. This $\boldsymbol{z}$ vector is then fed to the decoder, which maps the vector to $\hat{\boldsymbol{x}}$, which should be a good approximation of $\boldsymbol{x}$. The loss function attempts to minimize the distances between the two vectors (1).

$$Loss = \frac{1}{\mathrm{N}} \| \boldsymbol{x} - \hat{\boldsymbol{x}} \|^2 \tag{1}$$

Using an optimization technique like gradient descent, this loss function creates a model which finds the optimal way to map $\boldsymbol{x} \rightarrow \boldsymbol{z}$, such that $\boldsymbol{z}$ is an efficient representation in latent space of $\boldsymbol{x}$. It works similarly to Principle Component Analysis (PCA), in that it finds an optimal mapping to a lower-dimension so that as much variational information as possible is maintained within the data set. The decoder then takes this latent representation and can effectively reconstruct an inverse mapping from $\boldsymbol{z} \rightarrow \hat{\boldsymbol{x}}$. However, the deterministic nature of the autoencoder effectively constrains the latent distribution to a small distribution and strict mapping.

The VAE architecture attempts to improve upon this limitation by introducing a key idea: mapping $\boldsymbol{x} \rightarrow p(\boldsymbol{z})$, so the output of the encoder is a sample from $p(\boldsymbol{z})$. This makes the model stochastic, where the encoder is defined by the distribution $p(\boldsymbol{z}|\boldsymbol{x})$, and the decoder is defined by the distribution $p(\mathbf{x}|\boldsymbol{z})$. We know how to calculate these probabilities using Bayes Theorem (2). The traditional VAE assumes that $p(\mathbf{x}|\boldsymbol{z})$ (decoder) and $p(\mathbf{z})$ are both Gaussian, with $p(\mathbf{z})$ assumed to follow the Normal distribution (3) and $p(\mathbf{x}|\mathbf{z})$ having parameters defined by a function f and $\mathbf{z}$ (4), where c is a covariance vector and $\boldsymbol{I}$ is the identity matrix.

$$\mathrm{p}(\boldsymbol{z}|\boldsymbol{x}) = \frac{\mathrm{p}(\boldsymbol{x}|\boldsymbol{z})\,\mathrm{p}(\boldsymbol{z})}{\mathrm{p}(\boldsymbol{x})} \tag{2}$$

$$p(\boldsymbol{z}) \sim N(0,1) \tag{3}$$

$$\boldsymbol{z} \sim N(f(\boldsymbol{z}), c\boldsymbol{I}) \tag{4}$$

The encoder and decoder use neural network architectures to parameterize and approximate their respective distributions. In order to optimize the parameters, the encoder's initial neural network layers map to a latent vector $\boldsymbol{z}'$, which is then passed to two linear layers separately. One layer calculates the corresponding variances σ_x^2 (given by the function $g(\boldsymbol{z}')$), and the other the mean values μ_x (given by the function $h(\boldsymbol{z}')$). This creates a $|\boldsymbol{z}|$ number of distributions from which to draw for each value in $\boldsymbol{z}$. Therefore, $\boldsymbol{z}$ can be represented as a function of the mean and variance values (5), then reparametrized by multiplying a random sample from $N(0,1)$ by the correspond mean and standard deviation (6), so that optimization with gradient descent can be performed, where ζ is a sample drawn from $N(0,1)$.

$$\boldsymbol{z} = N\big(h(\boldsymbol{z}'), g(\boldsymbol{z}')\big) \tag{5}$$

$$\boldsymbol{z}_x = \sigma_x \zeta + \mu_x \tag{6}$$

This adjustment to the autoencoder structure allows for the mapping from $\boldsymbol{x} \rightarrow \boldsymbol{z}$ to be more agile when it comes to expressing the variation in $\boldsymbol{x}$. It allows slight overlapping of classes and helps with the problem of image super-sampling, since it is a one-to-many mapping problem. The loss function (7) for the VAE varies from the autoencoder in that it adds a second term, the Kullback-Leibler Divergence between the sampled $\boldsymbol{z}$ vector and the Normal distribution $N(0,1)$, where C is a scaling parameter. The Kullback-Leibler Divergence for two distributions, p, q, and a data sample X is given (8).

$$Loss_{VAE} = \frac{\mathrm{C}}{\mathrm{N}} \parallel \boldsymbol{x} - \hat{\boldsymbol{x}} \parallel^2 + D_{KL}[\boldsymbol{z}, N(0,1)] \tag{7}$$

$$D_{KL}(p,q) = \sum_{\mathrm{i}=1}^{\mathrm{N}} p(x_i)\, log\left(\frac{p(x_i)}{q(x_i)}\right) \tag{8}$$

The Kullback-Leibler Divergence is a measure of entropy between two distributions. For the VAE, it constrains $\boldsymbol{z}$ (5) to be similar to the Normal Distribution. In this way, the VAE improves on the traditional autoencoder by forcing the latent space to be probabilistic, allowing more variation to be captured in the latent space. This only adjusts the structure of the encoding network but leaves the decoder unchanged, compared to the traditional autoencoder network.

B. Image Frequency Information

As previously mentioned, the image super-sampling problem is many-to-one. The super-sampling (SS) algorithm must find a probable, scaled-up high-resolution (HR) image from a low-resolution (LR) input, given the information contained in the low-resolution image. A VAE can be thought of as having two components, an encoder and a decoder, but these two parts can also be categorized as an inference and generator function, respectively. We then want to be able to apply the VAE algorithm to image data. During training, a high-resolution image is passed through the entire network, and a new image is generated, ideally similar to the original image.

The goal of the network is for the latent space to compress the information contained in the original image while retaining the information integrity as best as possible. This information tends to be frequency information. Without going into too much detail, this is best explained by understanding the Fourier Transform (FT) of an image. The FT contains information about the underlying frequencies that make up the pixel values in the image [4]. Higher frequencies are what make up much of the detail in the image. Lower frequencies make up many of the larger details, like generic shapes, boundaries, and color patterns. We want the VAE model for an image to be sufficiently complex to be able to effectively encode both high and low frequency information in the latent space.

III. Prior Work

As mentioned previously, the work of using VAEs for image SS is a novel method that has been explored more extensively in the past few years. Most notably, NVIDIA uses a VAE network in its DLSS software that is available on its upper-level GPUs to vastly improve performance of high-quality graphics rendering, including running games at 4K, with a high frame rate and raytracing. NVIDIA's model is surely one of the largest and most complex, as it is able to perform real-time rendering and is conditioned on the motion vectors of the objects in-game [2], [3], which explains its incredible ability to produce high quality images in real time.

One of the most prevalent models, by Child from OpenAI, is VD-VAE, a Very Deep VAE that has a latent space composed of several tensors, each conditional on the previous [5]. It also takes advantage of the Ladder VAE concept, which conditions the generator parameters on the inference layers [6]. The idea of VD-VAE is that given sufficiently deep stochastic layers to represent the latent space, robust representations are able to be created which are simple for the generator to recreate. However, it is unclear how a new, low-resolution test image would be used by the VD-VAE to create a high-resolution image.

VD-VAE's ability to recreate high-resolution images subjected it to improvements by a few papers, included those by Chira et al. and Prost et al., which attempted to add on to the VD-VAE structure by conditioning the generator not only on the inference parameters, but also by a similar inference model that was trained on a low-resolution version of the image [7], [8]. Again, it is unclear exactly how new test cases with only a low-resolution input would be treated.

In addition, Liu et al. attempted to create a reference-based VAE model for image SS that reported ambiguous results when compared to other simple CNNs [9].

IV. Methods

A. Dataset

The data set is the Fashion MNIST data set, which consists of 70,000 gray-scale images of size 28x28, where each image has one of 9 labels of various fashion items. The

data set was downloaded from the PyTorch torchvision package. We used these images as our high-resolution images. We then created corresponding low-resolution images for each high-resolution image, with each image being of size 14x14. This was done by averaging the pixel values of each 4x4 block in the high-resolution image. We will refer to these images as $\boldsymbol{x}_{LR}$ and $\boldsymbol{x}_{\mathrm{HR}}$ for the rest of the paper. An example of these images is shown side-by-side is in Figure 1.

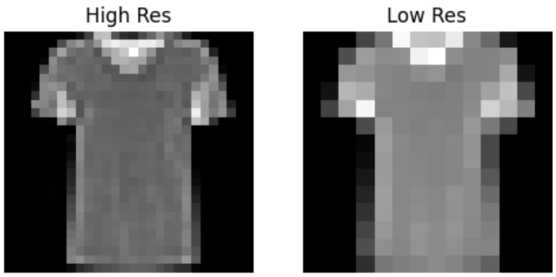

Figure 1: An example of a high-resolution image ($\boldsymbol{x}_{HR}$) and its corresponding low-resolution ($\boldsymbol{x}_{LR}$) counterpart.

B. Model Architecture

Using PyTorch in Python, we created a VAE model to handle images. In order to do this, the encoder and decoder are mostly convolutional layers, with linear layers at each end (Fig. 2). It has 3 component architectures that make up the entire forward process of the model: encode, reparametrize, and decode.

1) Encode

The encoder is a simple neural network which takes in an image of size 28x28 with 1 channel (grayscale) and performs two convolutions on the image to have a final image with 32 channels that is 7x7. Preferably, this method encodes the necessary frequency information into this new image, which will then be passed to the reparameterization.

2) Reparameterize

The output of the encoder is then vectorized and passed to the two different linear layers, which output the σ_x^2 values and μ_x values from which to draw the $\boldsymbol{z}$ latent distribution, one distribution for each pixel, meaning that for a 14x14 latent image, there are 196 distributions. The σ_x^2 and μ_x values are passed through a Hyperbolic Tangent and Sigmoid Function, respectively, to keep their values small and closer to the Normal Distribution. Accordingly, we wanted our latent space to be represented by $\boldsymbol{x}_{LR}$, which means that the encoder is essentially a stochastic mapping from $\boldsymbol{x}_{\mathrm{HR}} \rightarrow \boldsymbol{x}_{\mathrm{LR}}$, where $\boldsymbol{x}_{\mathrm{LR}} \approx \boldsymbol{z}$. We wanted to do this so that the decoder (generator) would essentially be a mapping from $\boldsymbol{x}_{\mathrm{LR}} \rightarrow \boldsymbol{x}_{\mathrm{HR}}$, thus being an efficient generator for a SS operation of a low-resolution image, $\boldsymbol{x}_{\mathrm{LR}}$ to a high-resolution image, $\boldsymbol{x}_{\mathrm{HR}}$.

3) Decode

The decoding stage of the model is entirely the same as in the traditional autoencoder and VAE models. The only difference in this case is the inclusion of a Convolutional Transpose layer that takes in the latent image. This operation is the inverse operation as the convolutional layers used in the encoder. After the image has been encoded with higher channels, this data is vectorized and passed into a linear layer, which outputs 576 values into a ReLU activation function and then a sigmoid function. These 576 values are then in the interval [0,1] and are reshaped to create our reconstructed 28x28 image, $\boldsymbol{x}_{\mathrm{HR}}$.

C. Experiments

During training and testing, we took a slightly different approach to the loss function than previous works. In the loss function for our VAE, which we will call CNN-VAE, we removed the KL term entirely, since the parameter values σ_x^2 and μ_x were kept small via their respective activation functions. We then added a term to minimize the difference between the latent image and our created low-resolution image, $\boldsymbol{x}_{LR}$ (11), where α is a regularization parameter. We also chose to use Binary Cross Entropy loss as our loss function (BCE) between any two images, $\boldsymbol{x}_{\mathrm{i}}$ and $\boldsymbol{x}_{\mathrm{j}}$ (12).

$$Loss_{\mathrm{RECON}} = \alpha * BCE(\boldsymbol{x}_{HR}, \boldsymbol{x}_{RECON}) \tag{9}$$

$$Loss_{\mathrm{LR\text{-}Z}} = (1-\alpha) * BCE(\boldsymbol{x}_{LR}, \boldsymbol{z}) \tag{10}$$

$$Loss_{VAE\text{-}\mathrm{CNN}} = Loss_{RECON} + Loss_{LR-Z} \tag{11}$$

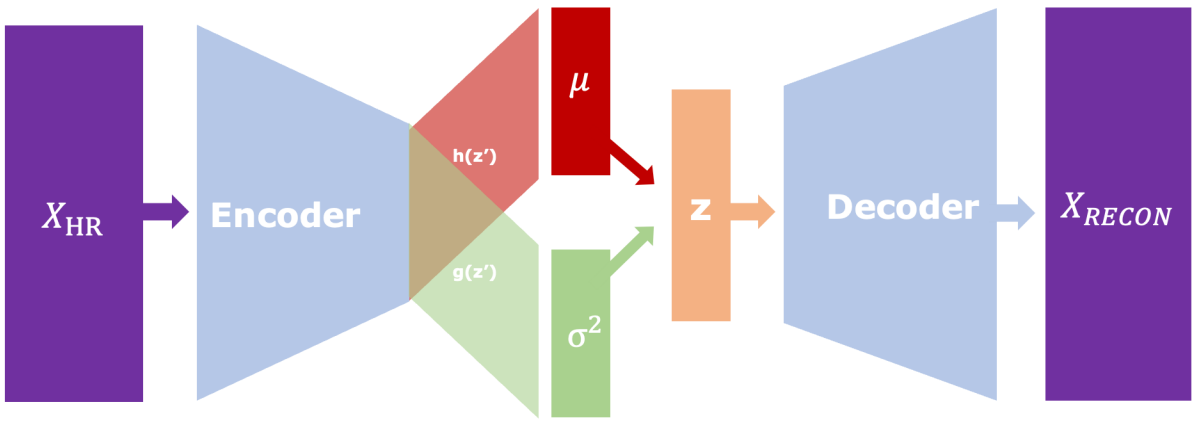

Figure 2: A simple visual representation of the CNN-VAE

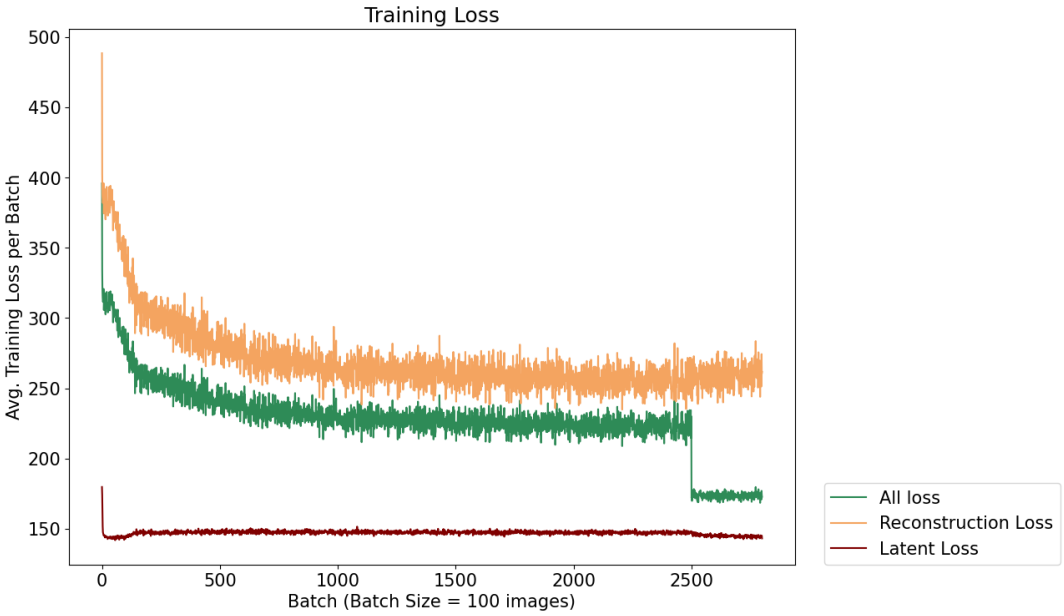

Figure 3: Training Loss across batches for the CNN-VAE. Notice the sharp drop in loss at ~ 2500 batches. This was due to changing the α parameter, resulting in a lower average loss.

$$BCE(\boldsymbol{x}_i, \boldsymbol{x}_j) = -\sum_{\mathrm{p}=1}^{\mathrm{N}} \boldsymbol{x}_i + log(\boldsymbol{x}_j) \quad (12)$$

This way, our model functions are trained to map the original image to the low-resolution image and then reconstructed back to the original high-resolution image. The idea is that, given only a low-resolution image, it can be passed through the trained decoder and automatically upsampled to a higher-resolution image.

Training was done using the Adam Optimizer with a learning rate of 5e-6, with batches consisting of 100 images. The optimizer finds an optimal set of weights between the linear layers and for the kernels used in the convolutional layers. 60,000 of the original images were used for training and the remaining 10,000 were used to calculate testing loss after every 50 batches were used for training. For training on the first 2500 batches, an $\alpha = 0.75$ was used. However, for the final 500 batches, in order to try and get the latent image to be more closely aligned with the low-resolution image, an $\alpha = 0.25$ was used (This is observed by a sharp drop in average training and testing accuracy on figures 3 and 4, respectively). Training and testing were done a MacBook Pro M2 chip, with optimization on PyTorch achieved via the MPS backend on Apple Silicon.

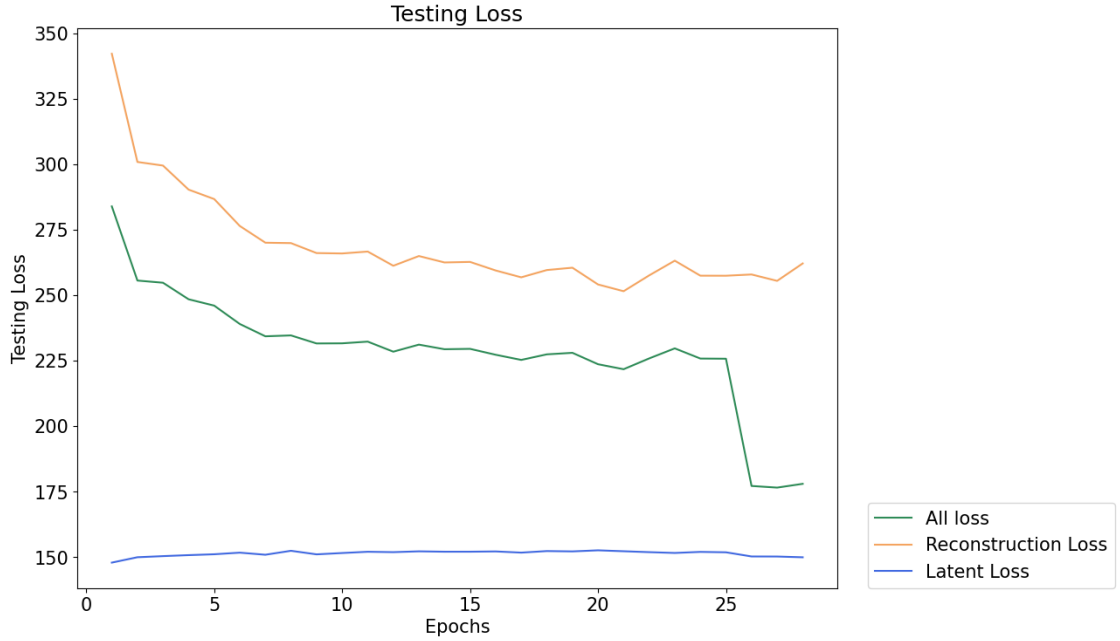

Figure 4: Test Loss across Epochs (every 500 batches) for the CNN-VAE. The same drop mentioned in Fig. 3 is also present here

V. Results

The training and testing phases had to be updated to find the best parameters and loss function for tuning and creating a model that effectively generates reconstructed images. Despite this, parameters were found that effectively reduced the training and testing loss to optimal minimums (Fig. 3, 4). We then wanted to take a few random samples of data and try to see how well they could be reconstructed, both by passing $\boldsymbol{x}_{\mathrm{HR}}$ through the entire VAE network (Fig. 5) and then attempting to pass a low-resolution image $\boldsymbol{x}_{\mathrm{LR}}$ through only the decoder network, treating $\boldsymbol{x}_{\mathrm{LR}} \approx \boldsymbol{z}$, which would see how well the decoder could successfully SS a new test image without the aid of a high-resolution image (Fig. 6).

VI. Discussion

Given the training and test loss graphs (Fig. 3, 4), we can see that that the optimizer was able to train the model and find an optimal minimum. However, looking at the reconstructed images tells us the true story.

When the entire model is used to reconstruct an image from a high-resolution input (Fig. 5), the model performs fairly well at recreating the image's overall shape, along with some minor details, like general color differences across the shoe. We also see that the images are still blurry and tend to be shallow reconstructions of the original image.

When only the low-resolution image is fed through the generator function, the images are less robust, but still noteworthy (Fig. 6). Overall, even without the high-resolution image for reference, the generator can maintain the overall structure of the image for most objects. Admittedly, a few of the objects are transformed into a totally different object, the pants, even when the original high-resolution image shows something completely different. It seems that the latent space is recreated in such a way that each type of clothing has its own distribution, which would explain why many reconstructed images look like pants.

These reconstructions tell us that the latent space is not sufficiently complex to fully separate out all the variation in the frequency space, as often the larger frequencies are maintained, like overall shape and outlines, while finer-grained details are entirely lost, resulting in grainy or blurry reconstructed images. For this project, it would be difficult to have much more complexity in the model, as it is run on a laptop computer with limited computational resources.

On the other hand, for future work, it would be interesting to see what items can condition the CNN-VAE so that it creates a better latent distribution space. NVIDIA uses motion vectors to condition the generator, and in a single image SS, it could be useful to condition the model on class, as different classes most likely have different latent distributions. Overall, I am satisfied with the results of this project given the constraints but look forward to future endeavors regarding this project and its improvements.

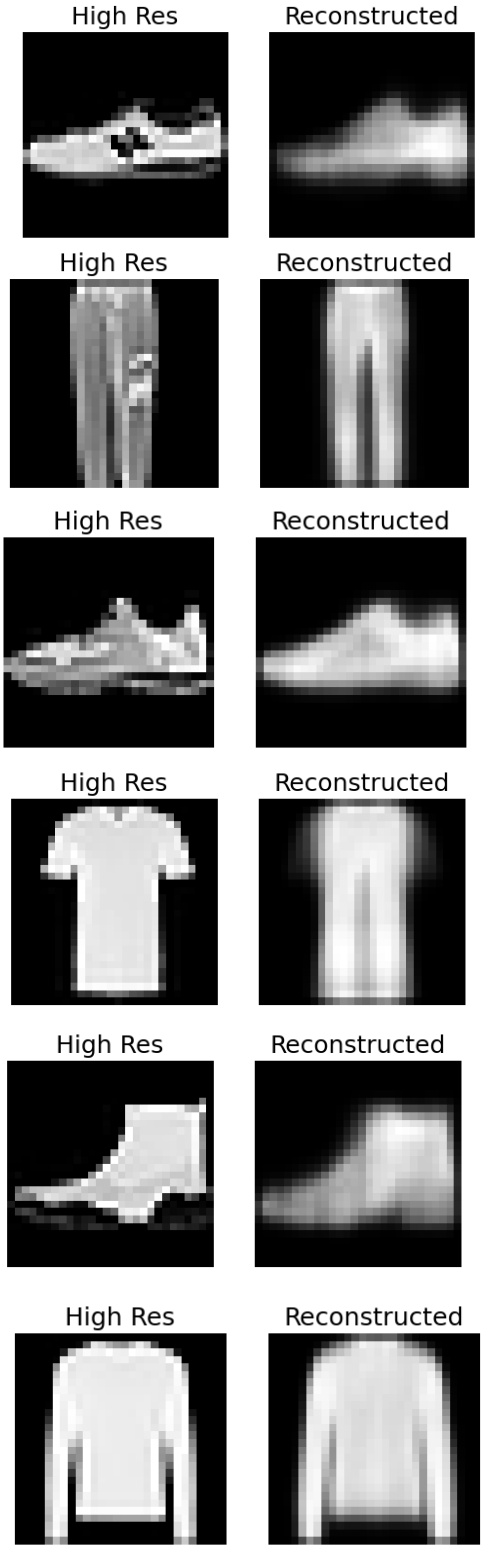

Figure 5: Reconstructed Images created by CNN-VAE when $\boldsymbol{x}_{HR}$ is passed through the entire network

REFERENCES

[1] S. Hyun and J.-P. Heo, "VarSR: Variational Super-Resolution Network for Very Low Resolution Images," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., in Lecture Notes in Computer Science, vol. 12368. Cham: Springer International Publishing, 2020, pp. 431–447. doi: 10.1007/978-3-030-58592-1_26.

[2] "Deep Learning Super Sampling (DLSS)," *NVIDIA Developer*, Sep. 18, 2018. https://developer.nvidia.com/rtx/dlss (accessed Apr. 26, 2023).

[3] A. Watson, "Deep Learning Techniques for Super-Resolution in Video Games".

[4] "The fast Fourier transform and its applications," *Guide books*. https://dl.acm.org/doi/abs/10.5555/47314 (accessed Apr. 26, 2023).

[5] R. Child, "Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images." arXiv, Mar. 16, 2021. Accessed: Apr. 21, 2023. [Online]. Available: http://arxiv.org/abs/2011.10650

[6] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder Variational Autoencoders".

[7] D. Chira, I. Haralampiev, O. Winther, A. Dittadi, and V. Liévin, "Image Super-Resolution with Deep Variational Autoencoders," in *Computer Vision – ECCV 2022 Workshops*, L. Karlinsky, T. Michaeli, and K. Nishino, Eds., in Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 395–411. doi: 10.1007/978-3-031-25063-7_24.

[8] J. Prost, A. Houdard, A. Almansa, and N. Papadakis, "Diverse super-resolution with pretrained hierarchical variational autoencoders".

[9] Z.-S. Liu, W.-C. Siu, and L.-W. Wang, "Variational AutoEncoder for Reference based Image Super-Resolution," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 516–525. doi: 10.1109/CVPRW53098.2021.00063.

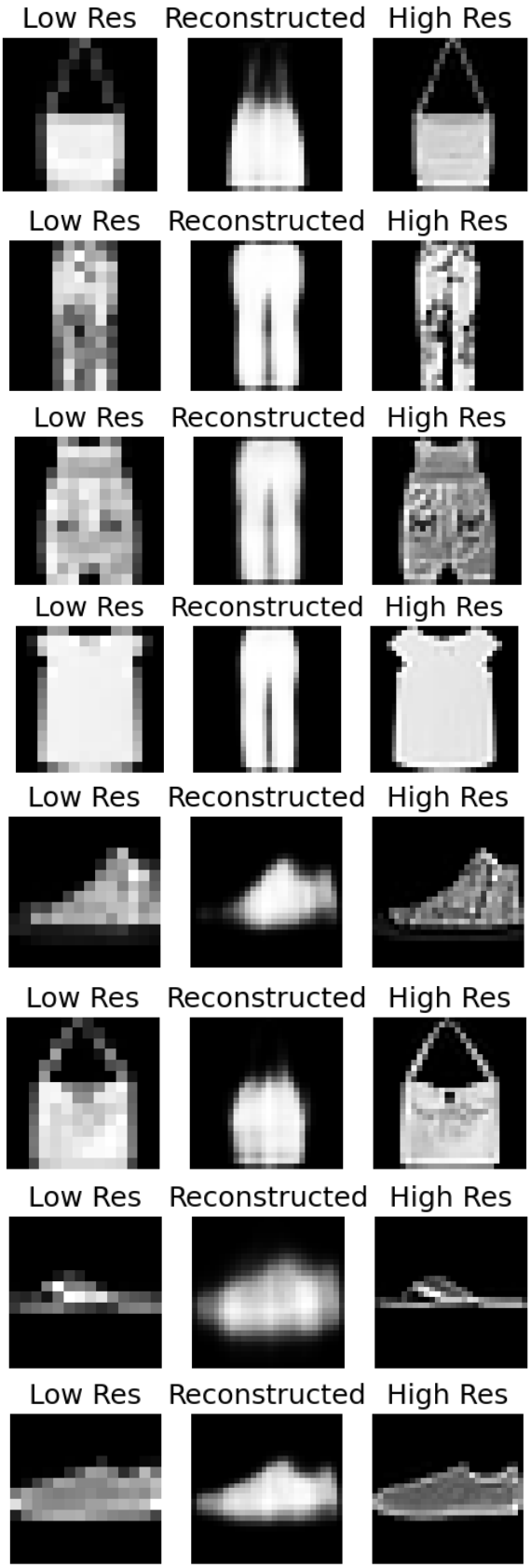

Figure 6: Reconstructed images created by only the decoder component of CNN-VAE when $\boldsymbol{x}_{HR}$ is passed through it as the latent image $\boldsymbol{z}$, along with what the true high-resolution representation looks like.