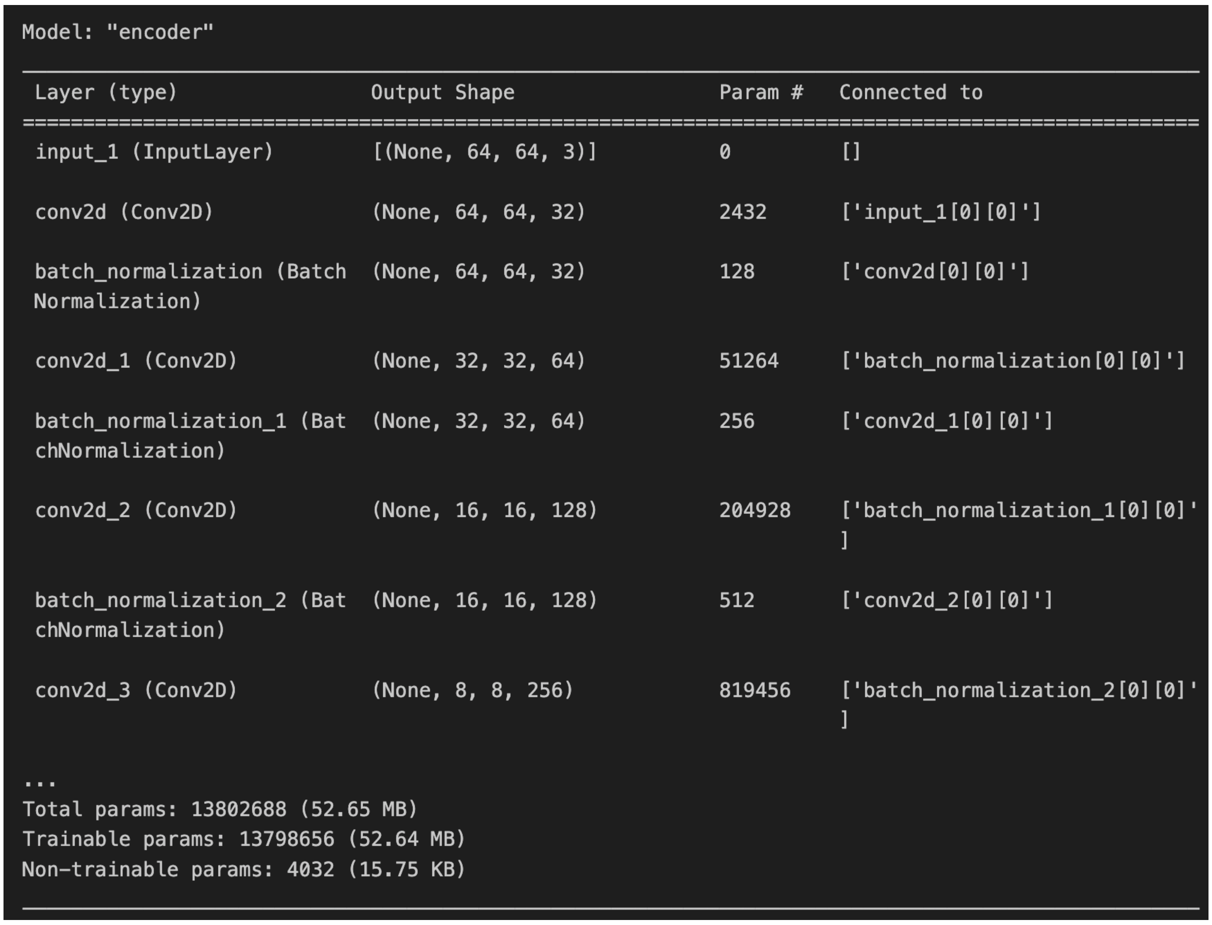
Image Generation using Variational autoencoder (VAE)

Problem Statement:

Develop synthetic data samples to complement or enrich an existing data set to train machine learning models. Explore the applications of generative models to generate extra training examples, especially in cases where collecting real data is pricey or tricky. In our situation, we make use of a database of images, to teach a model how to return other images, having the same style.

Encoder - Model Summary



```
Model: "encoder"
_____________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
=============================================================================================
 input_1 (InputLayer)        [(None, 64, 64, 3)]          0         []

 conv2d (Conv2D)             (None, 64, 64, 32)           2432      ['input_1[0][0]']

 batch_normalization (Batch  (None, 64, 64, 32)           128       ['conv2d[0][0]']
 Normalization)

 conv2d_1 (Conv2D)           (None, 32, 32, 64)           51264     ['batch_normalization[0][0]']

 batch_normalization_1 (Bat  (None, 32, 32, 64)           256       ['conv2d_1[0][0]']
 chNormalization)

 conv2d_2 (Conv2D)           (None, 16, 16, 128)          204928    ['batch_normalization_1[0][0]'
                                                                     ]

 batch_normalization_2 (Bat  (None, 16, 16, 128)          512       ['conv2d_2[0][0]']
 chNormalization)

 conv2d_3 (Conv2D)           (None, 8, 8, 256)            819456    ['batch_normalization_2[0][0]'
                                                                     ]

...
Total params: 13802688 (52.65 MB)
Trainable params: 13798656 (52.64 MB)
Non-trainable params: 4032 (15.75 KB)
_____________________________________________________________________________________________
```

Encoder - Hyperparameter Tuning

1. Kernel Size: Convolutional layers having 5x5 kernels are used. A kernel size decides the range of a sliding window while performing convolution. Smaller size kernels are used for finer details, meanwhile larger size one is for more global features. Tweaking such parameters should be done by a trade-between capturing local features and global patterns.

2. Activation Function: Leaky ReLU with a slope of - 0.02 is utilised. Unlike the vanishing gradient problem, LeakyReLU has a small slope when it is inactive. This is useful because when it is active, it has a large slope. Slope parameter tuning ensures the efficiency of gradient propagation by allowing the model to capture and bridge the gradients during training.

3. Strides: The convolutional layers have a stride of 2 implying the moving of the filter over two pixels hence in both, the height and width directions. It influences the layers sizes and results in increase of received data. Increased steps result in decreased frequency flow of the feature maps in spatial area, which may eventually lead to avoiding lower level features, but might also result in relatively coarse outcomes with less details.

4. Padding: 'Same' padding is used, which pads the input by the same number of pixels on all four sides and, as a result, the output has the same width and height as the input. This therefore preserves the spatial aspect (location). Changing padding can be how the convolutional layers would output their edges.

5. Batch Normalisation: The operation of batch normalisation is employed after each convolutional and deep layer separately. The simplicity of this technique normalises the activations of the previous layer, so it leads to more stable training and faster convergence. Scheduling batch size may contribute a lot to the training speed and training stability.

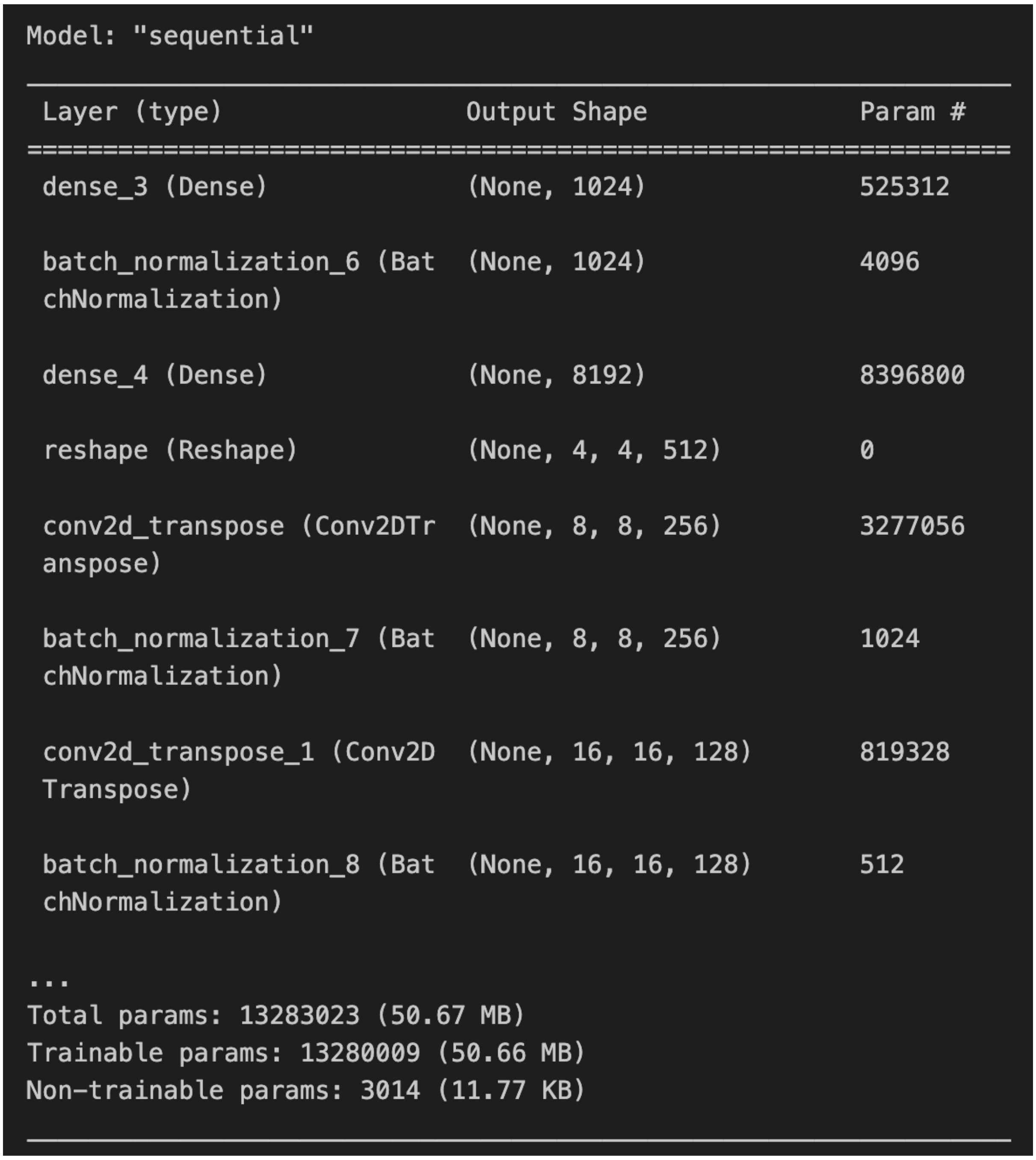
6. Filter Size: Respectively, after the convolutional layers, 3 types of filters are employed, they are [64, 128, 256, 512]. This is done through assigning the suitable number of filters (or channels) to each convolutional layer and by doing so the capacity of the model to learn the features at different abstraction levels is determined. By expanding the filter size, model capacity increase will be precise but we may also face the potential risk of overfitting.

7. Latent Dimension (latent_dim): A key hyperparameter of the dimension of latent space in VAEs is the dimensionality. It indicates the number of factors which can capture the main part of the variation in the data. Overall, dimensionality reduction is a useful tool for reducing the complexity of data and extracting meaningful insights from it. Determining the correct number of latent factors is a case of choosing between the desire to capture entities in the representation and the need of keeping the computation easy.

8. Activation Function for Dense Layer: SELU (Scaled Exponential Linear Unit) is used in dense layer as an activation function. The SELU algorithm has been demonstrated to possess self-normalisation abilities which safeguard the training process and speed it up at the same time. Tweaking activation functions involves trying a variety of functions to see which one functions best depending on the nature of the problem.

To ensure tuning of these hyperparameters, one can work with approaches like grid search, random search, or Bayesian optimization while keeping an eye on the model’s performance on a validation set. The adjustment of hyperparameters has significant influence in the performance of the model, this is because it affects the capability of the model to learn meaningful representations, the rate of convergence and generalisation ability.

Decoder - Model Summary



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 1024)              525312

 batch_normalization_6 (Bat  (None, 1024)              4096
 chNormalization)

 dense_4 (Dense)             (None, 8192)              8396800

 reshape (Reshape)           (None, 4, 4, 512)         0

 conv2d_transpose (Conv2DTr  (None, 8, 8, 256)         3277056
 anspose)

 batch_normalization_7 (Bat  (None, 8, 8, 256)         1024
 chNormalization)

 conv2d_transpose_1 (Conv2D  (None, 16, 16, 128)       819328
 Transpose)

 batch_normalization_8 (Bat  (None, 16, 16, 128)       512
 chNormalization)

...
Total params: 13283023 (50.67 MB)
Trainable params: 13280009 (50.66 MB)
Non-trainable params: 3014 (11.77 KB)
_________________________________________________________________
```

Decoder - Hyperparameter Tuning

1. Activation Function: Employed in the dens layers is the SELU (Scaled and Exponential Linear Unit) activation. SELU provides its own encoding which normalises itself and hence curbs the number of stages or parameters that may be required. Perfecting the activation functions is centred around trying many different functions to gather the most appropriate one for the work being accomplished.

2. Batch Normalisation: As batch normalisation, we do it between each dense and transpose convolutional layers. It constitutes as stability and speed-up mechanisms for the training by making the outputs uniform. Intervening for batch size can affect the speed and reliability of training.

3. Dense Layer Output Shape: The first dense layer produces an output of the form (1024,), where this form is based on the intrinsic dimension. The configuration of the current layer is in a way that it directs the others and eventually determines the architecture of the decoder.

4. Reshape Layer: Unprocesses the output of the first dense layer and forms it to (4,4,512) which sets it for the transposed convolutional layers.

5. Convolutional Transpose Layers: The lot of convolutional transpose layers is used for upsampling feature maps. With each successive layer, these are the factors which advance the spatial dimensions and are ever closer to the intended result. These settings involve kernel size, leaky ReLU activation function, strides, and the padding technique.

6. Output Layer Activation: The use of sigmoid function in the output layer is done by the output layer. Often it is a sigmoid that is applied to an output layer for binary classification purposes for the reason that it gives values from zero to one. There, the probable reason is to make sure that the value of the pixel doesn't go out of the valid range [0, 1].

To tune these hyperparameters, similar techniques to those used for the encoder can be applied, such as grid search, random search, or Bayesian optimization. Experimenting with different configurations while monitoring the model's performance on a validation set is crucial. Tuning hyperparameters can affect the model's ability to reconstruct input data accurately, its convergence speed, and its generalisation capability.

Some other models that work well for Image Generation

1. Generative Adversarial Networks (GANs): GANs are made of two neural networks, a generator and a discriminator, whose parameters are learned simultaneously in a way that pits the generator against the discriminator. The generator with the goal to produce authentic images from a randomly generated noise and the discriminator that classifies such images as real or generated is the setup. GANs are commonly used for generating photo-realistic and rich samples which has resulted in their usage for tasks like image synthesis and transferring style of one image to another.

2. Autoencoder-based Generative Models: Autoencoders still continue to be an increasingly popular type of generative model. In addition to modifications such as adding Adversarial Autoencoders (AAEs) and Adversarially Regularized Autoencoders (ARAEs) to generate their own trained data, autoencoders also combine the adversarial training framework of GANs. These models are able to learn generating images while facilitating the reconstruction process and turning the focus on both quality and variety.

3. PixelCNN and PixelRNN: PixelCNN and PixelRNN are autoregressive generative models which produce images step-by-step from a predefined probability distribution. It do this by formulating a multivariate conditional probability distribution of each pixel with the previous ones and thus have precise regulation over image production. Though they are data intense, these type of models can give high definition pictures having small details.

4. Variational Autoencoder with Adversarial Training (VAE-GAN): VAE-GAN integrated the statistical modelling of VAEs along with the adversarial learning of GANs which enhances the visuals as well as diversity. Through the use of simultaneously optimising reconstruction and adversarial objectives, VAE-GANs can produce images of sharper details and more closely resemble the real thing as opposed to traditional VAEs.

Using GAN VS VAE

1. Training Dynamics:
- GANs: GAN, on the other hand, adopts a twin architecture network of the generator and discriminator, which are being trained simultaneously but in a way to compete with one another. Generator identifies what kind of images the discriminator should not allow to get into the system, therefore, generator produces fake images to deceive the discriminator, while discriminator learns to distinguish true from generated images. The overall training process can turn out to be unstable and may fall into the mode collapse phase during the training.

- VAEs: VAEs obtain training by maximising the evidence lower bond (ELBO), which is composed of a reconstruction term and a regularisation term. The reconstruction term that is usually the MSE loss or Cross-Entropy persuades the model to create images that are similar to input data, while the Regularization term pushes the learned latent space to follow a certain prior distribution, usually the Gaussian distribution. VAE training does not struggle as much as GAN when the training process is concerned.

2. Latent Space:
- GANs: This generator is performed in two steps, when the the generator maps general noise from image space to latent representation space, rather than the other way around.

- VAEs: VAEs try to divide the input data on to a low-dimensional space for an in-depth understanding. Encoder network translates images in input into a space of distribution allowing us to interpolate and manipulate latent vectors for creation of new and edited images with targeted attributes.

3. Image Quality:
- GANs: Generative Adversarial Nets (GAN) mostly yield images with better clarity and perfect realism than those obtained by Variational Autoencoders (VAE). Notwithstanding, the level of images to be generated depends on the choice of architecture, training stability, and the kind of data used in the training process.

- VAEs: VAE mainly provides smooth and low-quality photos, while GANs give better textures generally. The goal of VAEs, on the other hand, is such that the model favours a mixture of interpretations for the latent variable, as opposed to finer versions, thus clouding outputs.

In retrospect, GANs for an image generation could have probably helped in the case of more skillful and diverse images than the ones obtained using the VAEs. One of the reasons why GANs are very efficient in capturing the fine details of the image and generating real samples is that they contain opposing networks. Moreover, the adversarial architecture of GANs pushes the model to learn how to fit the data good upper bound resulting in the improved image quality. Gans, though difficult and unstable to train,are widely popular and are known for their incredible results, Yet, a cautious model architecture design and hyperparameter tuning is demanded. However, the choice between GANs and VAEs might mainly vary with the particular image synthesis working conditions and the implemented tradeoffs between quality, diversity, flexibility, and training stability.

Why did we choose VAE over GAN

1. Stability in Training:
VAEs deliver much more robust dynamics in training by design in comparison with GANs. The optimization focus in VAEs aims at maximising the evidence lower bound (ELBO) that comprehends a reconstruction term and a regularisation term. This framework aims to obtain a clear optimization goal and usually leads to the algorithms with the reduced likelihood of getting stuck or unstable issues as normally appear at the side of training of GANs.

2. Explicit Latent Space Representation:
VAEs learn a latent space representation given the input data by providing them. As latent space representation of VAEs is responsible, it is reasonable to use these representations to perform a series of tasks as well. The encoder network brings the images to memory as a distribution in latent space, and allows for an extension of the latent vectors by interpolation and manipulation. This explicit manner makes the tasks of image reconstruction, interpolation and attribute modification etc possible.

3. Interpretability:
The stochastic nature of VAE implicitly offers a better interpretation of the latent space than the Generative Adversarial Networks. The fact that each dimension of the latent space represents a specific feature or property contributes to the simplification of the generation process as well as enables us to obtain a better control over it. This interpretability has the advantage over the tasks which demand identification of the elements that need to be modified for the source image.

4. Regularisation:
VAEs intrinsically encode a regularisation term in the optimization problem (as an objective function) which forces the learned latent space to assume a distribution from a prior distribution model, like gaussian. Such an agreement leads to regularisation and prevents a solution from being overfitted, that is why the latent space remains well-defined and interpretable.

5. Blending Reconstruction and Generation:
VAEs naturally structure their tasks of degradation and dissemination in one common framework. Within the genetic part of the model, the reconstruction term of the loss that is considered in the ELBO discourages the model to reconstruct input images flawlessly, but at the same time, the generative property of the model permits for new images to be generated. Hence, the VAEs have both generative ability and defective capacity, which makes it suitable for multiple tasks such as image generation and anomaly detection.

6. Robustness to Mode Collapse:
Meanwhile, VAEs are much less likely than GANs to generate blanks or lack of diversity. Mode collapsing is the mark of the generator violent to arrest all modes of the data

distribution generating poor sample diversity. To this end, the VAEs are equipped with functions to generate dirt-free samples having very distinct characteristics in order to cover the important aspects of the data.

Therefore, training image generation via a Vae contributes to the stability of modelling, clarity in latent space representation, interpretability, regularisation and the mode collapse resistance. These factors are actually the weaknesses and the strengths of VAEs at the same time. VAEs are the perfect choice in a case where both the restoring from inputs and the generating have to be done, namely where the stability, interpretability, and controllability over generation are necessary.

MADE BY:
Group 39
Rudra Joshi (J037)
Ravikant Jain (J032)
Lokesh Gupta (J027)