Ian Deal

12/13/2023

EE 456

A Novel Framework for Generating Images from Text

Abstract:

AI image generation is currently one of the hottest topics in the tech industry, with advancements being made regularly. However, even though current cutting-edge image generation models can create highly detailed images, these models still fail to replicate some of the nuances of the human imagery system.

The goal of my project is to construct a basic model of image generation that is based off theoretical mechanisms of human imagery. These mechanisms will allow the model to generate scenes of learned images composed in novel configurations.

Shortcomings of Current Image Generation Models and Cognitive Psychology Motivations:

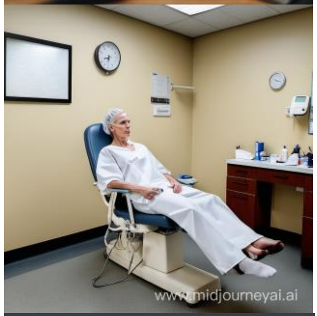
The current state-of-the-art image generation model is Stable Diffusion. Stable Diffusion generates an image from a text description by removing layers of noise from a vector of Gaussian noise until it represents a feature vector generated from the text input. This process is very good at generating an image, when images of similar features are included in the training set of the model. However, these models struggle at combining known objects in combinations that are not within the training set. For example, prompting a stable diffusion model to generate a picture of a black doctor, or a white patient in a waiting room, will result in an accurate representation of these scenes. However, prompt the model to generate an image of a black doctor and a white patient together and it will often fail to generate an accurate representation.

Fig 1-5: Samples of a Stable Diffusion Model inability to Combine Known Images in Novel Combinations: (all images were generated using MUSE AI, an online stable diffusion tool found at https://www.midjourneyai.ai/)

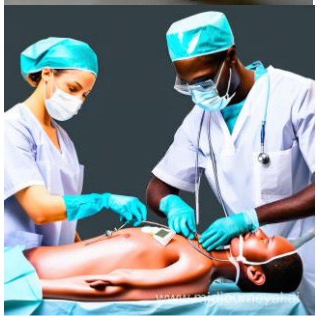
Text Description: **Generated Image:**

‘a black doctor in their office’

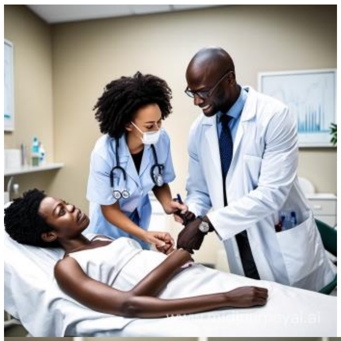
‘a white patient at their doctor’s office’



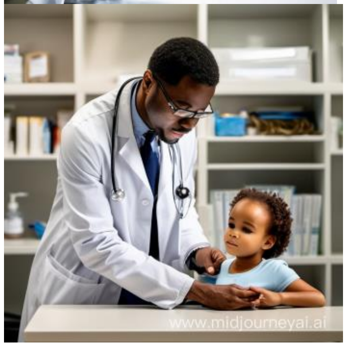
‘a black doctor operating on a white patient’



‘a black doctor treating their white patient’



‘a black doctor assisting a white child’



As shown in Fig. 1-5, the Stable Diffusion model can generate highly detailed images either a black doctor or a white patient individually but fails to generate an accurate representation of them interacting with each other. This is likely due to the model’s training set having many examples of black doctors and white patients on their own, but very few, if any, examples of the two interacting with one another. This specific example could be fixed by introducing more images of black doctors interacting with white patients to the model’s training set, however, this is only a band-aid patch over a real issue in the Stable Diffusion model.

This issue is that the Stable Diffusion model cannot generate novel combinations of known objects. This stands in contrast to the known behavior of human imagery.

I am basing the architecture of my image generation model off the ‘memory for latent representations’ (MLR) model proposed by Hedayati et al. in ‘A model of working memory for latent representations’ (1). MLR proposes that the human visual system encodes latent, ‘compressed’, representations of a visual stimuli’s ‘core components’ like shape and color. These latent representations can then be retrieved and decoded by the visual system and then placed into mental imagery at a specified location. This process would allow humans to imagine novel combinations of objects that they have previously seen separately but have never seen together.

This process is computationally modelled using a Variational Autoencoder (VAE), a neural network architecture proposed by Kingma and Welling in ‘Auto-Encoding Variational Bayes’ (2). This architecture is very similar to the theoretical architecture of the ventral visual stream, shown below in Fig. 6.

For my model I will build a similarly designed VAE, utilizing convolutional and batch norm layers as well as a set of fully connected layers to better improve the model’s ability to encode and decode images. My VAE will also use only one latent space in the bottleneck, and it will encode the combined shape and color representation of the input image.

Fig 6. MLR Architecture: (Fig.1, Page 2 Hedayati et al. (1)):

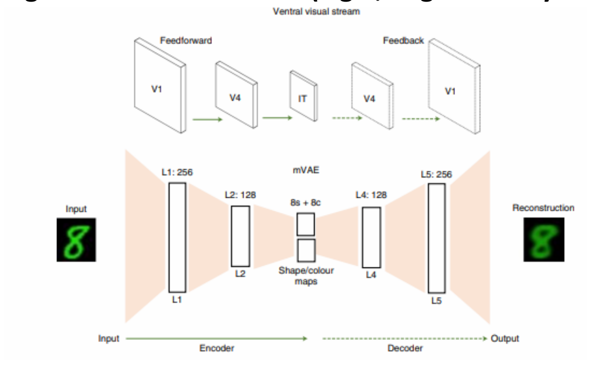
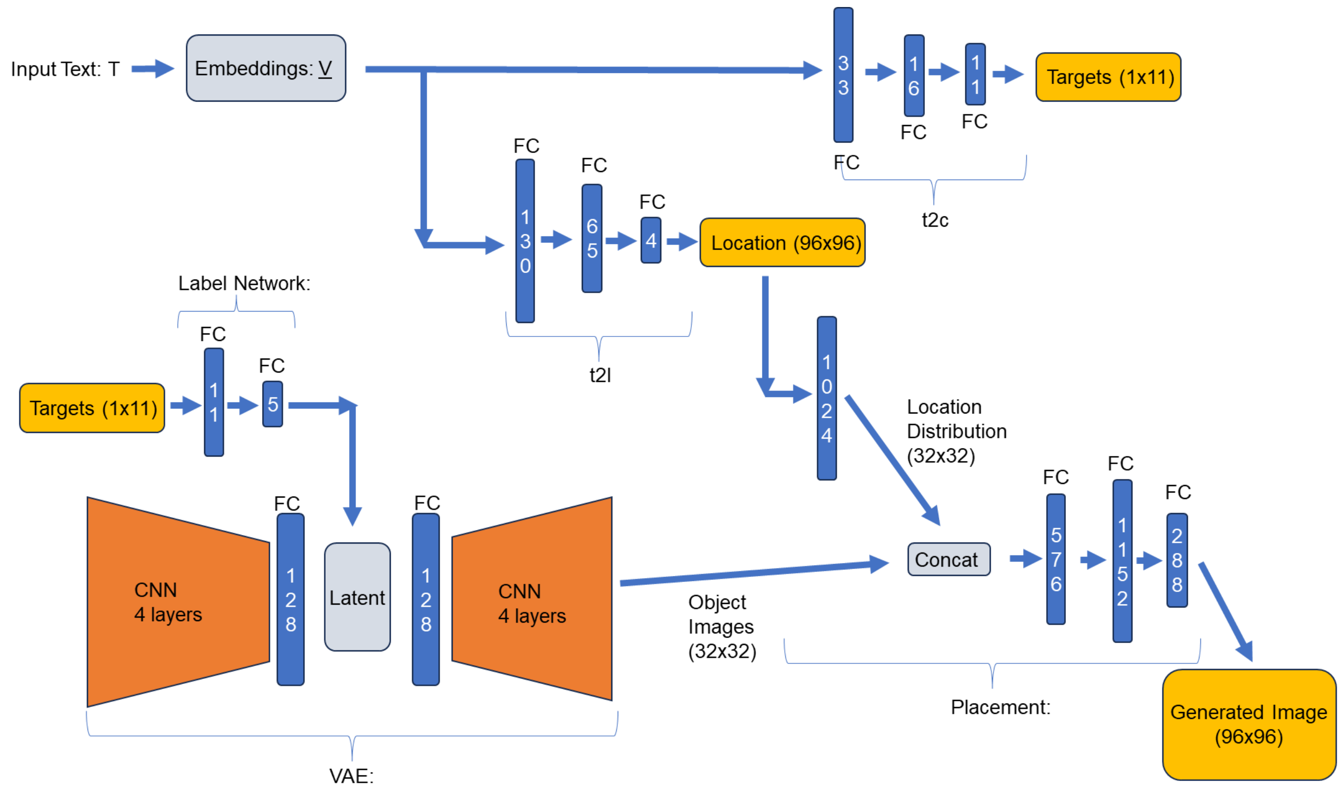


Image Generation Architecture:

Fig. 7: Complete architecture of the image generation model:



The model is written in Python using the PyTorch, NumPy, and scikit-learn libraries for construction and training. Each module in the architecture (embeddings_t2c, embeddings_t2l, VAE, label network, and placement) is written in its own Python script and has a corresponding training script as each module requires a unique training regimen and loss functions. Unless otherwise specified all layers use a ReLu activation function.

Embeddings: (embedding_t2c.py, Training_t2c.py, embedding_t2l.py, Training_t2l.py)

The embeddings module is comprised of two parts, **t2c** and **t2l**.

t2c: Corresponds to mapping a text input to a set of one-hot target vectors for the objects contained within the text description. The module consists of a scikit-learn CountVectorizer which performs tokenization and occurrence counting on the input text. The CountVectorizer is fit to a text Corpus of basic text descriptions of scenes each containing two cifar10 objects and a relationship between them. When a text input is tokenized, the CountVectorizer returns a vector of which known vocabulary words are contained within the text and where they occur, encoding the contents and context of the text. This vector is passed into a network of three fully connected layers, outputting an eleven-element vector representing which object targets are contained within the text. This network is trained by inputting a set of random words from the module's vocabulary (a set of 33 words containing the names of the objects (the cifar10 labels) as well as words describing their relationships), a Cross Entropy loss is used.

t2l: Corresponds to mapping a text input to a set of coordinates used to construct a set of (96x96) one-hot location matrices, used to place the object images into their corresponding locations in the output image. This module consists of the same CountVectorizer as **t2c** and a network of three fully connected layers outputting a four-element vector corresponding to coordinate pairs for two location matrices...

VAE: (cVAE.py, Training_VAE.py)

The VAE has an input and output dimension of 3x32x32 and a latent space dimension of 5. The VAE consists of three segments, an encoder, a latent space, and a decoder. The encoder and decoder are symmetric but in the reverse direction and consist of a series of convolutional/batchnorm layers as well as a fully connected layer to transition to/from the latent space, the encoder segment takes the input image and layer-by-layer compresses it to the reduced dimensions of the latent space, the decoder then takes a sampling of the latent space and layer-by-layer reconstructs the image to its original dimensions.

The latent space consists of two five-dimensional vectors, mu and log_var(represented by sigma). The sampling of the latent space passed to the decoder is calculated using the formula:

Sampling = $\mu + \left(e^{0.5 \cdot \sigma} \cdot \ randn(\sigma)\right)$

The VAE is trained using a subset of cifar10 images. All hidden layers use a ReLu activation function while the output layer uses a Sigmoid activation function, accordingly a Binary Cross Entropy loss function is used in training (2).

The VAE is trained on a subset of cifar10 images that includes only 20 samples of the distinct classes in cifar10, 200 sample images in total. This is due to the fact that within the entire cifar10 dataset there is a high variance between different sample images within the same class. This results in a high variance between latent representations of images within the same class, making it much harder for the label network to map one-hot target vectors to sampling vectors of the desired class. The tSNE (probabilistic dimension reduction) plots below in Fig.8-10 demonstrate how latent representations of distinct classes overlap with one another with greater frequency the more samples per class are used to train the VAE.

Fig. 8: A tSNE plot the latent representations of the 800 per-class cifar10 subset:

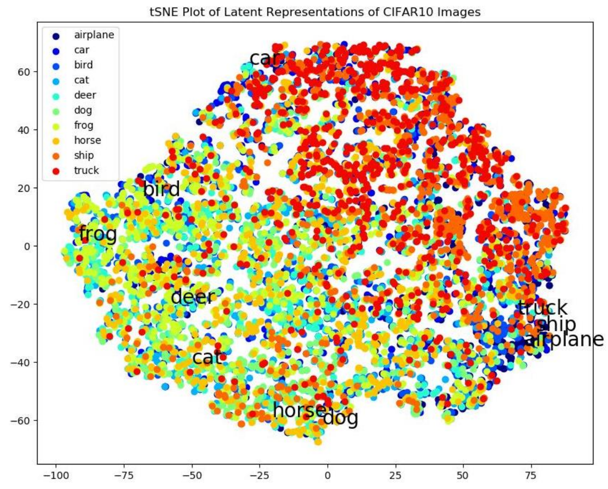


Fig. 9: A tSNE plot the latent representations of the 200 per-class cifar10 subset:

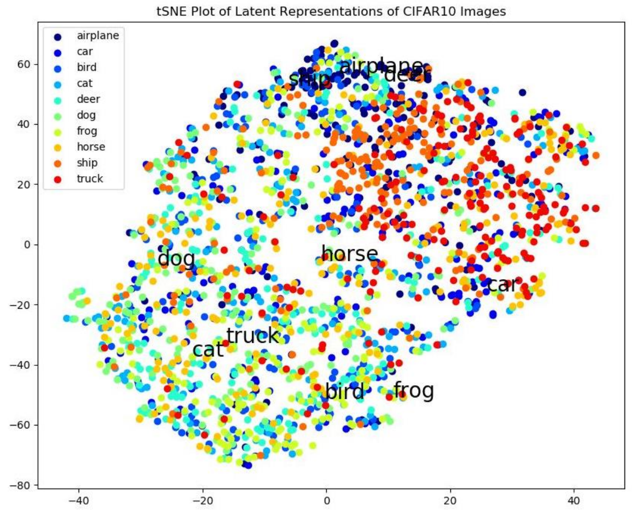
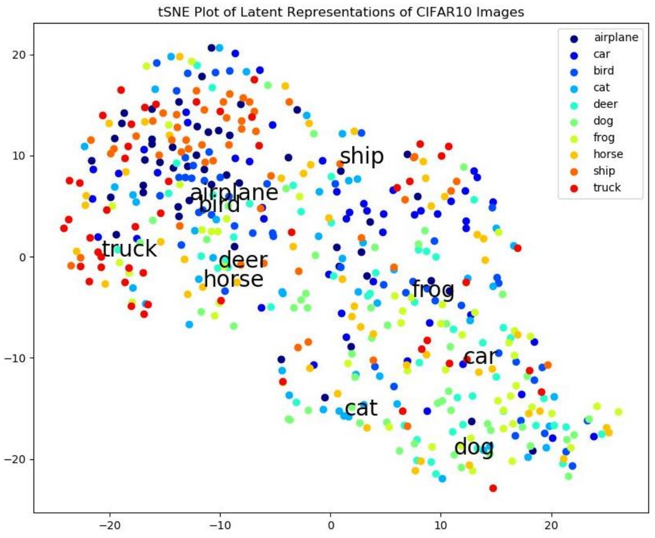


Fig. 10: A tSNE plot the latent representations of the 20 per-class cifar10 subset:



Label Network: (c_label_network.py, train_labels.py)

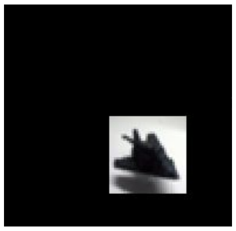
The label network maps one-hot target vectors to that image's latent representation within the VAE. It consists of two fully connected layers with an output of a 5-element sampling vector, matching the dimensions of the VAE's latent space. When an 11 element one-hot target vector is input to the network a corresponding sampling vector is output and then input into the VAE's decoder. The network is trained using MSE loss.

Placement: (placement.py, Training_pos.py)

The final module of the model is the placement network which takes as input a set of (96x96) one-hot location matrices and a set of (3x32x32) object images and outputs a (3x96x96) output image. The network first inputs the location vector through a single fully connected layer of output size 1024, this layer is trained to transform the location vector into a continuous location distribution of size (32x32), centered at the coordinates of the input location vector, this allows the placement network to learn a more precise placement. This location distribution is then concatenated with the corresponding object image and input through a network of three fully connected layers to a (3x96x96) output with a sigmoid activation function. This three-layer network has a unique architecture of layer sizes in order from input to output of 576 > 1152 > 288, this architecture allows the placement network to effectively map the location and image attributes into a higher dimensionality and effectively down sample to the output image.

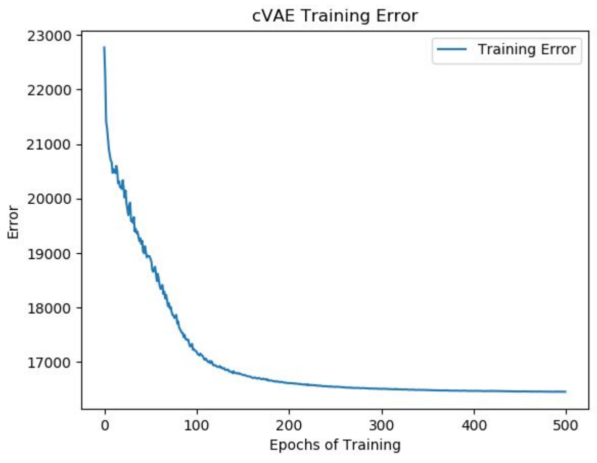
The placement network is trained with cifar10 images placed into a (96x96) blank background like the sample shown in Fig. 10. These training images are generated through my PadAndPosition transformation applied to the cifar10 dataset which returns a list: [the final image, the cifar10 image used, the location vector]. The network is trained using a binary cross entropy loss function.

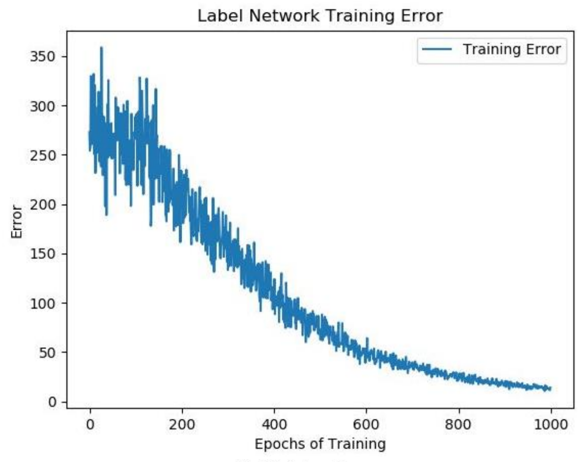
Fig. 10: A sample training image for the placement network:

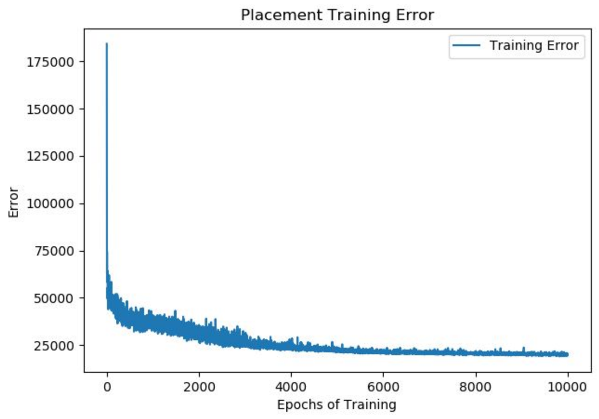


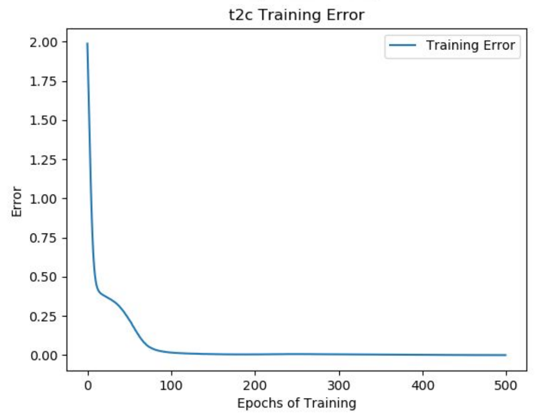
Results:

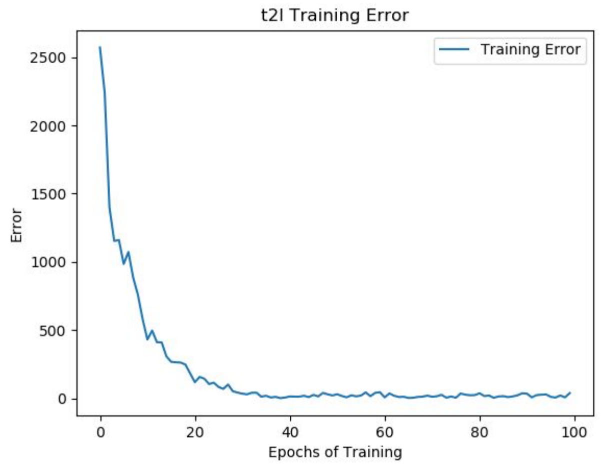
Training results:



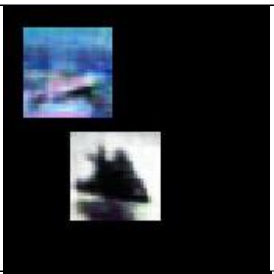
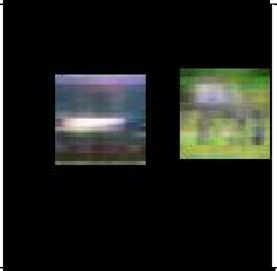
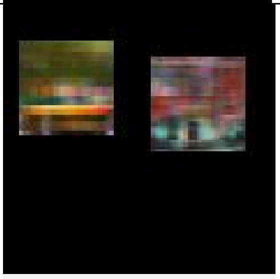
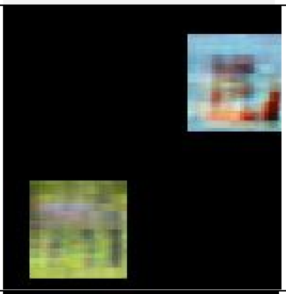
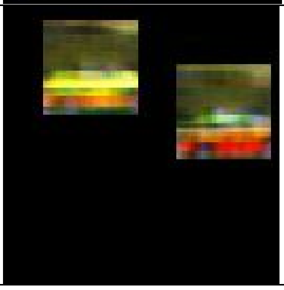








Text Description:	**In Corpus:**	**Generated Image:**

“a ship sales past an airplane”	**False**	
“A plane flies past a horse”	**True**	
“an automobile overtakes a truck”	**False**	
“a ship sales past a horse”	**True**	
“car overtakes a car”	**True**	

The resulting generated images demonstrate the model’s ability to generate rough scenes from modules that are trained on single images only. Additionally, there is no real noticeable difference between images generated from text descriptions found in the corpus used to train the Count Vectorizer and text descriptions not found in the corpus.

Conclusion:

The goal for this project was to build a basic model for an image generation mechanism for a text prompt that can compose learned representations of images into basic novel configurations, and the model has succeeded at that goal.

Moving forward the model would benefit the most from refinements in the text to location mapping module, training on a broader dataset, and adding additional latent spaces to the VAE. The largest drawback to using cifar10 images was the lack of context for each image, I think that the more context each training image contains, the easier it is for the model to create latent representations of its core components. A good candidate for this broader dataset is MS COCO.

Bibliography:

1: Hedayati, S., O'Donnell, R.E. & Wyble, B. A model of working memory for latent representations. Nat Hum Behav 6, 709–719 (2022). https://doi.org/10.1038/s41562-021-01264-9

2: Kingma, D., Welling, M. Auto-Encoding Variational Bayes. https://doi.org/10.48550/arXiv.1312.6114