Generating Garment Variations

Introduction

The objective of this project is to create a model that can generate variations of a garment from an uploaded image. Generating diverse garment variations has various applications in the fashion industry, such as virtual clothing try-on, personalized recommendations, and design ideation.

Research and Model Selection

Several models can be considered for this purpose. **Variational Autoencoders (VAEs)** are useful for image reconstruction and generation tasks due to their ability to model complex data distributions. **StyleGAN** excels at generating high-quality, realistic images with control over style and content, making it suitable for detailed and varied outputs. **Conditional Generative Adversarial Networks (cGANs)** extend GANs by conditioning the generation process on additional information, such as class labels, making them particularly effective for generating variations of different garment types.

Thus, Conditional Generative Adversarial Networks (cGANs) were selected as the most suitable model for this project.

Justification for cGAN Selection:

- **Conditional Generation**: cGANs can generate diverse variations of a garment by conditioning the generation process on the input image and the desired class label (e.g., T-shirt, pants, shoes).
- **Versatility**: cGANs have been successfully applied to various image-to-image translation tasks, including fashion-related applications such as virtual clothing try-on and fashion recommendation.
- **Scalability**: cGANs can be trained on large-scale datasets and have shown promising results in generating high-quality and diverse images.

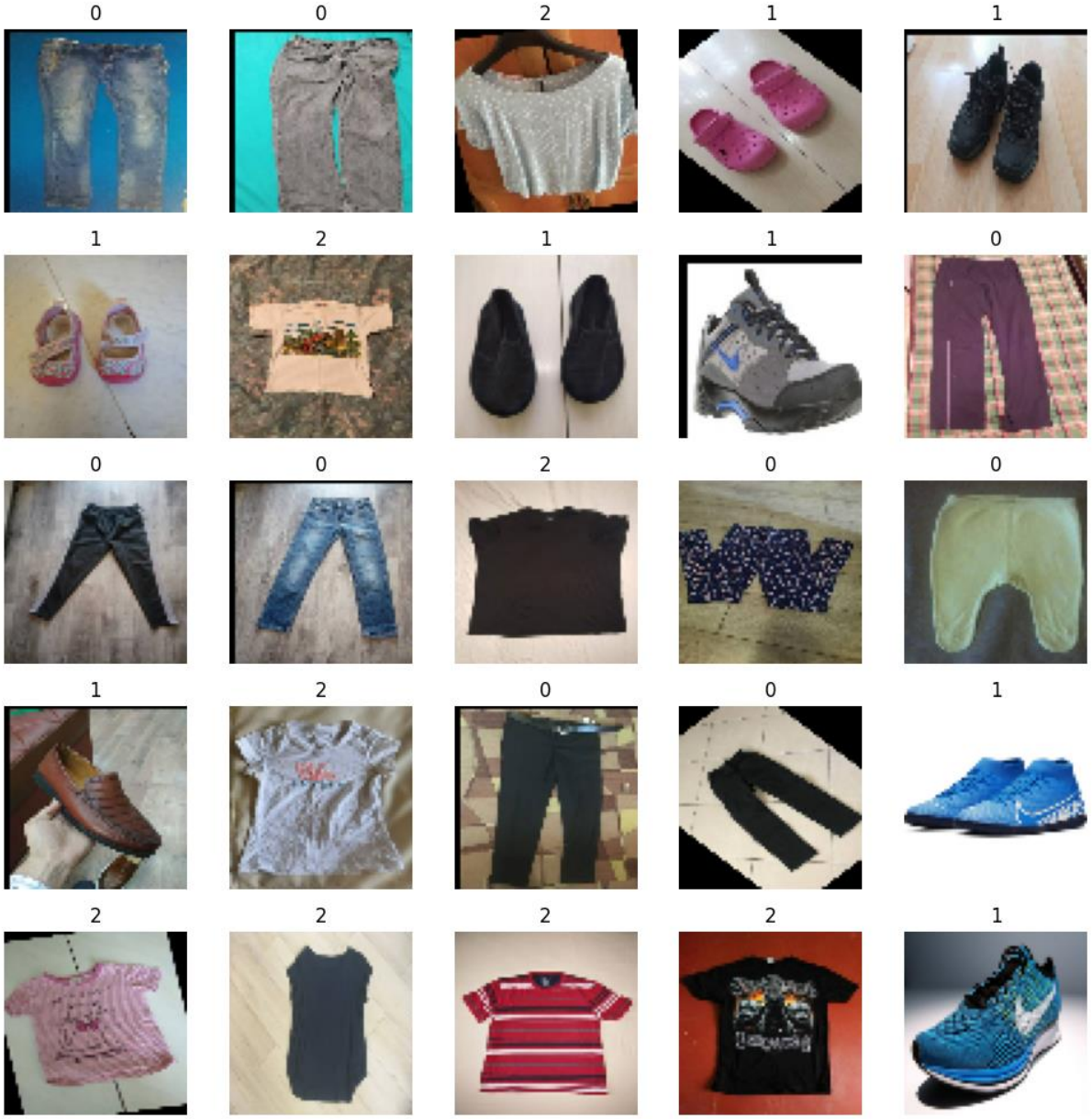
Dataset Preparation

- **Data Collection and Labelling**: Obtained images for 3 different classes namely T-Shirt, Pants and Shoes from the internet. The task of training cGAN requires a good number of images for generation of high-quality images.

- **Data Augmentation**: To increase the diversity of the dataset and improve the model's generalization capabilities, data augmentation techniques were applied, including flipping, rotation, and translation.

- **Libraries Utilised**: Pandas, Scikit-learn, Tensorflow and Matplotlib have been used to read, preprocess, and visualize our dataset.

Finally, the dataset comprised of more than 2700 instances per class. Below is a visualization of the images in the dataset with their corresponding labels.



Class Name	Label
Pants	0
Shoes	1
T-Shirt	2

*The instances per class and the number of classes can be increased but I've restricted to these because of computational limitations.

Model Training

Creating the cGAN Architecture:

Python, Tensorflow and Keras have been utilized for the creation of the cGAN architecture.

GANs can be extended to a conditional model by providing additional information (denoted as y) to both the generator and discriminator. This additional information (y) here are the class labels. In the generator, the prior input noise (z) and y are combined in a joint hidden representation.

Loss function and Optimizers: We first define the Loss function and optimizer for the discriminator and generator networks.

- Binary Cross-Entropy Loss (bce loss) is suitable for distinguishing between real and fake data in GANs.
- Adam optimizer is employed to update the trainable parameters.

Discriminator Network: It's work is to distinguish between real and fake data from the dataset and data generated by the generator.

- The discriminator takes both an image and a label as input. Input layer takes single value, which represents a label or class information. It's embedded into a 50-dimensional vector using an embedded layer.
- The label is reshaped then concatenated with the input. The combination of label information with the image is a way to provide conditional information to the discriminator.
- Convolutional layers are applied to concatenated input. These layers extract the features.
- The feature maps are flattened and dropout layer is applied to prevent overfitting. The flattened and dropout processed features are connected to the dense layer with a single neuron and a sigmoid activation function. The output is the probability score representing the input data is real or fake.
- A model is created using TensorFlow API, considering both image and label as input and producing the discriminator's output as the final layer.

Generator Network: It is the neural network responsible for creating (or generating) new data. The generator takes a label and noise as input and generates data based on the label.

- The input layer is used to provide the label as input to the generator.
- Embedding layer converts the label (i.e., single value) into vector representation of size 50.
- the input layer is used to provide the noise (latent space input) to the generator.
- The label are reshaped and the concatenated with the processed latent space.
- The merged data goes to a series of convolution transpose layers.
- The final output layer convolutional layer with 3 channels (for RGB color). It produces an image with size '64x64x3′ as the desired output.

Train Step: We create a train step function for training our GAN model together using Gradient Tape. Gradient Tape allows us to use custom loss functions, update weights or not and also helps in training faster. The generator and discriminator are updated alternately for training the GAN model.

Google Colab's GPU has been utilised in training the cGAN Model.

Parameters Setting and Tuning:

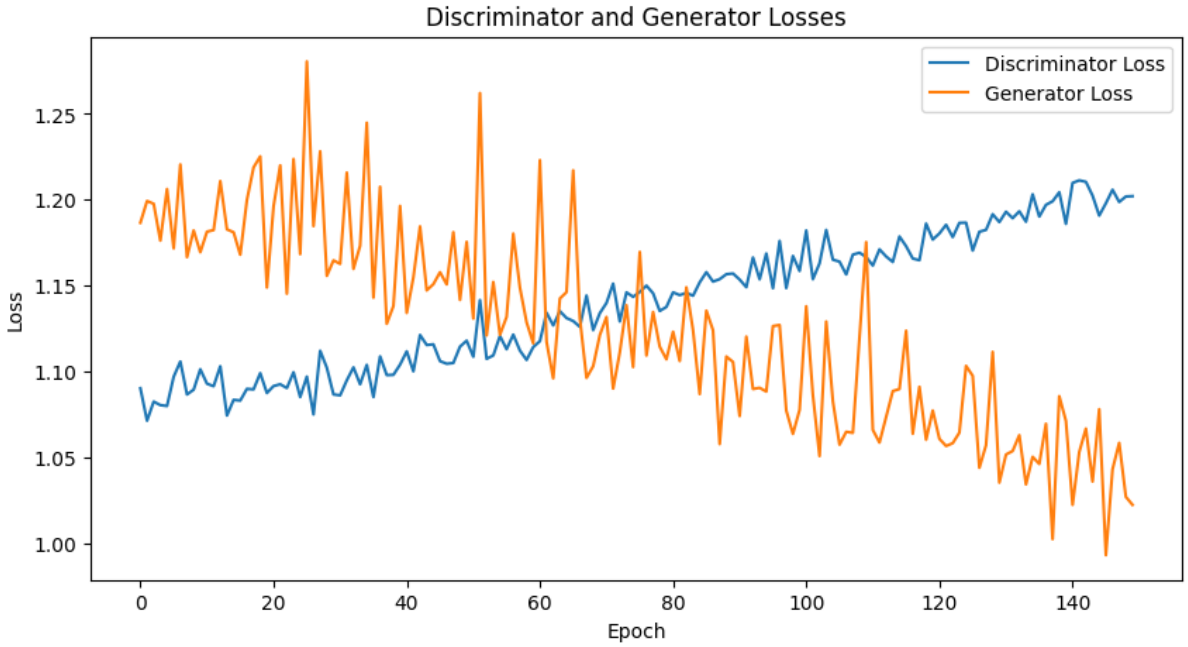
- **Hyperparameters**: Learning rate of 0.0002 provided the best results and batches of size 32 were taken for training for 150 epochs (Model performance was still improving but epochs were restricted due to computational limitation).
- **Regularization**: Dropout and batch normalization were applied to stabilize training.
- **Conditional Input**: Class labels were concatenated with the input noise vector to condition the GAN.

Monitoring and Evaluating:

Loss Functions: During the training process, both the generator and discriminator networks are optimized using appropriate loss functions. Monitoring these loss functions is crucial to ensure stable training and prevent issues like mode collapse.

- **Generator Loss:** Monitored the generator loss, which represents how well the generator is fooling the discriminator. A decrease in generator loss indicates that the generator is becoming better at generating realistic samples.
- **Discriminator Loss:** Monitored the discriminator loss, which indicates how well the discriminator can distinguish between real and generated samples. A decrease in discriminator loss suggests that the discriminator is becoming more accurate in its classification.

By keeping track of both generator and discriminator losses, we maintain a balance between the two networks and prevent one from overpowering the other, leading to stable training.



Above is the visual representation of the generator and discriminator losses over 150 training epochs.

Evaluation Metrics

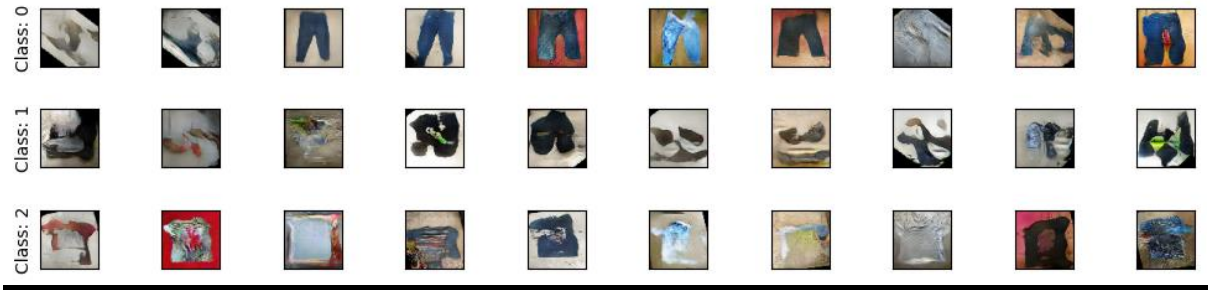
Inception Score: The Inception Score (IS) is a metric used to evaluate the quality and diversity of generated images in generative models. The IS consists of two components: the mean score and the standard deviation.

- **Mean Inception Score**: The mean IS indicates the overall quality and diversity of the generated images. A higher mean IS suggests that the generated images are of higher quality and exhibit greater diversity in terms of content and style.
- **Standard Deviation**: The standard deviation of the IS reflects the variability or consistency of the generated images across different splits or batches. A lower standard deviation implies more consistent performance across evaluations.

In our case, the mean Inception Score (IS) is approximately 1.0005, and the standard deviation is approximately 0.00025.

A mean IS of approximately 1.0005 indicates that, on average, the quality and diversity of the generated images are reasonably good.

Visualization of result: Below is the result of our cGAN model for our 3 classes in consideration.



Conclusion

Using cGANs for generating garment images is a well-justified approach. The ssteps for data preparation, model training, and evaluation ensure that the model produces high-quality and diverse images. Established metrics like IS and FID, along with visualization provide a comprehensive evaluation of the model's performance.

Links:

- **Google Colab file:** https://colab.research.google.com/drive/1E9Zb5aXK5UQaEDQjnHX9YP8X1Z0bz7sH
- **Dataset:** https://drive.google.com/file/d/1_HdoYVJagnB3k7ttAYm3l7o0VgYuLSok/view?usp=sharing
- **Csv file for labels:** https://drive.google.com/file/d/13Zkqf2O4RdxQDGqF_99AqK0_tiTTfyHr/view?usp=sharing