Project Group:
David Eskilson (deskilso)
Brady Garrison (bpgarris)
Jake Lippert (jlippert)
Sam Salseda (ssalseda)

Old Soles

A Sketch-to-Image Shoe Generation Model

Brown University, CSCI 1470

Introduction

We will be implementing a deep learning model in TensorFlow which generates images of shoes based on sketches. Our implementation will be based on the HAIFIT model by Jiang et al., which was created using PyTorch. The model architecture is a modified Generative Adversarial Network (GAN). We will first attempt to train the model on the same dataset used in the paper, before testing the model's broader applications using the UTZappos dataset. We implemented the model using TensorFlow/Keras, while the original implementation was in PyTorch.

The HAIFIT model differs from prior GAN based image generation models in two important respects:

1. A pyramidal Generative Adversarial Network structure is used to train the model on images of different resolutions
2. An LSTM-based Multi Factor Feature Encoder (MFFE) is used to create a latent space of both the sketches and images to improve feature representation and consistency.

The major stakeholders in our project are the end consumer and the designer of popular shoes. On one hand, an easy path from sketch to design could cut down on the costs associated with the creation of a new product, streamlining the process and lowering cost for the consumer. This advancement could also kill an industry of designers whose jobs are based in the transition from sketched images to more developed designs.

Challenges

The main challenges that we faced throughout our implementation were:

1. Converting between the syntax of Pytorch and Tensorflow/Keras as we implemented the original model. We encountered numerous normalization schemes and layers that we did not understand and that did not have a corresponding Keras layer. We took advantage of online documentation and examples from lab to make this transition easier.
2. Understanding the structure of the GAN Blocks and the progression from one GAN Block to another. To resolve this challenge, we reread the HAIFIT paper in closer detail and experimented with our implementation to better understand the progressive growth model.
3. Encountering shape issues since values that we had initially hardcoded in were not sufficiently generalizable to a situation with multiple potential shape inputs. This required hours of debugging and investigating individual layers of our model to resolve.
4. Finally, we had issues when training our model, where whether run on small or large batches and a small or large numbers of epochs, the loss consistently went up. We resolved this training issue by correcting bugs in preprocessing, changing hyperparameters, and debugging our loss functions.

Sources

Jiang et al. "HAIFIT: Human-Centered AI for Fashion Image Translation" In CVPR, 2024
Radford et al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" In CVPR 2016
Yu and K. Grauman. "Fine-Grained Visual Comparisons with Local Learning". In CVPR, 2014.
A. Yu and K. Grauman. "Semantic Jitter: Dense Supervision for Visual Comparisons via Synthetic Images". In ICCV, 2017.
Zhang et al.. "Toward Multimodal Image-to-Image Translation" In CVPR 2018

Methodology

Preprocessing / Dataset

We used the UT Zappos50k database to get a wide array of shoe images as the labels. Since this dataset did not come with corresponding sketches, we needed to generate these images to serve as the inputs.

We borrowed an image-to-sketch module from StyleMe, an existing deep learning project. After training the module, we processed 50,000 images of shoes downloaded from the UT Zappos50k database. The resulting database had a total of 100,000 images which could be broken down into 50k pairs. To complete the preprocessing of the data, we ran a Python program that normalized the size of the images and converted them into tensors. 35,000 images were saved as testing data and a random crop was applied to mimic the crop applied in the paper. 15,000 images were unmodified and saved as testing data. All images were saved in Numpy files to be accessed later.

Architecture

Our model is comprised of:

- A Progressive Growth GAN model - Progressively trains the generator/discriminator on images of increasing resolution (Figure 1)
- A Multi-scale Feature Fusion Encoder (MFFE) - Composed of a Shallow Convolutional Module and an LSTM-inspired Abstract Feature Representation Module, which together learn long-term dependencies between various features in the input image (Figure 2)
- A Cross-level Skip Connection Module (CSCM) - placed between levels in the PGGAN to extract features from previous generators, which are then used as attention maps for the current Generator.

Figure 1: Overall model / Progressive Growth GAN Architecture

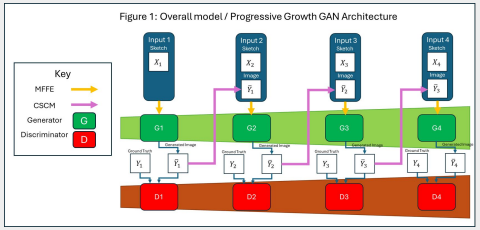
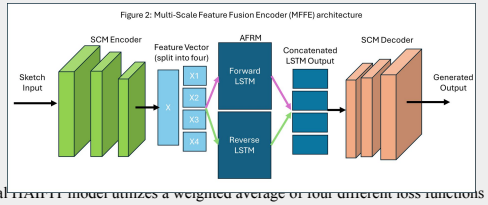


Figure 2: Multi-Scale Feature Fusion Encoder (MFFE) architecture



Losses

The original HAIFIT model utilizes a weighted average of four different loss functions that can be adjusted with hyperparameters. We decided to use three of these loss functions, as the perceptual loss function involved the use of a pre-trained model. The functions that we used for our model are:

1. L2 loss, which checks the similarity between the pixel features of the generated and real images.
2. Adversarial loss, which is the traditional min-max loss function for a GAN network, to ensure that the generator is trying to generate realistic images and that the discriminator is trying to discern real images from generated images.
3. Style loss, which utilizes Gram matrices (the multiplication between a matrix and its convolution) to evaluate the similarity between stylistic textures and details of the real and generated images.

Training

Due to resolution of a variety of errors related to image visualization and loss handling, we only had the opportunity to perform limited training. We trained for 10 epochs on 150 images for this poster.

Results

Metrics

Accuracy metric: we will be using PSNR (Peak Signal to Noise Ratio) and SSIM Structural Image Similarity Index) as our primary evaluation metrics. These metrics compare the noisiness and structure of images in order to ensure that they approximately match reference images.

Experiments

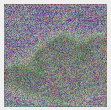
We use a portion of our data that the model has not trained on and evaluate the accuracy of the model on that data. Taking specific sketches, both from the data set and self made, and seeing if our model has reasonable generative outputs would also be a great way to measure the model's effectiveness.

Visualization/ Outputs

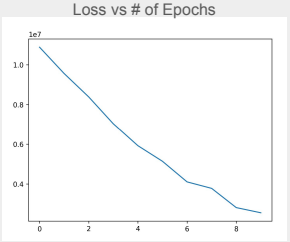
Input

Sketch

Output



Loss vs # of Epochs



Reflection/Discussion

We are ultimately happy with how our project turned out, since we were ultimately able to write code that addressed our base, target, and stretch goals.

Our model architecture did end up roughly being how we expected it to, since we followed the architecture of the paper to include the SCM encoder, AFRM, Residual Blocks, and SCM decoder. We also included concatenation of a previous output with the next output per the prescribed architecture of the paper.

While the overall approach of our model remained consistent with our plan and with the paper, we did need to make some pivots/cuts. Specifically, the perceptual loss implementation would have required loading in an outside model and would have added extensive overhead, so we decided not to include it for the sake of time.