Visual Essence: Text to Image Modality Translation using Image Homography Networks

Micah Reich*

Georgia Institute of Technology, The Weber School
April 23, 2021

Abstract

Image classification and image generation are long-standing problems in the fields of artificial intelligence and computer science research. While many modern approaches to these tasks make use of deep neural networks like Generative Adversarial Networks and Convolutional Neural Networks, we demonstrate that simple image recognition and generation may be accomplished with classical methods in linear algebra, including Principal Component Analysis and K-means clustering.

1 Introduction

The task of text-to-image translation has been explored in great depth recently, most notably with the use of Transformer networks–often used to turn text into latent representations–and GANs, which are able to learn a mapping from latent space to image-space. Many GANs rely on creating new images from latent variables alone, though the use of conditional GANs and auxiliary-classifier GANs has been explored in order to condition the result of the GAN output rather than rely completely on the stochasticity of the latent input. In addition, researchers have explored the use of conditional GANs in order to perform image-to-image modality translation, conditioning the input to the network on an initial image, then converting the input into a different modality, such as converting a real image into a painted version of the image.

Version one of Visual Essence sought to combine feature extraction techniques and image-conditioned GANs in order to create a unified icon representation from three separate icons used as input. Visual Essence presents a fairly niche yet computationally inexpensive method of text-to-image translation as the task is split into many parts with a small GAN or homography network responsible for image composition rather than relying on a large-parameter LSTM-GAN3 or Transformer to perform the entirety of the task.

A key challenge that motivates Visual Essence stems from a lack of data: the network architecture is motivated by the large number of icons available without available data regarding mappings between textual descriptions and image representations. With both the task of text-to-image translation and the lack of data mapping text descriptions to image representations in mind, we decided move away from a Transformer or LSTM-GAN based architecture in favor of a simpler, distributed architecture that split the task into smaller pieces that worked well with the data at our disposal.

2 Neural Networks

2.1 Introduction

Deep neural networks have become very popular tools for machine learning tasks in recent years due to the wide availability of large datasets and the abundance of increasingly powerful computers that can perform the operations necessary for deep neural networks to function. In essence, machine learning and neural networks behave as function approximators. Linear regression serves as a canonical example of machine learning: using various data points, an affine function is fit to best fit the data and minimize the error between each data point and it's predicted value by the function. A simple linear regression model takes as input one parameter, x, and returns one output, y, using two weights in order to transform the input into the output, the slope of the line and the y-intercept.

*mreich@gatech.edu

2.2 Backpropagation and Gradient Descent

Modern training methods for many deep neural networks involve a forward and backwards pass over the network, slowly improving the model's weights until a local minimum in error is achieved. For linear regression, a forward pass involves multiplying the input by the slope then adding the y-intercept before returning the final value. Most neural networks are initialized with either zero or random weights before training begins in order to provide an initial guess. After an initial guess has been made during the training phase, the model must improve; the error of the model is known as the loss and is measured via some error function–for our example, the loss function is most often the Mean Squared Error function (the sum of the squares of the difference between the true and predicted value of each data point after being fed through the model and returning an output).

$$E := \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - (w_0 x_i + w_1)\big)^2$$

Mean Squared Error loss function; returns the average square of the difference between the true and guessed value of each data point; weight zero is the slope and weight one is the intercept of the linear model.

> Each data point generates an output from the model, and this output is used to determine the error of the model after each training iteration. At the end of each iteration, the model performs a backwards pass to tune the model weights and improve the model's performance over the dataset. The backwards pass, called backpropagation, uses a method of iterative optimization known as gradient descent in order to optimize each model weight and thus improve model performance. As the name suggests, gradient descent updates each model weight according to its effect on the model's error by considering the gradient–similar to the derivative–of the error function with respect to each weight. Since the linear model is differentiable with respect to each weight, the derivative of the error with respect to a given weight may be calculated via the chain rule; since the error is a function of the model's prediction, and since the model's prediction is a function of the model's weights, a gradient vector of each partial derivative of the error w.r.t each weight may be calculated. In effect, gradient descent changes each weight by a small amount to decrease the error by analyzing how the error function changes as each weight changes. The magnitude of the change made to each weight is denoted by alpha, or the learning rate, which is a hyperparameter that may be tuned to get the best results in training–if alpha is too large, the model might overcompensate or miss the local or global error minima, but if it is too small, training may take far too long.

$$w_n := w_n - \alpha\frac{\delta}{\delta w_n}E(w_o, \ldots, w_n)$$

Gradient Descent update rule; modifies a given weight value according to the partial derivative of the error function E with respect to the weight proportionally to a constant known as the learning rate alpha; the objective of training is to minimize E by changing the weight values.

While a closed-form solution to linear regression may be found via the Normal Equations, many problems in deep learning and many deep neural networks' optimal weights do not have a known, closed-form solution, so the approach of gradient descent allows for iterative weight optimization. The forward and backward passes as well as the gradient computations and weight updates are performed many times until a local or global error minima has been reached, at which point the training stops and the model's weights are saved, ready to be used by the model at test time. Neural networks are meant to be specific enough to fit the training dataset well but general enough to predict new, unseen data accurately as well. The more training data supplied, the more general the model may become due to the variety of examples it has seen. While linear regression is a fairly simple application of machine learning methods, the same principles apply to much larger and more complex neural networks.

2.3 Deep Neural Networks

Deep neural networks get their name from the presence of many sequences of layers present within the model–while linear models have only one layer with one neuron or node that has two weights, neural networks capable of image recognition or machine translation have millions of weights and sometimes thousands of layers, each meant to extract salient features of the input in order to create an accurate output. The accuracy and precision of a neural network scales as the number of weights increases, meaning larger networks that have many layers with many nodes are able to tackle larger datasets and more complex problems.

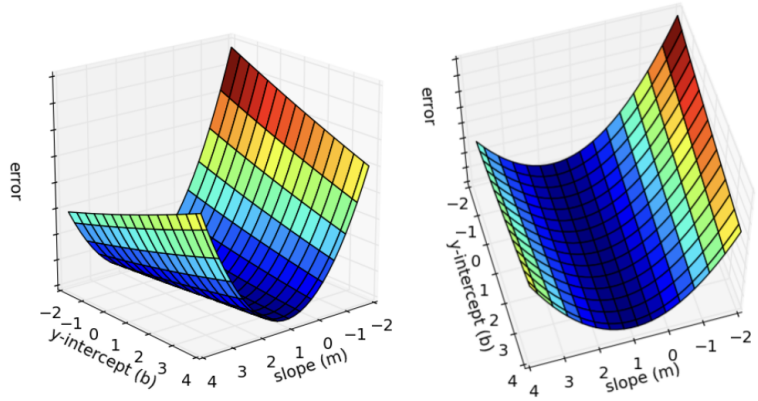

Figure 1: Visualization of Error as a Function of Slope and Intercept; Gradient Descent seeks to reach the lowest part of the dark blue region of the error curve, or the global minimum.

One notable type of deep neural network is a convolutional neural network (CNN). CNNs are often used to perform feature extraction from images and image recognition; CNNs learn which features to extract from images as well as which features belong to which categories in order to make predictions. Black and white images may be treated as a matrix of values denoting the darkness of each pixel in the image that is fed into a neural network; color images are treated as 3D matrices of red, green, and blue pixel values. A matrix of weights, known as a filter, is applied to the image to extract notable features from the image, converting the image to a lower dimensionality as it progresses throughout the network where a final layer of neurons interprets the transformed image and predicts a probability distribution in order to classify it into a discrete class. As is discussed later, CNNs may also be used to predict a continuous output rather than classify an image into a category.

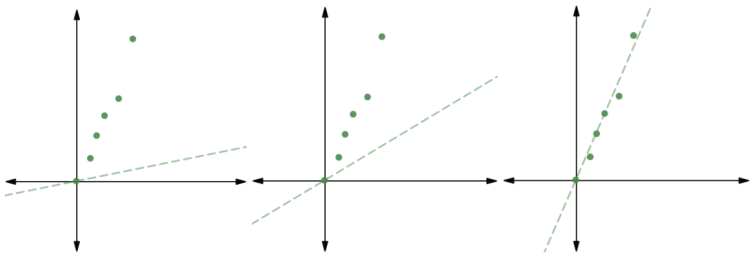

Figure 2: Convergence of a Linear Model with One Parameter; left to right displays the training data accompanied by the initial linear model (with a randomly initialized slope weight) followed by the model in the middle of training followed by the final model achieved through gradient descent optimization; the particular line in the figure has a slope of 2.2.

3 Methods and Implementation

For the task of Visual Essence, we seek to turn two or three sentence descriptions into emoticon representations of the textual biography. The data allotted to the project consisted solely of textual descriptions collected from 1000 people via Amazon Mechanical Turk service as well as a large database of titled icons and emoticons. As previously mentioned, no examples mapping a textual description to it's image representation existed in the dataset, so the function that we seek to learn is implicit and cannot be learned via a strictly supervised model due to the lack of data.

To separate the task into smaller, more digestible and trainable networks, the workflow is as follows; first, we extract 3 salient words from the biography using either Term Frequency-Inverse Document Frequency (TF-IDF) search or parts-of-speech tagging + TF-IDF. Next, we search over our emoticon database using the extracted keywords to find a representative sample of 3 icons which we wish to fuse together into one representation. The composition network was implemented as both an "image-homography" network and a GAN; model testing proved the homography network to perform better than the GAN and with more computational efficiency due to the relative size difference between the two models when comparing the number of parameters.

3.1 Salient Feature Extraction

First, TF-IDF works by assigning a "unique-ness" score to each word in the corpus by analyzing its frequency relative to other words in the document; the number of occurrences of the term is multiplied by the log of the ratio of total documents to the number of documents containing the term. When the number of documents is taken to be constant, and a large number of documents contain the term, the logarithmic term will be close to one and the word's score will be close to zero. If the number of documents containing the term is small, the logarithmic term will be greater than 0 and the word's score will increase. If the logarithmic term is taken to be constant, the word's score is proportional to the term frequency within an individual document, meaning the TF-IDF measures the relative rarity of a word in the entire document dataset, assigning high scores to words that appear frequently in one document but infrequently throughout the dataset. This method, on its own, does a mediocre job of performing salient feature extraction, however in a document where a user uses an arbitrary word, TF-IDF assigns said word a high score, when in reality this word may be irrelevant to the task of representative word selection. Combining TF-IDF with parts-of-speech tagging, however, performs much better, since we can assist TF-IDF by pre-selecting nouns or adjectives to search over.

$$w_i, j := \mathrm{tf}_{i,j} \cdot \log \frac{n}{\mathrm{df}_i}$$

Term Frequency-Inverse Document Frequency Score; the word score for a word i in document j is proportional to the term's frequency within it's document and inversely proportional to its frequency within documents of the remainder of the text corpus.

3.2 Image Generation: Part I

Next, we must turn the words extracted from the biography into icons. One dataset we employed was the open-source icon dataset known as The Noun Project. Using TNP API, we were able to search through millions of icons in order to select the best icons that represented the query words. One unique challenge presented itself in the stylistic consistency of icons selected from the icon dataset; The Noun Project icons are black-and-white and come from many different artists with different artistic styles, and when selecting icons from search results, we noticed that certain icon triplets did not seem to fit with one another well; the difference in styles mostly emerged from having icons that were mainly filled-in black with icons that were only outlined in black with thin lines.

To enforce stylistic consistency, we implemented a simple K-means clustering model to discriminate between icon styles. Using a random dataset of around 5000 icons from TNP, we implemented and trained a clustering model to group icons into one of three categories depending on the stroke width of each icon. As K-means is an unsupervised learning algorithm, we based the label of each icon on the median number of consecutive non-white pixels in each row and column of the image in order to determine stroke width. Using this model allowed us to select icons of one style so that our final image would be the most aesthetically pleasing.

An important assumption that was made while designing and training the image fusion networks was the idea that a good conglomeration of images should be stylistically similar to a single icon, as the conglomerate is meant to serve as a single visual representation of a person much like how a regular, single icon represents a single idea. In practice, this meant that we trained neural networks to create icons that would fool an icon-detection network; if the amalgamation passed through the network trained to identify single icons with sufficient accuracy, we know that our final amalgamation looks like a single icon and is thus arranged well.

First, we trained a CNN on the task of binary image classification using icons from TNP in order to create a "discriminator network" that was able to sufficiently tell the difference between a real, single icon and a random amalgamation of 3 random icons placed with no heuristic guidance. This network is capable of telling apart a single icon and a bad, random conglomeration of three icons. Next, we experimented with two "icon-placement" networks that sought to find the best arrangement of icons in order to fool the discriminator. The first network was a conditional-generative adversarial network with a pretrained discriminator (as described above).

> GANs operate within the bounds of an adversarial, two-player game with the ultimate task of image creation. The discriminator's job is to take in two images, one real and one fake, and tell the two apart. The generator's job is to create images that look real enough to fool the discriminator; in the end, the generator has become skilled enough such that the discriminator fails to tell the difference between true image samples and samples created from the generator, and the generator has won the game. In a canonical example, a GAN might seek to learn how to create realistic images of handwritten digits. In the beginning, the generator fails to create anything that looks realistic, and the discriminator has an easy time telling the difference between real and fake. Though, as the generator learns which images do and do not work, it begins to create better and better images until it has won.

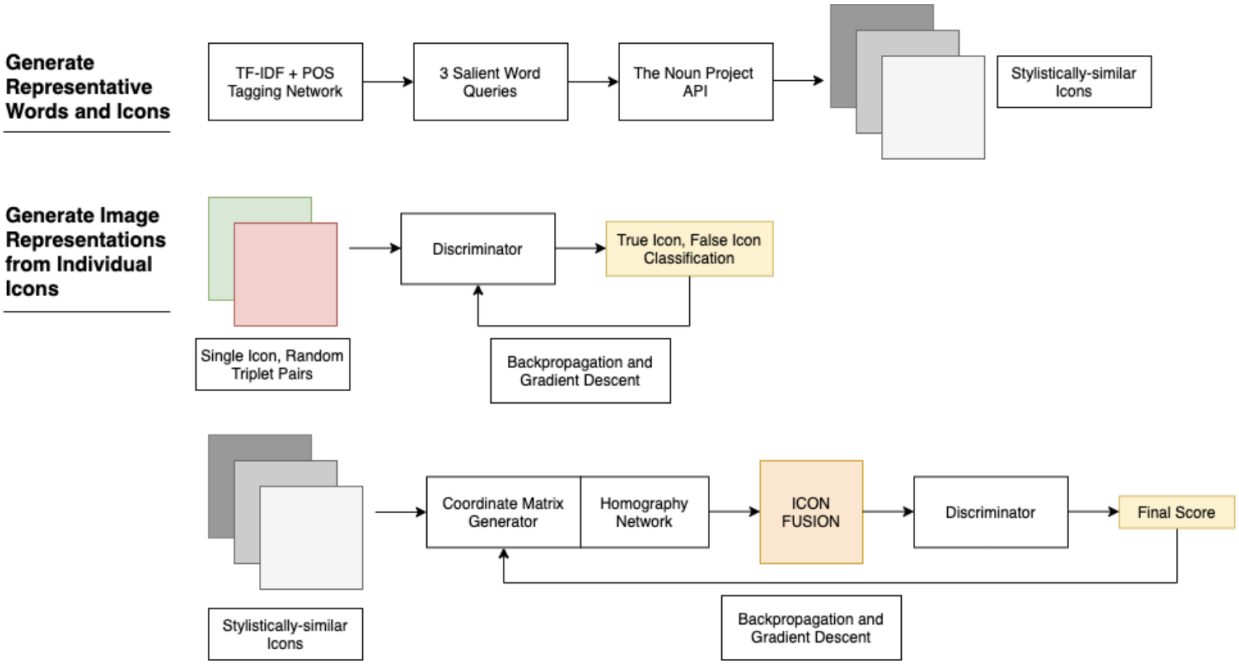

Figure 3: Full network diagram; salient feature extractor (top) and image generator (bottom).

The GAN was conditioned on the three input images + a latent vector and utilized an encoder-decoder network structure using convolutional and deconvolutional layers to first compress the input images into feature vectors, concatenate them with the latent vector, then upsize using deconvolutional layers to achieve a final, single icon. In practice, this method failed, because the GAN was trying to learn a representation of the best image amalgamation with a discriminator network that was trained only on random amalgamations and single icons. Simply put, the GAN did not have enough information to update the generator weights and improve the discriminator score since we lacked a true dataset of the representations we sought to learn. GANs are most powerful when the discriminator is looking for exactly what the generator is producing–trying to manipulate the architecture to create a true generator with no true dataset often fails in practice.

3.3 Image Generation: Part II

To remedy this, we decided to approach the generator network from a simpler standpoint. Instead of trying to create an image from a featurized version of each icon, we created a network that took as input three images, featurized them, but instead of then upsampling to a new image, the network returned a three by two matrix representing three (x, y) pairs–the locations of the centers of each image on the final icon amalgamation. Afterwards, a layer with no gradients would perform a deterministic placement process, placing the three images together on a final canvas according to the network's coordinate outputs. Finally, the canvas and images were fed into the aforementioned discriminator to judge the quality of the amalgamation. This generator network–while not a "true" generator in the sense of the word in the field of generative adversarial networks–performed much better than the previous upsampling method. One challenge presented itself in gradient propagation: since the algorithm of backpropagation requires each layer in the neural network to be differentiable, the deterministic placement layer that pasted each image on the canvas had no defined gradient, creating an untrainable network.

However, since the pasting process is indeed deterministic, the gradient may be treated as a constant ¿ 0 since the placement layer has a constant influence on the final image regardless of the vector input (if the gradient were zero, all previous gradients would be zero via chain rule computation and training would become impossible). The training process for this GAN involves using the output from the discriminator as the error function; propagating this score backwards, the generator updates it's weights to create a better coordinate matrix that makes the final icon arrangement better looking. A diagram of the models is pictured below:

4 Results

While many results from the first network proved unsatisfactory, many from the second iteration of the network provided some interesting creations:

Science is my passion and I work at a hospital as a nurse practitioner. I live with my kids, husbands and my puppy.

This result suffers from odd word choice as a result of using purely TF-IDF salient feature extraction.

I'm a communications major in college and I work at a farmers market.

This result suffers from poor icon placement and has many artifacts from the layering network as a result of poor convergence by the coordinate-prediction network.

I'm a 23-year old girl living in New York city. I like to hang out with my friends on the weekends and walk around the city.

I'm a mellow guy who likes to code on my computer and I'm interested in mathematics.

Science is my passion and I work at a hospital as a nurse practitioner. I live with my kids, husbands and my puppy.

5 Conclusions and Future Work

While the results from Visual Essence are not as complex as other text-to-image networks, we present a unique and data-aware solution to the problem of modality translation. Recent breakthroughs in text-to-image translation–and the reverse task–by OpenAI in their DALL-E4 and CLIP networks present a unique opportunity for the task of modality translation by using Transformer networks to perform the tasks of feature extraction and image creation all in one network; however, CLIP and DALL-E rely on large datasets of text-image pairs that are not readily available. Future work for Visual Essence lies in creating a more advanced image fusion network capable of creating images that resemble real people in real environments while remaining conscious of the strict data bottleneck that many consumers, companies, and researchers must face.

6 Acknowledgements

I would like to thank Dr. Devi Parikh of the Georgia Institute of Technology and the Visual Intelligence Lab for all her assistance and support in guiding me through this project. I would also like to thank Abhinav Moudgil for his advice regarding the implementation and mathematics behind generative networks. Thank you to Mrs. Shalania Van-Dyke of The Weber School for your unwavering encouragement and support of this project.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. (2014). Generative Adversarial Networks.

[2] Mehdi Mirza, Simon Osindero. (2014). Conditional Generative Adversarial Nets.

[3] Jinsung Yoon, Daniel Jarrett,Mihaela van der Schaar. (2019). Time-series Generative Adversarial Networks.

[4] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, Ilya Sutskever. (2021). Zero-Shot Text-to-Image Generation.