Capstone Project #2: Project Report

Earnest Long, Jr.

Project Title: U.S. Trademark Smart Search App

INTRODUCTION

The concept of intellectual property (IP) is crucial to the goal of spurring innovation by granting special rights to inventors and creators. In the United States, the United States Patent and Trademark Office (USPTO) is the legal organization that issues and maintains IP in the form of patents and trademarks. A trademark is a design / visual mark (e.g., a logo) or phrase (e.g., a slogan) that serves to identify a company or organization. This helps companies distinguish themselves (and their products) from competitors. Trademarks must be unique and clearly visually distinguishable from those of other companies in that market space. Therefore, in order to avoid the expense of submitting multiple trademark applications, it is advisable to conduct a search to ensure that the mark attempting to be registered is not overly similar to other already registered trademarks.

For this project, I will create an app that helps people determine the likelihood of success of their trademark applications for design marks based on similarity to currently registered marks. The user will upload an image of their prospective mark and some brief text describing it, and my software will return several registered design marks that are ranked according to *content* and *style*. To determine content & style similarity, I will borrow concepts from Neural Style Transfer (NST) *without* generating an image. In NST, you compute cost functions between the newly generated image and a reference image for certain activations and train the network to minimize these costs. Instead of generating a new image, however, I want to compare the activations DIRECTLY for each pair of images (the user's uploaded image and one of the USPTO search results) in order to generate a metric that indicates similarity of content & style. The tool will then rank the results according to similarity, thereby saving the user significant time.

EXPLORATORY ANALYSIS

There is no dataset to clean / wrangle in this project; rather, the app consults the USPTO's database of trademarks through an API. The first order of business, then, was to understand the format and information contained in the API response. The API (documented HERE - "/v1/trademark/documents") specifies a query that takes many

possible parameters and returns a JSON object containing all of the matching results. The JSON object contains a list of documents, each one its own JSON object containing copious amounts of information about each application, such as the application number, information about the mark itself, and information about the authors of the trademark app. Because the only things I seek from this query are the registered design mark images themselves, I needed to find a way to retrieve them. Browsing the USPTO's Trademark Electronic Search System (TESS), revealed that a design mark image file associated with each application was available at a URL containing the serial number of the mark. So, I programmed my tool to comb through the JSON response and retrieve serial numbers for each mark, discarding those applications that lacked a design mark altogether (i.e. word marks).

The next step was to compute what I call a "content score" and "style score" for each pair (the user's image and a design mark image). In NST, the content and style cost functions compare the activations of certain layers of the model (here, as in the NST paper, I'm using VGG-19) in both the reference (content or style) image and the generated image; the neural network then, through gradient descent, learns how to generate a new image with smaller content and style costs at each iteration. Because I am not interested in generating a new image in this project, I simply use the cost functions to generate a cost which is then used to calculate content and style metrics for each pairing. Images that are more similar in content will have smaller costs in the layers associated with content, and similarly for style. When all of the content and style scores are calculated, the results are then ranked and returned (with the most similar image first).

MODEL TUNING

Because the VGG network was already trained and optimized, no hyperparameter tuning was necessary for it. However, I still have two degrees of freedom for calculating costs: the layer whose activations are used to compute the content costs and the layers whose activations are used to compute the style costs. In order to determine the optimal layers for these costs, I downloaded test images (design logos, since this app will compute the similarity of design marks) and ran cost trials, varying the layers used to compute those costs. The results of these experiments are below:


Figure 1: From left to right: “Test Logo 1”, “Test Logo 2”, “Feline Innovations Logo”, “Cat Test Logo”

	conv 1_1	conv 1_2	conv 2_1	conv 2_2	conv 3_1	conv 3_2	conv 3_3	conv 3_4	conv 4_1	conv 4_2	conv 4_3	conv 4_4	conv 5_1	conv 5_2	conv 5_3	conv 5_4
TL1 vs. TL2	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
FI vs. Test Cat	-13.46%	-27.48%	36.6%	23.0%	20.05%	27.08%	28.39%	22.83%	31.12%	24.84%	33.12%	39.98%	20.35%	6.95%	12.57%	90.0%
FI vs. TL1	39.61%	35.68%	22.49%	35.6%	34.02%	43.81%	51.14%	52.71%	59.74%	55.25%	48.89%	37.98%	40.56%	39.25%	27.37%	40.0%

Figure 2: Content cost % comparison to the cost between Test Logo 1 and Test Logo 2

Because TL1 and TL2 are derivatives of each other, they computed very low content costs. Compared to those costs, we would expect FI vs. TestCat to have slightly higher cost (not the exact same image elements but a cat in both for semantic similarity), and FI vs. TL1, being very distinctly different images, to have a much higher cost. Layer ‘conv5_2’ represented the expected cost variation, so it is the best content layer for this application.

I performed a similar experiment for the style cost layers, using images of Impressionist Monet paintings for comparison with the test logos. This revealed an optimal style layer set of ‘conv1_1’, ‘conv2_1’, ‘conv3_1’, ‘conv4_1’, and ‘conv5_1’.

RANKING ALGORITHM

Finally, I needed an algorithm for determining the ranking of the results, as well as a way to present the calculated scores. The raw content and style costs can be large numbers - up to 1x10^7. Furthermore, smaller costs indicate greater similarity, which, if

presented as-is, would go against human intuition. So, in order to generate intuitive content and style *scores* from the costs, I took the reciprocal of the cost (so that a higher score would now indicate *greater* similarity) and multiplied by a factor (100 for content and 1x10^8 for style) to produce scores on the order of 1 to 100. A single metric was needed for determining the rankings though, so I created a composite score that is a linear combination of the content and style scores. The formula is: composite_score = w1 * content_score + w2 * style_score. I want to weight content and style evenly to ensure results that score highly in either category are near the top, so I chose w1 = 1.0 and w2 = 1.0.

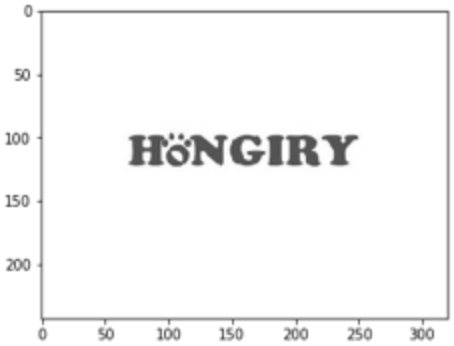
RESULTS

Taking a slightly different version of my company's logo and plugging it into the ranking algorithm produces my company's registered trademark as the top result, as desired.

Figure 3: Input Image (Search query: 'cat three gears')

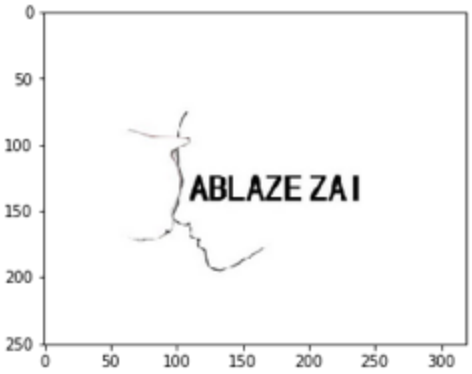
Composite score: 16.99
Content score: 10.4
Style score: 6.59

Figure 4: Algorithm result #1



Composite score: 10.64
Content score: 9.0
Style score: 1.64

Figure 5: Algorithm result #2



Composite score: 9.81
Content score: 7.91
Style score: 1.9

Figure 6: Algorithm result #3

Interestingly, the second result produces an only slightly lower content score as a stylized word mark with no cat image - this is likely due to my test image having more text than our registered mark. The style score is significantly higher than all of the other results though, as expected.

CONCLUSION AND FUTURE WORK

As expected, NST-style content and similarity produces a much more human-intuitive notion of image similarity than traditional metrics. The algorithm works well and returns images that are most similar to the input image. Future work that could be done to improve the app includes:

- More testing for optimum content and style layer selection
- Exploring the effects of color on content/style costs
- Considering alternative means of weighting content and style scores
- Workaround for needing to provide descriptive search terms for the USPTO API (purely visual search!)