CS230

Generating websites from mockups using GANs

Siddhant P.Pardeshi
SCPD
sidppar@stanford.edu

Pranit Kothari
SCPD
pranit@stanford.edu

Abstract

Front-end web development is a creative process, frequently involving transformation of UX mockups into code. Recent studies in Generative Adversarial Networks (GANs) (1) have experimented with generating images of nature and everyday items. This study explores the application of GANs in applying style transfer on websites, particularly in transferring fonts and colors across different style domains. A fine-tuned version of the CycleGAN architecture is used for the model and qualitative and quantitative analyses are presented.

1 Introduction

One of the primary roles of front-end developers involves developing UI (User Interface) code based on Design mockups created by UX (User Experience) engineers. Product teams sometimes like to experiment with multiple variants of a design. However, as this is largely a manual process involving a significant turnaround time, such iterations are costly, leading to a slow churn for the Idea $\rightarrow$ Experiment $\rightarrow$ Implement circle. With recent advances in deep learning, we believe it is possible to automate this process of GUI code generation from mockups, making it easier for product teams to iterate and AB Test different variations for effectiveness. Such variations themselves can also be generated by neural networks, which adds to our interest in this field.

With the above goal in mind, this project presents a model that transfer style (color and font) from website domain to another. This can be useful when adding a new layout to website that already exists on another website, but possibly in a different style. The model helps visualise the layout of domain A in the style of domain B.

The input to our algorithm is an image of a website from domain A. We then use a neural network to output an image that represents the website with the layout of domain A and style of domain B.

2 Related work

Style transfer on paintings and landscapes using neural networks has been the subject of several works in recent times. For instance, Gatys et.al (2) explored style transfer without GANs while Zhu et.al (3) and Isola et.al (5) explored it using GAN-based architectures.

Of these, approaches using GANs have been found to perform better than those without GANs. Using GANs for style transfer thus is now a state-of-the-art approach. The process usually doesn't need any manual hand engineering. Thus far natural landscapes and paintings have been the subject of style transfer, but the concept has not been popularly applied on websites at the time of writing. This study aims to undertake the novel task of applying style transfer on websites, using state-of-the-art GAN techniques.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset and Features

The study involves analysis on two datasets. One dataset represents real-world screenshots of actual websites, specifically Amazon and Flipkart, two major e-commerce platforms in India. This dataset contains 10718 manually captured screenshot images in RGB domain, each of dimensions 512 * 512. Of these, 10000 images were randomly selected for train set and 718 images for dev set.

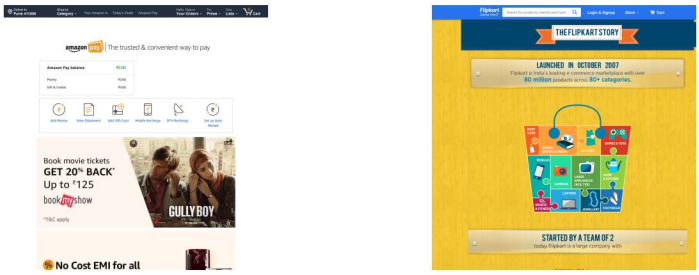

Figure 1: Sample images from the Amazon and Flipkart dataset

The second dataset represents color and font-specific features in websites and consists of manually generated content. Specifically, each image contains a table with placeholder text. For data augmentation, we used a native C-sharp HTML generator to iterate over fonts and colors for the table. During this process, the generator creates screenshots of the rendered HTML, which are used to formulate our dataset. These images were of dimensions 224 * 224, and 10950 in number. 10000 images were randomly selected for the train set and 950 for dev set.

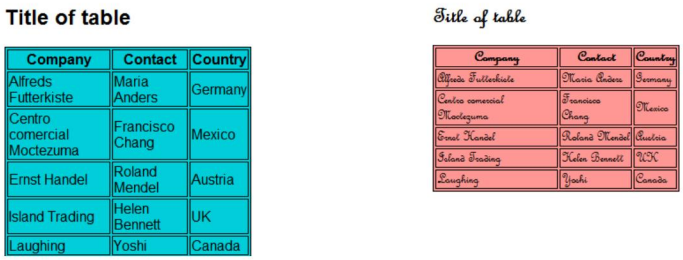

Figure 2: Sample images from the Generated table dataset

4 Methods

We evaluated several state-of-the-art GAN models with an Optimizing metric as per-pixel accuracy > 0.5, which is required for our website usecase and satisficing metric as per-class accuracy > 0.10. This evaluation was carried out in the original CycleGAN (3) paper on the Cityscapes dataset, which is based on recognizing structural features and is in one way, similar to recognizing structural elements in websites. We place the results below for reference.

FCN-scores for different methods, evaluated on Cityscapes labels→photo.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN	0.45	0.11	0.08
BiGAN/ALI	0.41	0.13	0.07
SimGAN	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	**0.58**	**0.22**	**0.16**
pix2pix	0.85	0.40	0.32

Figure 3: CycleGAN model comparison

Our baseline model is based on CycleGAN. It allows for unpaired image-to-image translation from a source domain X to a target domain Y. The model learns a mapping G: X $\rightarrow$ Y such that the distribution of images from G(X) is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained, it is coupled with an inverse mapping F: Y $\rightarrow$ X and a cycle consistency loss is introduced to push F(G(X))$\approx$X (and vice versa).

For the mapping function G: X $\rightarrow$ Y and its discriminator D_Y, the objective is expressed as:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = &\mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]\end{aligned}$$

where where G tries to generate images G(x) that look similar to images from domain Y, while D_Y aims to distinguish between translated samples G(x) and real samples y. G aims to minimize this objective against an adversary D that tries to maximize it, i.e., $min_G max_{D_Y} L_{GAN}$(G, D_Y, X, Y). A similar adversarial loss is used for the mapping function F : Y $\rightarrow$ X and its discriminator D_X as well: i.e., $min_F max_{D_X} L_{GAN}$(F, D_X, Y, X).

To satisfy backward cycle, the cycle consistency loss is given by:

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = &\mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

The full objective is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

where λ controls the relative importance of the two objectives.

The aim is to solve:

$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

The overall model can be visualized as per the image below.

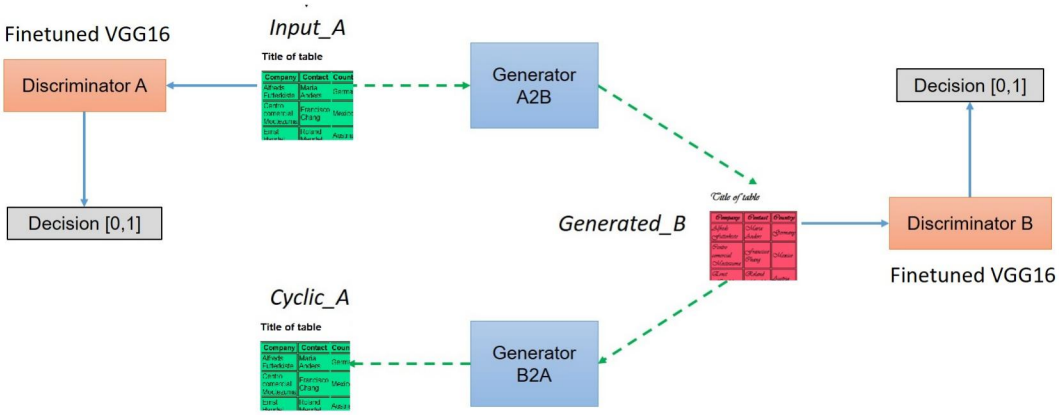

Figure 4: Our CycleGAN Model

5 Experiments/Results/Discussion

While the CycleGAN model's performance on datasets such as horse-to-zebra and artistic paintings is commendable, the qualitative results on our Amazon-to-Flipkart or custom table website dataset was not visually appealing. Hence we set out to improve performance by fine-tuning the model. Our optimizing metric was to minimize loss and satisficing metric was a Generator A model size less than 15 MB, to enable convenient usage with mobile platforms and low-end data-transfer scenarios. We carried out two experiments in this regards. As a first experiment we applied transfer learning to a ResNet50 (6) model pre-trained on ImageNet (7). However, in the last softmax layer we set the output values to 2 instead of a 1000 in the original model. We observed a Generator A loss of 0.851, as opposed to 0.974 for the original model.

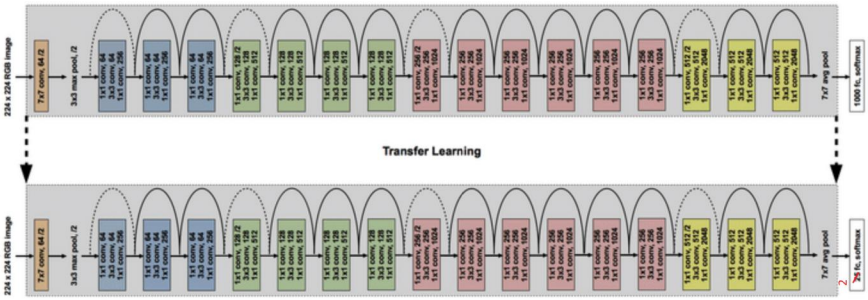

Figure 5: ResNet50 architecture with pretrained model

As a second experiment, we applied a VGG-16 (8) model pretrained on ImageNet for Discriminator A and Discriminator B, by freezing the previous layers and training the last layer. Again, we set the customized output layer to have 2 outputs instead of 1000. In this case we observed a loss of 0.258, which is the better than the original and ResNet50 based models.

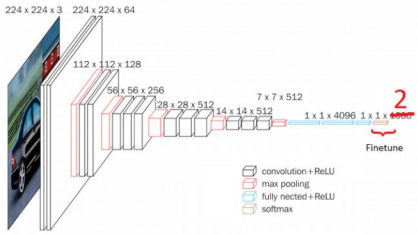

Figure 6: VGG16 finetuned architecture

Performance plot for each model is presented in figure 7.

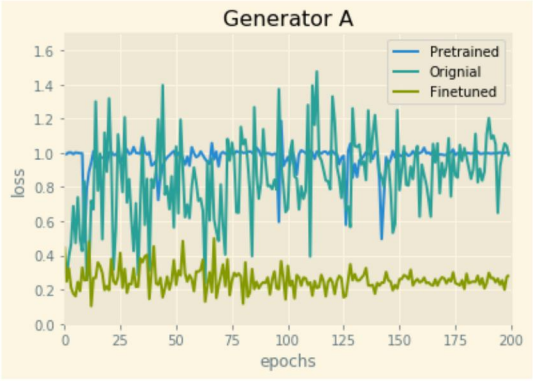

Figure 7: Generator A loss for each model

During the course of these experiments, we tweaked a number of hyper-parameters and present in table 1 the combination that worked best.

Hyperparameter	Choice	Comments
Learning rate (Adam (9))	0.0002	Based on original CycleGAN implementation
Mini-batch size	8	Based on infrastructure limitations
Number of epochs	200	Found to be optimal, for convergence

Table 1: Choice of Hyperparameters

Table 2 presents quantitative analysis of results obtained with the VGG-16 fine-tuned model on our Tables dataset.

Model	Loss	Comments
Baseline	0.974	Original CycleGAN model
Pre-trained	0.852	ResNet50, on ImageNet
Fine-tuned	**0.258**	Pre-trained VGG-16, on ImageNet

Table 2: Model Evalutaion

Qualitative results for our experiment with the finetuned VGG-16 model can be seen in figure 8 and figure 9.

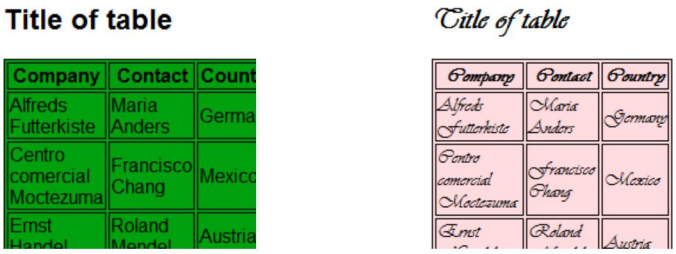

Figure 8: Real A and Generated B from our Tables dataset

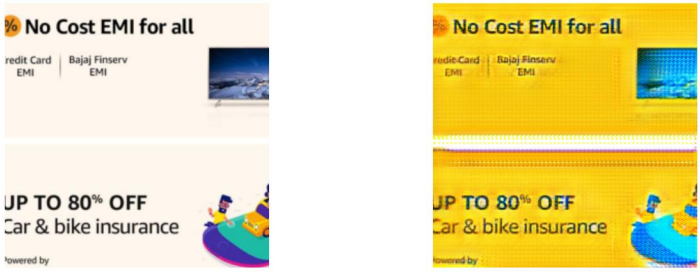

Figure 9: Real A and Generated B from our Websites dataset for Amazon India and Flipkart India

6 Conclusion/Future Work

CycleGAN architecture with our VGG-16 fine-tuned model offers high precision solution for website style transfer. Results were good due to use of transfer learning and fine-tuning. Our code is uploaded to `https://github.com/siddhantpp/UiGAN`

Novelties of the approach

- Lower error compared to baseline model.
- Ability to transfer font and themes across website domains, a hitherto unexplored area.

7 Contributions

Both authors contributed equally to the project in every aspect, starting from conceptualization of the idea to writing the code and interpreting the results, along with writing the report. Working in the same corporate office and team allowed us to sit together and spend time after working hours for all course deliverables.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

[2] Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

[3] Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2223-2232).

[4] CycleGAN code implementation - https://junyanz.github.io/CycleGAN/

[5] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[7] ImageNet website - http://www.image-net.org/

[8] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[9] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[10] Pytorch and torchvision libraries - https://pytorch.org/