Building Deep Learning Architectures to Understand Building Architecture Styles

Caroline Ho
Department of Computer Science
Stanford University
cho19@stanford.edu

Cole Thomson
Department of Mechanical Engineering
Stanford University
colet@stanford.edu

Abstract

Architectural style differentiation is a relatively unexplored subset of computer vision, with applications in geo-localization and urban surveying. Using a dataset of building images belonging to 25 different architectural styles, we built two models: a CNN to classify a building's style from an input image and a conditional GAN to generate images of buildings corresponding to different styles. With the classifier, we can predict a building's architectural style with an accuracy of 79.6%, significantly outperforming the dataset's previous benchmark. While our generator outputs blurry images, one can still see the approximate stylistic forms in the results. We conclude that while architectural styles are fairly simple to recognize from given features of a building, it is far more difficult to take those features and create a realistic building from them.

1 Introduction

In this paper we explore whether a deep neural network is capable of understanding the differences between architectural style. Architectural style recognition has broad applications, ranging from urban surveying with street view images to aiding geo-localization (determining where an image was taken). However, it presents a unique set of challenges that set it apart from typical image-based classification and generation problems. On the one hand, buildings can have widely different forms but share the same style. For example, though the curvy Guggenheim Museum Bilbao looks nothing like the blocky Dallas Wyly Theatre, they are both deconstructivist buildings (*Fig. 1*). On the other hand, some buildings share similar forms but are of different styles. For example, the Ickworth House rotunda and the University of Virginia rotunda look fairly similar with regard to their shapes but are respectively Georgian and Palladian (*Fig. 2*).

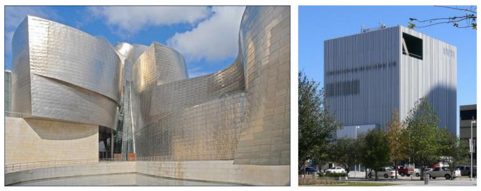
Fig. 1

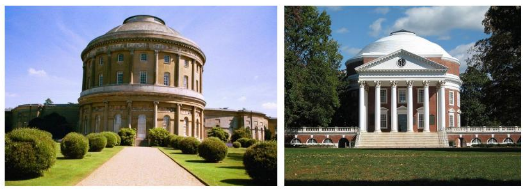
Fig. 2

We explore the challenge of architectural style recognition using two different methods. The first of these is to use a Convolutional Neural Network (CNN) as an architectural style classifier. We input an unlabeled image of a building, and the model outputs its architectural style. Our second model is a Conditional Generative Adversarial Network (cGAN) that generates images of buildings conditioned by architectural style. This network takes an architectural style label as its input and (after being trained on images of buildings labeled with their architectural styles) outputs a randomly generated image of a building in that style.

2 Related Work

Very few papers have explored applying machine learning to the problem of classifying architectural styles. Shalunts et al. provided an early contribution in 2011 by using clustering to classify building facade windows into 3 style categories [1]. More recently, Xu et al. used multinomial latent logistic regression to classify buildings of 25 architectural styles with an accuracy of 0.4621 [2], and Pesto used ResNet to classify U.S. house images of 5 architectural styles with an accuracy of 0.798 [3]. We build on their work by applying CNNs to Xu et al.'s broader set of architectural styles.

Regarding our architecture image generation task, no papers exist that use GANs to generate any sort building. Previous applications of conditional GANs, developed by Mirza and Osindero [4] (and to be further discussed in part four), include generating different classes of written digits (using the MNIST dataset) and faces (using the CelebA dataset). Radford et al. improved GANs with the Deep

Convolutional GAN (DCGAN) that performs better on unsupervised learning tasks (further discussed in part four) [5]. We combine cGANs and DCGANs to create a cDCGAN for the purpose of architectural style-conditioned building image generation.

3 Dataset and Features

3.1 Description

We use Xu et al.'s dataset [2], which contains 4,794 RGB images of buildings from 25 architecture style classes ranging from Achaemenid to Tudor Revival. The number of images per class is not uniform, ranging from about 70 to 200. Additionally, the images have differing dimensions, resolutions, and aspect ratios (as seen in *Figs. 1* and *2*, whose images were originally different heights but were resized).

For our classification experiments, we split the dataset into an 80/10/10 train/dev/test split.

3.2 Preprocessing

We resize our images to have a uniform minimum dimension and then center crop them (224 px for DenseNet and ResNet classifiers, 299 px for Inception classifiers, and 64 px for our cDCGAN), as seen below.

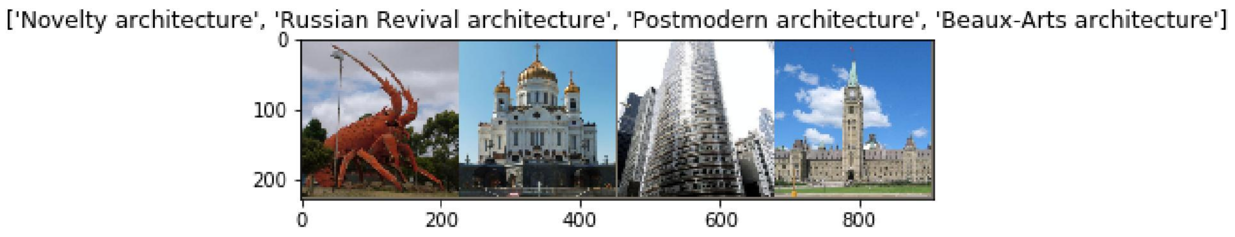
['Novelty architecture', 'Russian Revival architecture', 'Postmodern architecture', 'Beaux-Arts architecture']
0
100
200
0
200
400
600
800

Fig. 3: Cropped and Resized Sample Images from Dataset

We further normalize our data. For classification, the pre-trained models we use with PyTorch require that we normalize with a mean of [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225] [6]. For image generation, we normalize our images to the range [-1, 1] for more efficient training.

4 Methods

4.1 Classification

We perform transfer learning on three CNN different architectures that were pre-trained on ImageNet: ResNet152, DenseNet201, and Inception v3. In order to tailor the networks to architectural style classification, we replace the last layer of the network with a softmax for the 25 architectural style classes. We compare results from fine-tuning (updating the weights of the network) and fixed feature extraction (freezing the weights of all but the last layer of the network). Our model uses Adam optimization with the default PyTorch learning rate of $\alpha = 0.001$ as well as learning rate decay with a decay factor of $\gamma = 0.1$. The model optimizes with a cross-entropy loss function that compares the output class probabilities $\hat{y}$ with the ground truth y across m classes (*Fig 3.*)

$$J(\hat{y}, y) = \frac{-1}{m} \sum_{i=1}^{m} (y^{(i)} log(\hat{y}^{(i)}) + (1 - y^{(i)}) log(1 - \hat{y}^{(i)}))$$

Fig. 4: Multiclass Cost Function for Cross-Entropy Loss

4.2 Generation

We use a conditional GAN (cGAN) to generate style-conditioned images of buildings. The model's generator takes an input of random noise and class labels and generates labeled images to output. The discriminator trains on real images of buildings and their respective labels, learning to recognize both if an image of a building is real and if its style matches the image's class label.

$$min_G \; max_D \; V(D, G) = E_{x \to p_{data}(x)}[log D(x|y)] + E_{z \to p_z(z)}[log(1 - D(G(z|y)))]$$

Fig. 5: Conditional GAN Value Function [4] (X = Real Building Images, Y = Style Labels, Z = Random Noise)

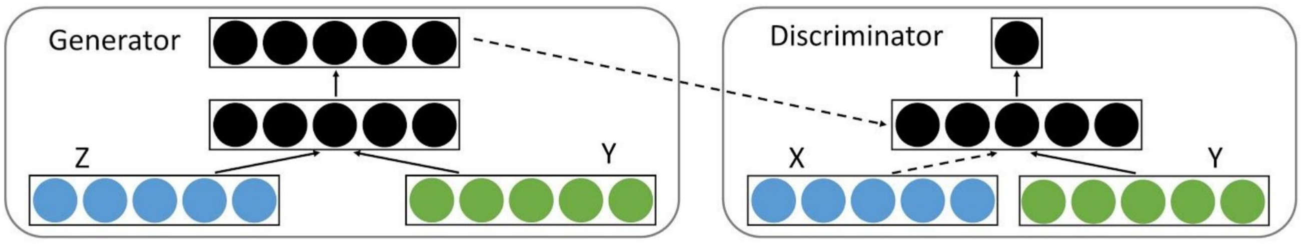

Fig. 6: Conditional GAN Architecture Overview [4]

We utilize a deep convolutional architecture (DCGAN) for both the discriminator and the generator, as this performs better in unsupervised learning tasks. The generator takes in a noise vector of dimension 625, then projects it into a convolutional layer with 1024 channels. Four 4x4 strided convolutions reduce the number of channels to 512, 256, 128, and finally 3, outputting a 64x64 RGB image. Each of these layers uses ReLu and batch normalization except for the last layer, which uses Tanh and no batch normalization.

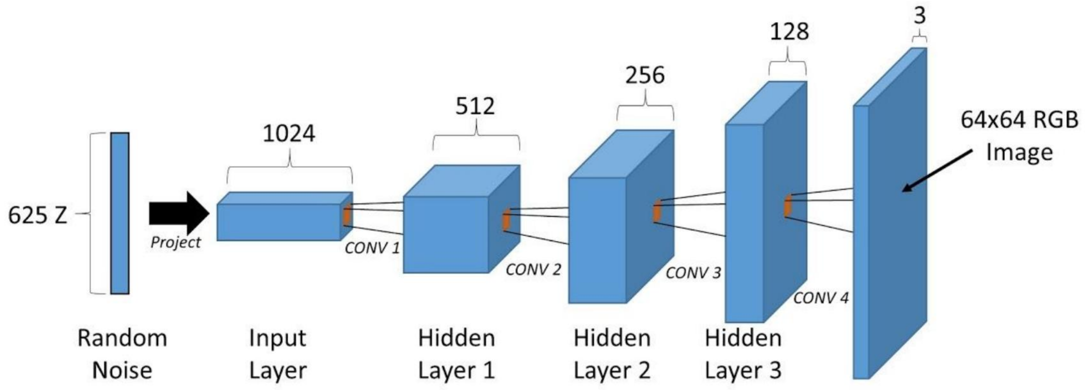

Fig. 7: DCGAN Generator Architecture

The discriminator has similar architecture to the generator, but functions in reverse. It takes an input image and convolves it over four strided 4x4 layers, ending in a single node that feeds into a Sigmoid function for classification. The other three layers have a Leaky ReLu activation function, and all layers but the first and last use batch normalization.

We use a learning rate of 0.0002 for both the generator and 0.0001 discriminator, as Radford et al. suggests a lower learning rate than the standard 0.001 for DCGANs [5]. We also use Adam Optimization for both components.

5 Experiments/Results/Discussion

5.1 Classification

Our transfer learning models all perform quite well. Most notably, our pre-trained DenseNet using fixed feature extraction achieved an accuracy of 0.795833 and an F1 score (averaged over all classes) of 0.789431. While this seems to outperform Xu et al.'s previous benchmark of 0.4621 accuracy (with a whopping increase of 0.33), we are unable to make a direct comparison since we do not have access to Xu et al.'s original code or data split. We hypothesize that DenseNet with fixed feature extraction performed so well since many of the layers pre-trained on ImageNet pinpointed useful features for the architectural classification task as well.

Model	Style	Accuracy	Macro F1 Score
DenseNet	**Fixed Feature Extraction**	**0.795833**	**0.789431**
DenseNet	Fine-tuning	0.675000	0.601738
ResNet	Fixed Feature Extraction	0.747917	0.724015
ResNet	Fine-tuning	0.500000	0.410742
Inception	Fixed Feature Extraction	0.654167	0.623107
Inception	Fine-tuning	0.722917	0.651555

Table 1: Classification Accuracy and F1 Scores

When examining the class-by-class F1 scores for our top-performing model, we see that our classifier performs well nearly across the board, scoring above 0.9 for 6 classes and well above 0.5 for all but Tudor Revival. We note that the classes on which our model performs best are mostly older styles (e.g., Ancient Egyptian) and/or are generally quite different from other classes in our dataset in form (e.g., Deconstructivist, with fewer straight edges). The classes on which it performs less well are generally closer in form or time period with other forms and thus are more likely to be confused (e.g., Palladian and Georgian – the former is sometimes considered to be a sub-style of the latter).

Achaemenid	American craftsman	American Foursquare	**Ancient Egyptian**	Art Deco
0.92307692	0.71111111	0.8	**1.0**	0.72527473
Art Nouveau	**Baroque**	Bauhaus	Beaux-Arts	**Byzantine**
0.77647059	**0.91304348**	0.66666667	0.71428571	**0.94117647**
Chicago school	Colonial	**Deconstructivism**	Edwardian	Georgian
0.77777778	0.78571429	**0.9047619**	0.72727273	0.69565217
Gothic	Greek Revival	International	Novelty	Palladian
0.8	0.85294118	0.79166667	0.77777778	0.75
Postmodern	Queen Anne	Romanesque	**Russian Revival**	Tudor Revival
0.68965517	0.85057471	0.76190476	**0.94736842**	0.4516129

Table 2: F1 Scores – DenseNet with Fixed Feature Extraction, Scores > 0.9 Bolded

When examining our confusion matrix (*Fig. 8*), we indeed see that most of the top-performing model's errors come from confusing relatively similar architectural styles (e.g., Postmodern and International Style, American Foursquare and American craftsman). However, it occasionally confuses very different styles (i.e., Achaemenid and Beaux-Arts).

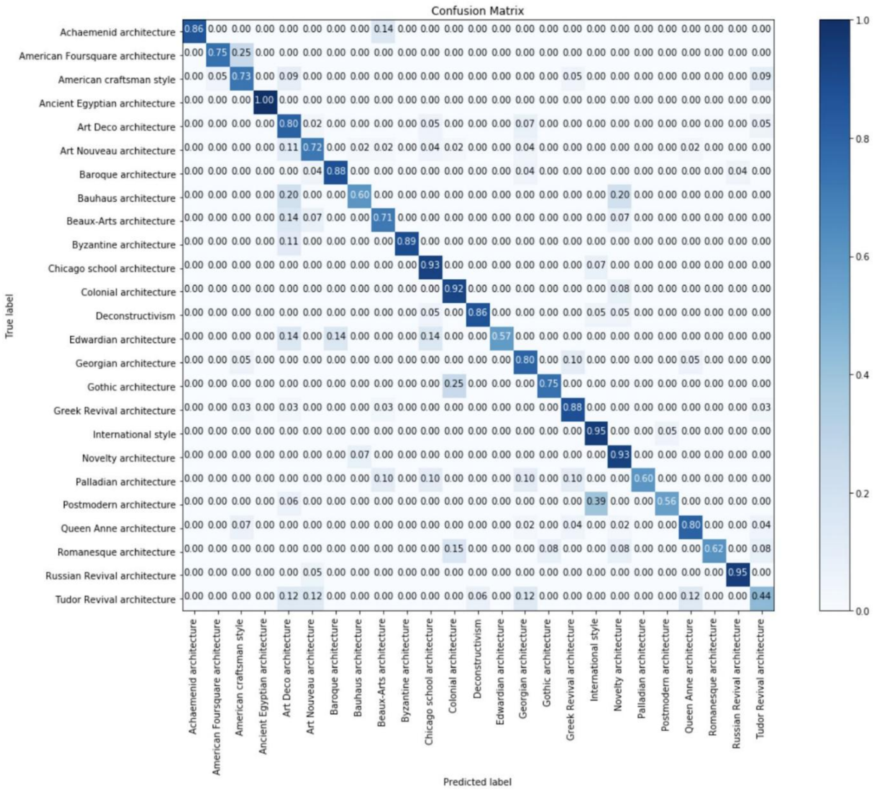

Fig. 8: Confusion Matrix – DenseNet with Fixed Feature Extraction

5.2 Generation

While the images produced by our generator are fairly abstracted and blurry, when taking a closer look, one can find features of the given style in certain images. Some of the more comprehensible style images include Ancient Egyptian (pyramid), Baroque (cupola), and Tudor Revival (steeply pitched brown roof). Even the images we initially assumed were completely incomprehensible contain elements of their style – for example, when examining the American craftsman image, we realized that the dark blob represented the edges of a brown gabled roof, highlighting the white walls of the house (*Fig. 11*). Similarly, we see hints of a light blue onion dome in the Russian Revival image (and perhaps even some smaller domes around it), the form of a skyscraper in the Chicago school image, and oriel windows in the Queen Anne image. We hypothesize that many styles are so diverse, as discussed in the introduction, that they do not lend themselves well to creating a "typical" building in that style (especially ones such as Novelty). Another reason many images seem fairly abstract is that the input image size (64 px) was too small to capture intricate details, especially for more ornamental architectural styles such as Beaux-Arts and Gothic.

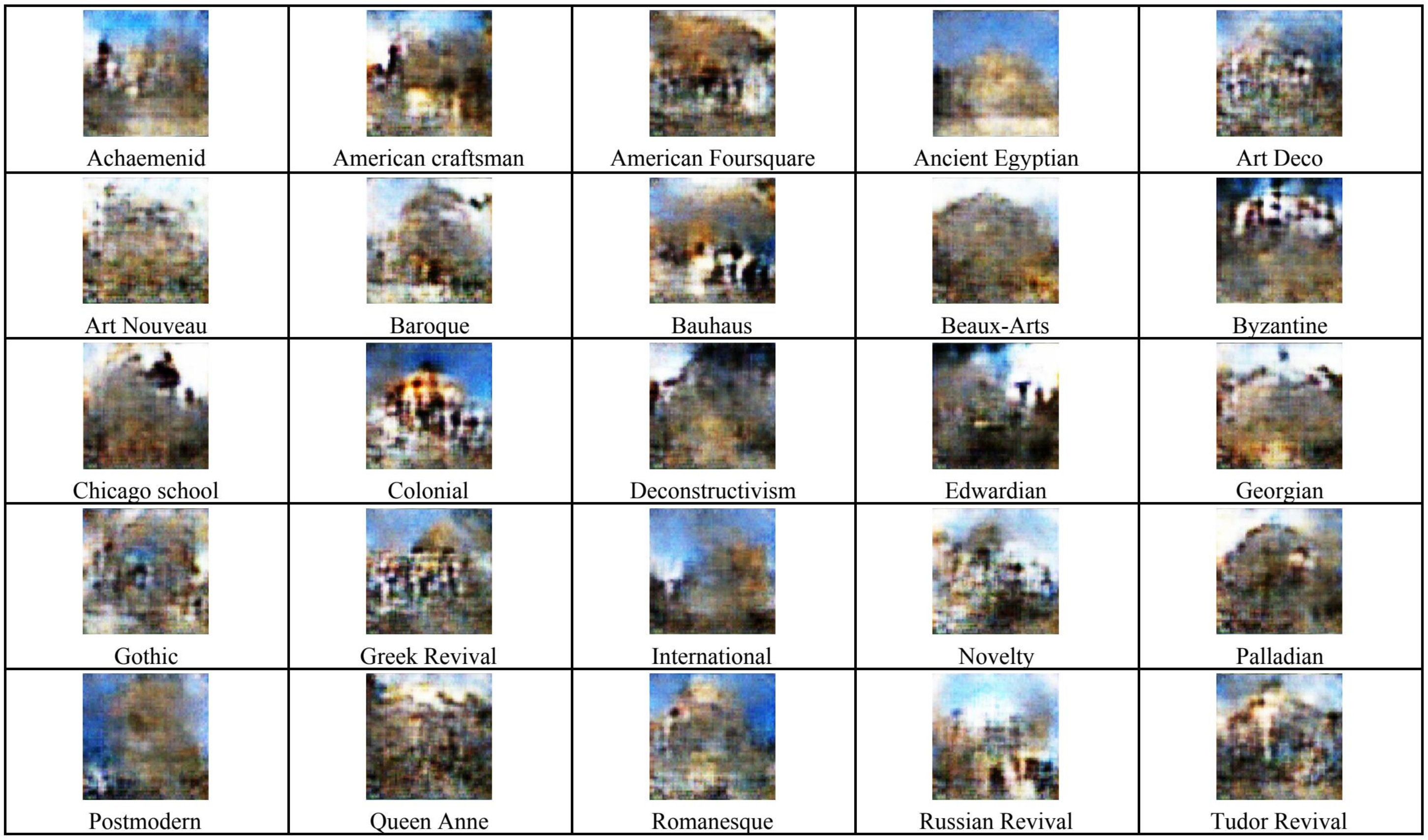

Fig. 9: cDCGAN-generated images of architectural styles

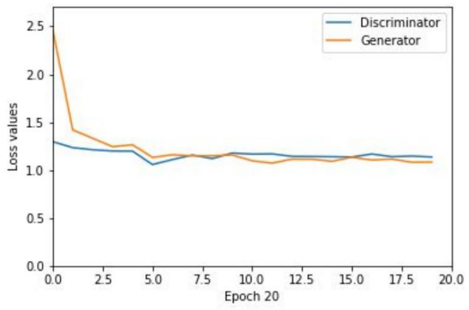

Fig. 10: cDCGAN Loss

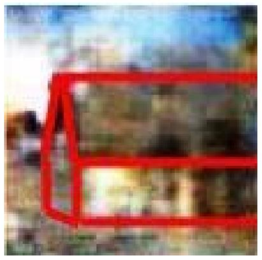

Fig. 11: American craftsman (red outlining house)

6 Conclusion/Future Work

On the one hand, our classifier performed very strongly, with our best model achieving an F1-score of 0.789431 and making errors mostly related to similar style classes. On the other hand, while our generator produced images that contained visible features of a given style, its results were still rather abstract. From this, our key takeaway is that architectural style is easier to recognize from features than it is to create from them.

With more time, we would perform interpretability analyses to gauge the effectiveness of our model at classifying based on style rather than contextual background features (e.g. the desert background of most ancient Egyptian architecture). The first of these would be occlusion sensitivity, which occludes different parts of an image with a gray square and then calculates the confidence on the true class given the occlusion. We would also perform gradient ascent to visualize what the model is judging to be the quintessential features of each architectural style.

As for our generator, with more computational resources, we would try inputting larger images into our model to determine whether results become less abstract and show more intricate details. Additionally, we would try re-implementing the model using different GAN architectures, such as the Wasserstein GAN (which improves the stability of learning) [7], to see if we would get better generated images.

7 Contributions

Ho implemented the classifier and generator in PyTorch, which involved creating a script to split the data into train/dev/test sets, rewriting prior implementations of image classifiers [8][9] and cDCGAN [10] to handle the architectural style dataset, and writing documentation. Thomson researched and implemented experimental architectures for the generator, created graphics, and built the corresponding poster. Both members contributed equally to the writing of the report.

Our code is on GitHub at `https://github.com/carolineh101/deep-learning-architecture`.

References

[1] G. Shalunts, Y. Haxhimusa, and R. Sablatnig. Architectural Style Classification of Building Facade Windows. In *Advances in Visual Computing 2011*, pages 280-289. Springer, 2011.

[2] Z. Xu, D. Tao, Y. Zhang, J. Wu, and A. C. Tsoi. Architectural Style Classification using Multinomial Latent Logistic Regression. In *ECCV 2014*, pages 600–615. Springer, 2014.

[3] C. Pesto. Classifying U.S. Houses by Architectural Style Using Convolutional Neural Networks. 2017.

[4] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*, 2014.

[5] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[6] Torch Contributors. torchvision.models. https://pytorch.org/docs/stable/torchvision/models.html, 2018.

[7] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[8] S. Chilamkurthy. Transfer Learning Tutorial. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html, 2017.

[9] N. Inkawhich. Finetuning Torchvision Models. https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html, 2017.

[10] M. Kim. cDCGAN. https://github.com/togheppi/cDCGAN, 2017.