# Object Detection in Artwork using Neural Style Removal

Nick Hale David Li Aurik Sarker

**Abstract**

This paper examines the performance of "traditional" deep learning object detection architectures on the *cross-depiction problem*, the problem of recognizing objects in images independent of what style (drawn, photographed, painted) they are shown in. Furthermore, we investigate the suitability of neural style transfer for performing "style reduction" for improving performance of CNNs on the cross-depiction problem. For this project, we test various methods of style reduction on a high-performance object detection model in You Only Look Once (YOLO).

## 1 Introduction

Detecting objects in artwork, especially abstract art, is a fairly challenging task, even for humans. Most of the time, images are easily discernible to the human eye. However, with highly abstract art, sometimes objects are up to interpretation. We are interested in classifying objects in artwork in contrast to "natural" images (photographs). This is known as the cross-depiction problem, the problem of recognizing objects in an image independent of what style they are depicted in. Since there is not enough readily available data to train our own classifier, we are interested in improving current state-of-the-art object detection methods by what we call "neural style removal".

A fundamental problem in computer vision is object detection, the task of assigning a bounding box to an object and classifying the object within the bounding box. Since there may be multiple objects of various different classes within an image and objects within the image may be differently sized, object recognition is a difficult problem at its core, even ignoring potentials issues such as occlusion.

Another recent work in computer vision is the concept of neural style transfer, in which the style of a piece of artwork is transferred onto the content of a picture. The result is typically a picture with elements of abstract brushwork or shapes that maintains the recognizability of the content of the original image. This project is focused on neural style *removal*, the inverse task of neural style transfer. The ultimate goal is to reduce the style of a piece of artwork so objects can be more easily detected. We tried three different methods to achieve this. The techniques used were: (i) applying neural style transfer in reverse, (ii) applying style transfer in reverse in series, (iii) and finally by analyzing the algorithm and adjusting it to fit our needs of style removal rather than style transfer.

In this paper, we will examine current work and progress on the cross-depiction problem and on neural style transfer in Section 2. Then, we will describe the object detection methods we used and the techniques of neural style removal that we explored in Section 3. Results of our experiments including sample images and test performance will be displayed in Section 4. Analysis of results is done in Section 5, and concluding remarks and areas for improvement and future work are in Section 6.

## 2 Related Work

### 2.1 The Cross-Depiction Problem

*Cross depiction* is the ability to recognize objects in a medium regardless of its photographic or artistic style. The human ability to do so is quite robust; from a very young age, children already distinguish animals and items from many different types of images. Machines, however, rely on identifying and comparing visual attributes, typically over-fitting to one particular depiction. Solving this problem will require development of a model which encodes spatial relations between object features, just as the human brain does subconsciously.

Object detection itself is not a new topic; over the past few decades countless methods have been developed to tackle this problem. In fact, recent advancements in deep learning techniques have led to significantly improved performance and detection speed in object detection. However, while research devoted to this subject is very extensive, the literature addressing automated cross depiction is actually quite sparse, even though a solution to this problem would have great application in computer graphics and web image searching, among many other areas. In their paper, Hall et al. [5] detail some methods of automated cross depiction:

- *Domain Adaptation* This method projects the two (source and target) images onto a lower dimensional subspace, and uses this projection to categorize images accordingly. This projection may be implemented in different ways, two of which are listed here. Geodesic Flow Kernel (GFK) performs dimensional reduction (using PCA, LDA, for example) on the images, then calculates a geodesic distance between the image subspaces. Subspace Alignment (SA) simply learn a transformation between the subspaces.

- *Part Based Models* This method determines spatial relationships between large-scale features in objects. Deformable Parts Model (DPM) constructs a "constellation map" - some sparse representation - from these features. Multi-labeled Graphs (MG), developed by Wu et al. [9] extends this idea by creating a fully-connected graph from the identified features, calculating weights between them, and training an SVM from these weights.

- *Deep Learning* Convolutional Neural Networks (CNNs) have yielded excellent results in object recognition; in particular, R-CNNs (regions with CNN features) have been adapted to extract feature vectors for each proposed bounded block in the image. These encoded features may then be used to learn an SVM for classification.

In a succeeding paper, Cai et al. [1] compare object recognition on artwork using both naïve and adaptive models. To do the former, they create a Bag of Words (BoW) model using select features from photographs, then train an SVM on the constructed vocabulary. Of the feature detection methods used by the authors, Geometric Blur (GB) seemed to produce the best results. Training on natural images provided by the PhotoArt50 dataset, testing accuracies were expectedly not very good, with the best model (GB-trained BoW) achieving a 0.50 recognition rate. To improve these results, cross depiction methods were tested. While the domain adapted models performed even worse than naïve methods, spatial models such as MG had very good classification rates. An overall comparison between the best naïve model and various cross-depicted models can be seen below in Figure 1. Keep in mind we care only about the final row, where training is done on the photos and testing on the artwork.

| model | | BoW | DPM | MG | CNN |
|---|---|---|---|---|---|
| train | test | GB | HOG | 2×HOG | learned |
| P | P | 77 | 88 | 85 | 97 |
| M | P | 72 | 85 | 90 | 96 |
| A | P | 60 | 78 | 83 | 91 |
| A | A | 72 | 83 | 89 | 89 |
| M | A | 67 | 80 | 89 | 87 |
| P | A | 50 | 68 | 83 | 73 |

**Figure 1: Classification on PhotoArt50.** This table is adapted from Table 1 in Cai et al. [1]. Each row is a unique train / test combination of Art, Photo, or Mixed. Each column is a separate algorithm and feature detection pair.

## 2.2 Neural Style Transfer

Neural Style Transfer is a recent topic pioneered by Leon Gatys [3]. The goal of this work is to generate artwork using two source images by finding a representation of an image's style and using the style to augment the content of another image. The project takes advantage of the architecture of neural networks in an interesting manner in order to achieve this goal.

The original method from Gatys used a neural network, but rather than training a neural network on data to generate a model, this method utilizes neural networks simply to optimize two loss functions with backpropagation. That said, there has been work done on training a network on a specific style so that it can perform style transfer on arbitrary content images, as well as arbitrary style transfer [4] which involves training a network which can take, in addition to any content image, any style image and perform style transfer.

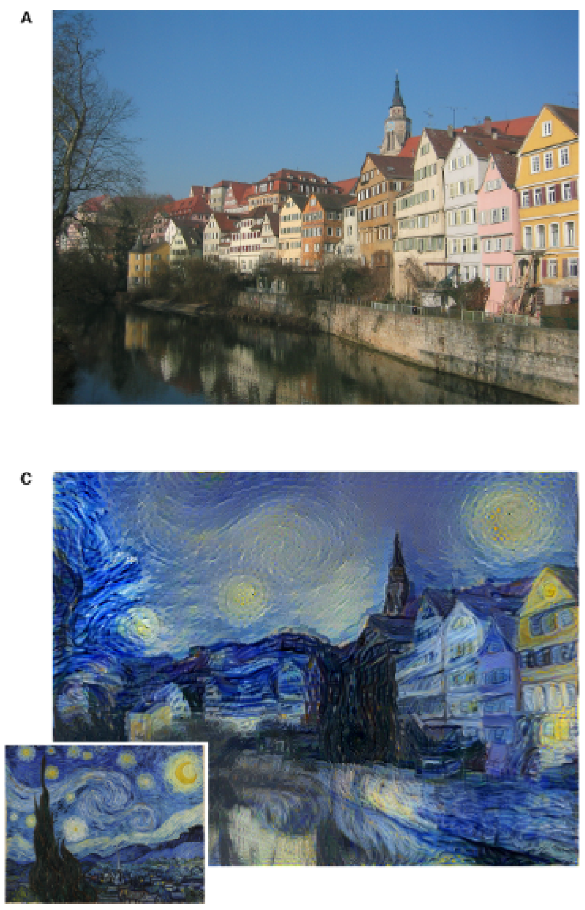


**Figure 2:** "Traditional" Neural Style Transfer. Figure from Gatys et. al [3]

How exactly does neural style transfer work? First, the input to the model is two images, a style image and a content image. Then, we define a loss function associated to each of these, we call them style loss and content loss. The first step in this optimization is to initialize an output image– in practice this is either random noise or simply the content image. Then, the three images are passed through a pre-trained convolutional neural network (in our case VGG16); at several specific layers, the network computes the Euclidean distance between the images for the loss function. These intermediary values are then summed up. This leaves us with the following function describing content loss:

$$\mathcal{L}_{content} = \sum_{\ell} \sum_{i,j} \left(\alpha C^{\ell}_{i,j} - \alpha P^{\ell}_{i,j}\right)^2 \tag{1}$$

To explain where this came from, the summation over $\ell$ is simply summing over all of the specified intermediary layers. The coefficient $\alpha$ is called the content weight, which affects the influence of the content image on the loss function and thus the output image. While the function for style loss is almost identical, there is one vital difference: rather than using raw outputs in the loss function, Gram matrices are used instead. A Gram matrix, attained by multiplying a matrix with the transpose of itself, is used in the style loss function as described by the following equation:

$$G^{\ell}_{i,j} = \sum_{k} F^{\ell}_{i,k} F^{\ell}_{j,k} \tag{2}$$

Why are Gram matrices used instead of raw outputs? We use them to remove the influence of the content within the style image when applying its style. By using the Gram matrix, any content within the image is distributed throughout the image, leaving only stylistic features, such as texture, shapes, and colors. Hence, we are left with the following equation for style loss:

$$\mathcal{L}_{style} = \sum_{\ell} \sum_{i,j} \left(\beta G^{s,\ell}_{i,j} - \beta G^{p,\ell}_{i,j}\right)^2 \tag{3}$$

To get our total loss function, we take the sum of the content and style loss functions.

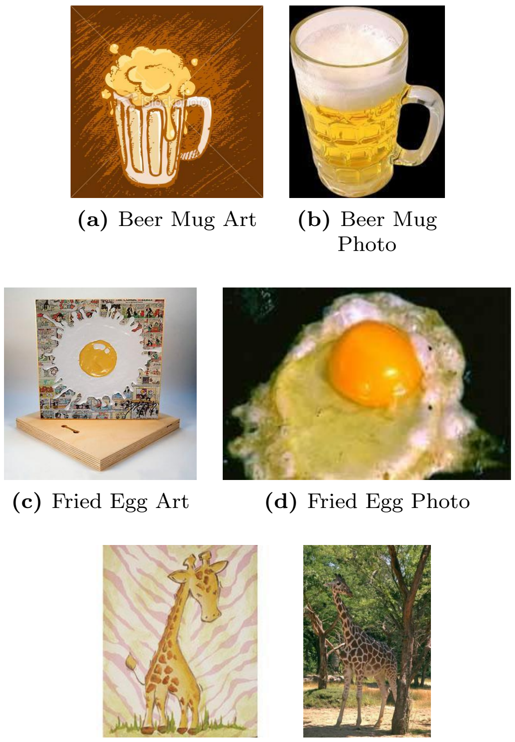
$$\mathcal{L}_{total} = \mathcal{L}_{content} + \mathcal{L}_{style} \tag{4}$$

With a loss function defined, we proceed as usual: the output of the network is backpropagated through the net in order to adjust the weights to reduce the loss function. After enough iterations, we get an output image that contains the content of the content image and the style of the style image.

The issue with this implementation is that it takes quite a long time to go through the optimization loop; using a neural net that has already learned an image's style would be much faster. However, that comes with its own difficulties. We'll discuss the choices of neural style transfer techniques for style removal in Section 3.2.

## 3 Methods

### 3.1 Datasets


**(a)** Beer Mug Art **(b)** Beer Mug Photo

**(c)** Fried Egg Art **(d)** Fried Egg Photo

**(e)** Giraffe Art **(f)** Giraffe Photo

**Figure 3:** Examples of Beer Mug, Fried Egg, and Giraffe in both artwork and photos from PhotoArt50

The PhotoArt50 dataset contains approximately 5,700 images, both artwork and photos, that are categorized into fifty different object classes. Images are annotated with coordinates of bounding boxes around the object (s). The dataset includes a wide variety of object classes, including inaminate objects such as beer mugs, lightbulbs, and pyramids, animals such as crabs, giraffes, and penguins, and food such as fried eggs, hamburgers, and ice cream.

The PeopleArt dataset is made up of approximately 6,500 images categorized into 43 distinct artistic styles, one of which is photography. Each image is also annotated with the coordinates of bounding boxes around people (if present). A wide variety of art styles are covered, ranging from more abstract styles like cubism and cubo-futurism to more realistic style such as Rococo and photorealism.

### 3.2 Neural Style Removal

Typically, neural style transfer [3] is used to increase the presence of a certain style in the input image. The result is an output image now completely stylized in appearance, yet still retaining crucial information from the input, such as edges and object structures. However, we hope to show that a reversal of this process  transferring the style of a natural, realistic image onto an abstract-looking artwork image, effectively *removing* its previous abstract style  may allow for better object recognition of objects in the artwork.

We tried three techniques for style removal – naïvely transferring style of a photograph onto artwork, naïve style transfer with multiple sources, and transfer after segmentation. While there are methods for training a network to do style transfer given a style image, it was impractical for our purposes since our style images are the ones changing, and the content images are the ones staying constant. There was also the consideration for a general style transfer system, as demonstrated by Google Brain [4]. However, code for arbitrary style transfer was not available at the time we started this project. Thus we decided to go with the "basic" version pioneered by Gatys.

#### 3.2.1 Naïve Style Removal

The first thing we tried was naïve style removal, which was just reversing the content image and style images relative to "typical" neural style transfer. As seen earlier, the photo is usually the content image.

Naïve style removal for 400 iterations took, on average, about 5 minutes to run. While running fewer iterations would be faster, it produced considerably worse results – not as much style was transferred. As expected, performing object detection on these semi-stylized images gave worse results than their fully destylized counterparts, on which style removal ran for all 400 iterations. An example of successful style removal is shown in Figure 7.

#### 3.2.2 Multi-source Serial Style Removal

In an attempt to improve results, we performed style transfer of multiple source images onto a singular content image. This operation is executed in series; once the content image is processed through the style transfer with just one style image, the output image becomes the content image for the next style transfer processed again, this time with a different style image. This is repeatable for any number of style images. The goal here is to get output image which would have a more general style, and reflect what real images of the object look like. This method's computation time has the same runtime as the naive style removal, since we're doing the same thing, just with potentially different style images at intermediate points.

## 3.3 Image Segmentation

Our problem of style removal can be thought of as an application of information preservation; we would like to remove as much of the image detail as possible while still retaining the most important part: the content. Image segmentation is just that; it attempts to remove the background from an image, keeping only the important part: the foreground. If we can discard as much useless information through image segmentation beforehand, perhaps style removal will yield better results and lead to more accurate object detection.

### 3.3.1 Edge Detection

We began by preserving the edges, as they typically correspond to regions of high information content. First we tried Sobel edge detection, which requires some thresholding of the magnitude image. Unfortunately, batch processing using Sobel is impractical, as every image would need a different threshold to be segmented properly. Instead, we turned to Canny edge detection [2]. Because it suppresses false maxima in the gradient, it can localize edges much better than other edge detectors, and is more robust as a result. Figure 4 shows both our edge detected image and attempted style transfer on it.

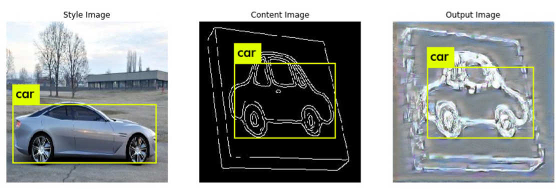


**Figure 4: Style Removal with Canny Edge Detection**
Attempt at transferring style from the real car image onto the Canny edge-detected version of car artwork

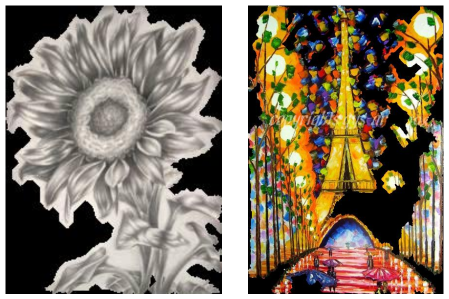
As illustrated by the figure above, the style transfer here is quite bad. Edges, while important, simply do not carry enough information about the original object for content to be preserved in the output, and for the detector to make an informed classification from it. We need to prioritize retaining as much of the foreground as possible to prevent any loss of information.

### 3.3.2 Foreground Segmentation

Our goal now is to eliminate as much background as possible without losing content in the foreground. We start from the edge-detected image, since its features correspond to ares of high information density in the image. We would like to cover the foreground using a flood-fill operation on holes within the edges of the foreground. For objects with weaker features, we perform a combination of morphological thickening and bridging to close gaps in the edges before filling, and repeat this process until the images reaches a sufficient area.

### 3.3.3 Active Contouring

To avoid losing parts of the foreground during segmentation, we tend towards oversegmentation of the image. In doing so, however, we may keep too much of the background and lose fine details at its edges. To resolve this issue, we tested performing active contouring on the original image, using the segmented foreground as a mask. Finally, we performed pixel-wise multiplication of this mask with the original image to effectively set all its (identified) background pixels to zero. Two examples of artwork images segmented this way are shown below, in Figure 5.



**(a)** Sunflower **(b)** Eiffel Tower

**Figure 5: Segmented Artwork Images (PhotoArt50)**
The difference in object characterization between artworks of different depictions is quite apparent here. While the flower is much easier to identify now, the algorithm is unable to detect the Eiffel Tower here due to its intense style.

As Figure 5 exemplifies, this process can work quite well. However, in many cases, quirks in the artwork render the algorithm ineffective at producing an accurate segmentation. This test revealed a strong correlation between segmentation quality and object detection rate; an image which is difficult for the machine to distinguish will also be hard to segment. As a result, segmentation may not actually improve our results where it matters.

## 3.4 Object Detection Methods

Once style was sufficiently removed from the content image, these new images were fed through some object detector, with the hope that our processing would increase its recognition accuracy. We elected to use YOLO as our network model, since it outperforms R-CNN when classifying objects in artwork [7]. Choosing just one model for testing allowed us to focus our analysis on the various neural style removal techniques we implemented in order to determine which results in the most accurate object classifications.

### 3.4.1 You Only Look Once (YOLO)

Typical neural network approaches to object detection, like R-CNN, first determine macro regions in an image, then identify features from there. This pipeline tends to be quite slow, as every component must be trained separately. YOLO drastically improves classification speed by performing unified detection on the full image. This holistic approaches also allows it to make informed classifications, using information from the entire image to make its decision.

Unified detection by YOLO is two-fold. First, the network divides the image into a grid cells, each assigned a probability for its identified class. At the same time, it predicts possible bounding boxes for each cell. In this way, it may determine which bounding boxes correspond to actual objects, *and* know what object it is, as it has already been classified! The actual network is characterized by 24 convolutional layers and 2 fully connected layers [7] pretrained by ImageNet.

Because YOLO both imposes a constraint on the number of bounding boxes and downsamples many times during testing, the network struggles when dealing with fine features bunched in a small area. As a result, YOLO does not perform nearly as well as other systems for general object detection. However, it does tend to outperform others in classification of artwork; its average precision in identifying objects in the Picasso dataset was 0.533, compared to 0.104 for R-CNN. Figure 6 exemplifies YOLO's superiority in classifying artwork. YOLO's holistic image representation technique tends to work much better with abstract images, where objects will rarely match individual pixels but will be relatively similar in shape and size to their real-life counterparts in photographs.

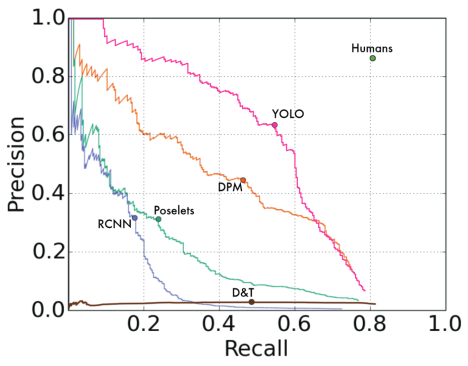
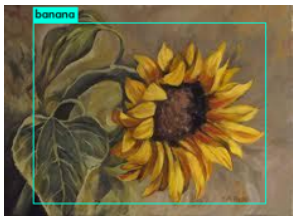
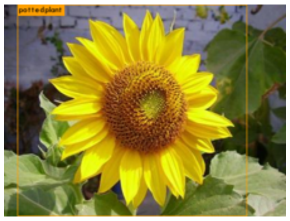


**Figure 6: Picasso Dataset precision-recall curves** [7]

## 4 Results

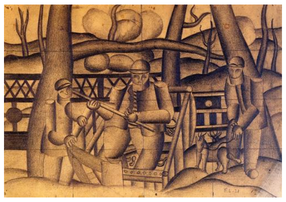


**(a)** Sunflower Picasso

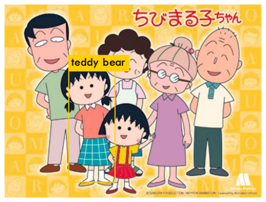


**(b)** Sunflower Picture

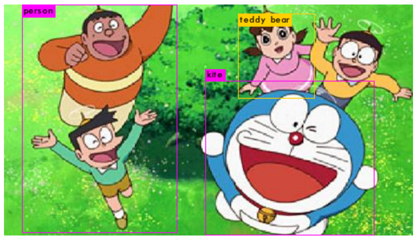
**Figure 7: Cross-depiction testing with YOLOv2.** (a) and (b) are the object images we did most testing on. In the photo, the sunflower is correctly classified as a plant but the painted sunflower is incorrectly classified as a banana.



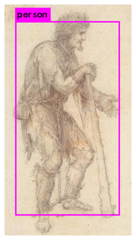
**(a)** Cubism



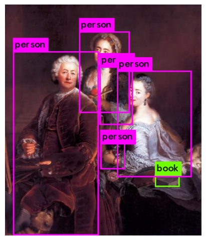
**(b)** Cartoon 1



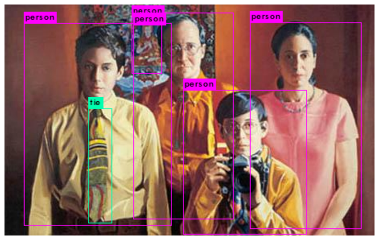
**(c)** Cartoon 2



**(d)** High Renaissance



**(e)** Rococo



**(f)** Photorealism

**Figure 8: YOLOv2 object detection of people in various artwork styles.** With artwork, style caused some misclassification of objects such as in (b), (c), (e). More realistic depictions such as (d), (e), (f) had correct detections but highly abstract cubism in (a) completely fooled the classifier.

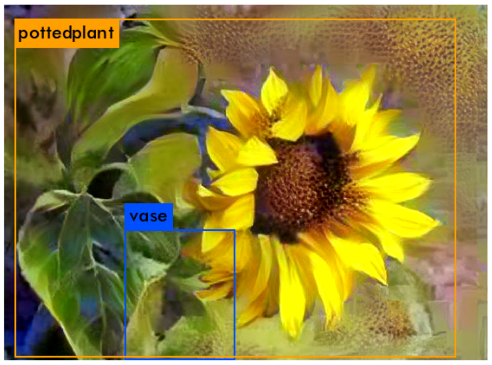
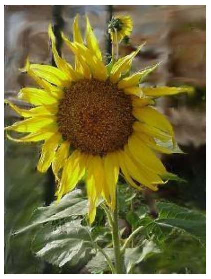
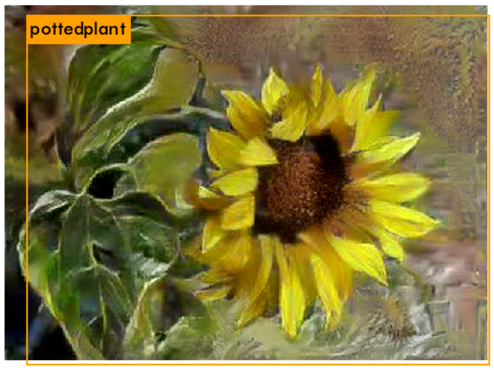


**Figure 9: Naïve style removal (Sunflower).** For this we simply reversed the input style and content images. We want the artwork to seem more realistic, so we want to transfer the style of the photograph onto the artwork. YOLO classifies this as a potted plant with 35% confidence – an improvement over the original banana prediction

**(a)** Sunflower photo 1

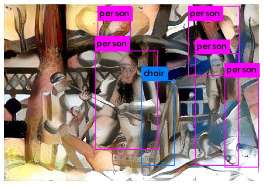


**(b)** Sunflower photo 2

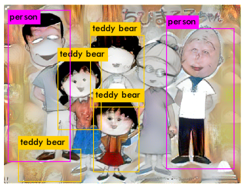


**(c)** Result from passing Image 7a through (a) and (b)

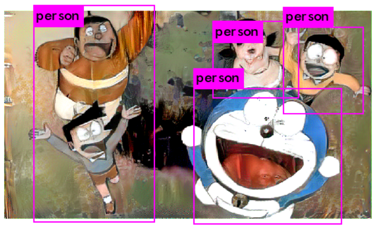
**Figure 10: Style Removal in Series (Sunflower).** For this we used two source images (a), (b). The original image was passed through the style transfers in series with the output of the first style transfer going in as the content image for the second style transfer. YOLO gives 27% confidence – lower than 9 but still correctly classified.



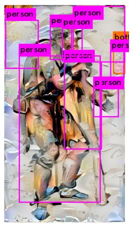
**(a)** Cubism



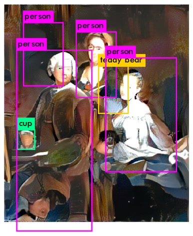
**(b)** Cartoon 1



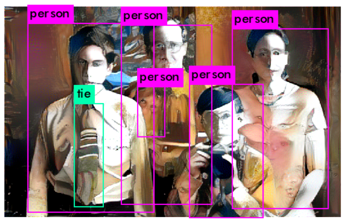
**(c)** Cartoon 2



**(d)** High Renaissance



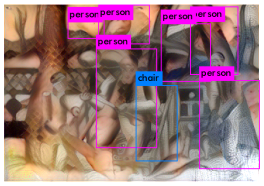
**(e)** Rococo



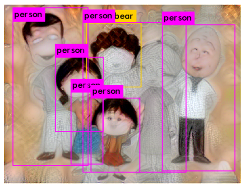
**(f)** Photorealism

**(g)** Photo of People (style)

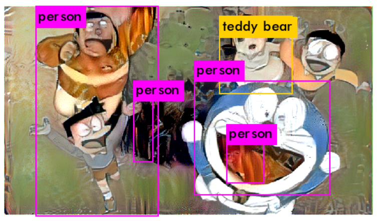
**Figure 11: Naïve style removal (Person).** This was done similarly to Figure 9, but with (g) as the source style and all the images (a)-(f) from 8



**(a)** Cubism



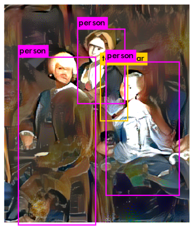
**(b)** Cartoon 1



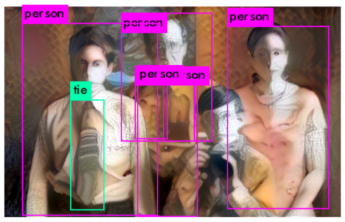
**(c)** Cartoon 2

person
person
person

**(d)** High Renaissance



**(e)** Rococo



**(f)** Photorealism

**(g)** Person photo 1

**(h)** Person photo 2

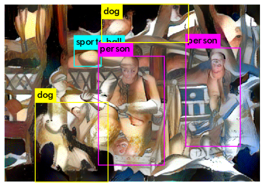
**Figure 12: Style Removal in Series (Person).** For this we used two source images (g), (h), with ordering set randomly. The procedure was the same as in Figure 10.

The two images in Figure 7 are test images for which we verified style transfer before applying our methods to the two datasets. We also did testing on images of different artistic styles in PeopleArt (see Figure 8). Images are annotated with object predictions from YOLOv2.

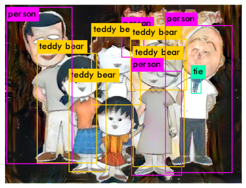
In Figure 9, we have an example of an image after applying style transfer for 500 iterations with style weights $\{1000/n^2 : n \in \{64, 128, 256, 512\}\}$. These values for weights were determined empirically. In this example, we see that style transfer significantly brightened the color and made it more realistic.

The two images in Figure 7 are the images from PhotoArt 50 for which we verified style transfer before applying our methods to the rest of the dataset as well as the PeopleArt dataset. We also did testing on images of different artistic styles in PeopleArt (see Figure 8). Sample images with object predictions from YOLOv2 are shown.

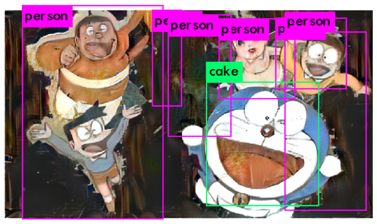
In Figure 9, we have an example of an image after applying style transfer for 500 iterations with style



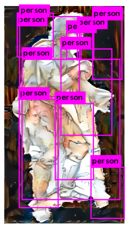
**(a)** Cubism



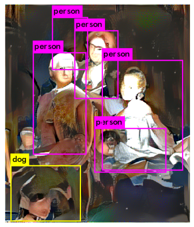
**(b)** Cartoon 1



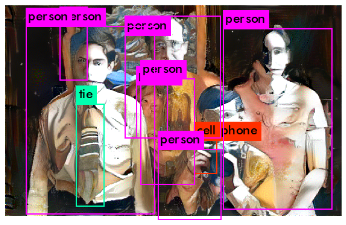
**(c)** Cartoon 2



**(d)** High Renaissance



**(e)** Rococo



**(f)** Photorealism

**Figure 13: Style removal with segmentation.** This was done similarly to Figure 11, but all the images (a)-(f) had segmentation applied before transferring style. Style transferred from 11g.

weights $\{1000/n^2 : n \in \{64, 128, 256, 512\}\}$. These values for weights were determined empirically. In this example, we see that style transfer significantly brightened the color and made it more realistic.

In Figure 10 we applied two style transfers to Image 7a in series, transferring from style image (a) before transferring from style image (b), with 400 iterations of style transfer each (800 total).

Figure 11 has examples of naïve style removal on the images we showed previously in Figure 8. We showcase the results of multi-source serial style transfer and segmented style transfer methods in Figures 12. and 13

## 5 Analysis

Surprisingly, naïve style removal did well qualitatively, improving classifications in most of our examples. Although we weren't able to get test data over the entire datasets, due to the prohibitive style transfer time, we were impressed that YOLO was able to correctly classify the objects after style removal. In particular, the style removal in Cubism in Figure 10(a) did surprisingly well – after not detecting any objects in the original, YOLO picked up on the person! We think some of this might be attributed some "bleeding" of content from the style image, since the style image has an instance of a person. More testing is required to confirm this hypothesis

Given the success of the basic style removal, we expected multi-source serial style removal to do similarly, if not better. We tried two different methods of serial style removal – one with 400 iterations per source image, and one with 400 total iterations, split evenly among the source images. In both methods, the sunflower detection confidence decreased, but the results on people improved. Of course, we need more conclusive testing on the entire dataset, but the interim results are quite promising. Again, we have some concerns about style bleed, and we'd like to try transferring style from images without people to see if we can reproduce similar results.

Finally, segmentation gave great performance as well, and qualitiatively we think it surpasses both of the previous methods. This makes sense intuitively – the foreground information is more important the background for object detection, and we're mostly detecting objects the foreground. Furthermore, segmentation typically detects one "blob" of objects, but it's possible that objects will be disconnected. For future testing we would also consider images where the object is in the background, and disconnected objects.

The choice of dataset also had an effect on the quality of the output image. The advantage of using the provided datasets was that they were already sorted by class; we could ensure both style and content images. In reality, it may not be possible to match the objects in both the content image and the style image. As a result, some of the incorrect content from the style image may become present in the content image, leading to incorrect classification.

That said, we think our results from testing style removal are very promising and raise some good areas for potential further research.

## 6 Conclusion

While we have good results from testing our methods overall and think style removal is an interesting area to continue researching, we also have some concerns that would need to be addressed if we were to continue future work.

We are actually uncertain as to how much style removal helps because the photographs we use to remove artistic styles could be changing the art in ways that artificially increase classification accuracy, especially since our style images were positive examples of the object depicted in the artwork. For example, consider a piece of art depicting a person that is initially misclassified by YOLO. It is possible that, after performing style removal on the art image using a photograph containing a person as the style image, the output is classified correctly not because the human in the artwork looks more realistic, but because the photograph used in the style image placed a feature in the output that YOLO classified as human.

While such a phenomenon shouldn't happen because we use Gram matrices of style images, we observed this exact trend when we accidentally performed style removal using photographs containing people as the style images and artwork of a sunflower as the content image. See Figure 14 below.

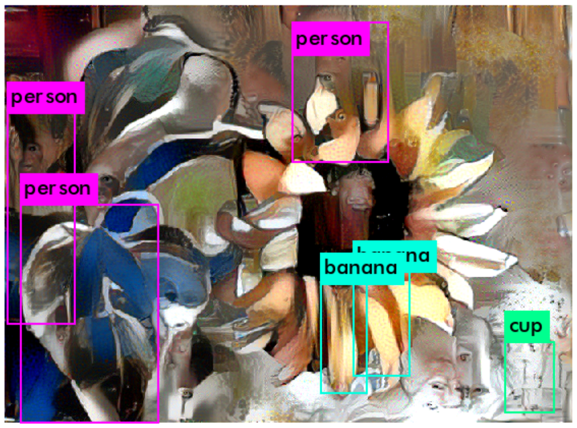


**Figure 14:** Classifying parts of a sunflower as a person after using people as style image.

The output image contained regions that YOLO classified as people, despite the fact that every image containing people was transformed into a Gram matrix before being used to compute the style loss.

One obstacle we ran into while conducting our research was that, while we could tell how well our object detection was performing from the confidence measure, we had no such indication of performance for style removal. If we were to move forward with this research, we would attempt to create a CNN for detecting artistic style which we could use to measure the performance of our style removal. With such a model, we could compare predicted art style of an image before and after applying style removal in order to get a more direct measure of how well our style removal methods work. In addition to providing this useful measurement, we could also use this classifier to determine which art styles do not need to have style removal performed on them in order to increase object detection accuracy. This would provide a marginal speedup in testing, especially when testing art styles that are already very realistic, such as photorealism.

## References

[1] Cai H, Wu Q, Hall P. Beyond photo-domain object recognition: Benchmarks for the cross-depiction problem. InProceedings of the IEEE International Conference on Computer Vision Workshops 2015 (pp. 1–6).

[2] Canny, J. A computational approach to edge detection. In IEEE Transactions on Pattern Analysis and Machine Intelligence 1986 (Vol. Pami-8 No. 6)

[3] Gatys LA, Ecker AS, Bethge M. Image style transfer using convolutional neural networks. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016 (pp. 2414–2423).

[4] Ghiasi G, Lee H, Kudlur M, Dumoulin V, Shlens J. Exploring the structure of a real-time, arbitrary neural artistic stylization network. arXiv preprint arXiv:1705.06830. 2017 May 18.

[5] Hall P, Cai H, Wu Q, Corradi T. Cross-depiction problem: Recognition and synthesis of photographs and artwork. Computational Visual Media. 2015 Jun 1;1(2):91–103.

[6] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. InAdvances in neural information processing systems 2015 (pp. 91–99).

[7] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016 (pp. 779–788).

[8] Westlake N, Cai H, Hall P. Detecting People in Artwork with CNNs. In European Conference on Computer Vision 2016 Oct 8 (pp. 825–841). Springer International Publishing.

[9] Wu Q, Cai H, Hall P. Learning graphs to model visual objects across different depictive styles. In European Conference on Computer Vision 2014 Sep 6 (pp. 313–328). Springer, Cham.