# EE5934 Deep Learning

## Project 2 Report

Submitted by

| | |
|---|---|
| Chen Yansong | A0232804M |
| Jin Lexuan | A0232696W |
| Zhang Xiaoyu | A0232760L |

*April, 2022*

# Contents

# LIST OF FIGURES

# 1. *Kaggle*: I’m Something of a Painter myself

## 1.1 Motivation and introduction of our algorithm

### 1.1.1 Introduction of CycleGAN

After the previous weeks of the course, we learned that we can build a Generative Adversarial Network (GAN) to transform the image style. For example, an ordinary landscape photo can be converted into a Monet style one, or a game screen can be transformed into a real-world screen, etc. This is called Domain Adaptation.

One of the solutions for this is Pix2Pix [1], but Pix2Pix requires the training data to be paired, and in real life, it is quite difficult to find images that appear in pairs in two domains (painting styles).

Therefore, CycleGAN [2] was born, which only needs data from two domains without the strict correspondence, which makes CycleGAN more widely used. In this project, the algorithm we implement is CycleGAN.

### 1.1.2 Implementation

Suppose we now have two datasets $X$ and $Y$ that hold landscape photos and Monet paintings, respectively. We want to train a generator $G$ that eats a landscape photo and spits out a Monet painting.

$$G(x) = y', x \in X \tag{1}$$

At the same time, we also want to train another generator $F$ that eats a

Monet painting and spits out a landscape photo.

$$F(y) = x', y \in Y \tag{2}$$

To achieve this, we also need to train two discriminators $D_X$ and $D_Y$ to judge the degree of imitation of the pictures generated by the two generators respectively:

If the picture $y'$ generated by the generator does not look like the picture $y$ in the dataset $Y$, then the discriminator $D_Y$ should give it a low score (with a specified minimum score of 0), and conversely if the picture $x'$ looks like the picture $x$ in the dataset $X$, then at this point discriminator $D_X$ should give it a high score (with a specified maximum of 1).

In addition, the discriminator $D_Y$ should always give a high score to the real image $y$. The same is true for the discriminator $D_X$. The whole model framework of CycleGAN is shown in the following figure.

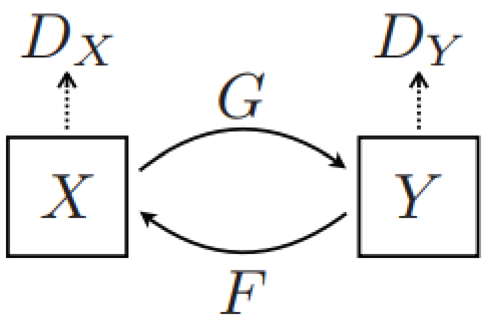


Figure 1.1 CycleGAN structure

When we fix the parameters of the generators $G$ *and* $F$ to train the discriminators $D_X$ *and* $D_Y$, the discriminators learn better discriminative techniques, and when we fix the parameters of the discriminators to train the generators, the generators rea forced to produce better quality images in order to fool the now more powerful discriminator. The two then evolve in this iterative learning process, eventually reaching a dynamic equilibrium.

In CycleGAN, we not only want the generator to produce images $y'$ that draw the same style as the images in the dataset $Y$, we also want the generator to produce images $y'$ that have the same content as his input images $x$.

To achieve this, we need to introduce Cycle Consistency Loss, which is to input $y'$ into $F$ again and make $x'' = F(y')$ as similar as possible to $x$. The same is true for generating $x$.

In other words, we hope that

$$F(G(x)) = x \tag{3}$$

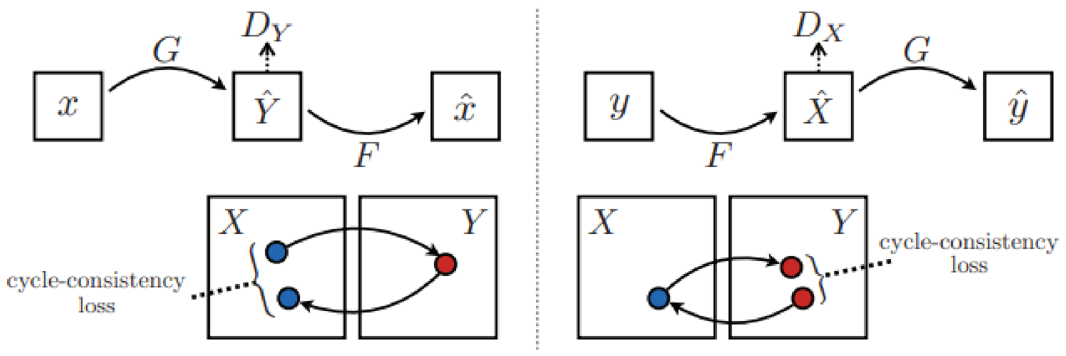


Figure 1.2 Cycle-consistency loss

Therefore, CycleGAN Loss consists of two parts:

$$Loss = Loss_{GAN} + Loss_{cycle} \tag{4}$$

$Loss_{GAN}$ ensures that the generator and discriminator evolve with each other, thus ensuring that the generator produces a more realistic picture.

$Loss_{cycle}$ ensures that the output image of the generator is only different in style from the input image, but the content is the same.

Our full objective is:

$$\begin{aligned}\mathcal{L}(G,F,D_X,D_Y) = {} & \mathcal{L}_{GAN}(G,D_Y,X,Y) + \mathcal{L}_{GAN}(F,D_X,Y,X) \\ & +\lambda\mathcal{L}_{cyc}(G,F)\end{aligned} \tag{5}$$

where:

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

and

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

We also add identity loss to further ensure that the content of the generated images remains unchanged.

$$\begin{aligned}\mathcal{L}(G,F,D_X,D_Y) = {} & \mathcal{L}_{GAN}(G,D_Y,X,Y) + \mathcal{L}_{GAN}(F,D_X,Y,X) \\ & +\lambda\mathcal{L}_{cyc}(G,F) + \mathcal{L}_{identity}(G,F)\end{aligned} \tag{6}$$

where:

$$\mathcal{L}_{\text{identity}}(G,F) = \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(y)-y\|_1] + \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(x)-x\|_1]$$

## 1.2 Performance evaluation: Style transfer examples and FID score

Initial effect: what is generated is basically noise.

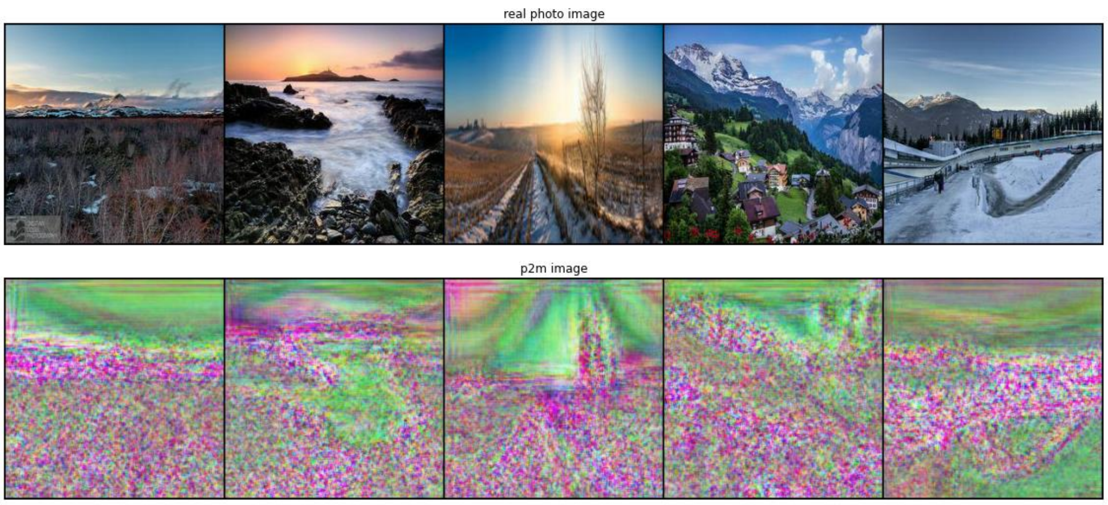


Figure 1.3 Effect without training

After 10 epochs:

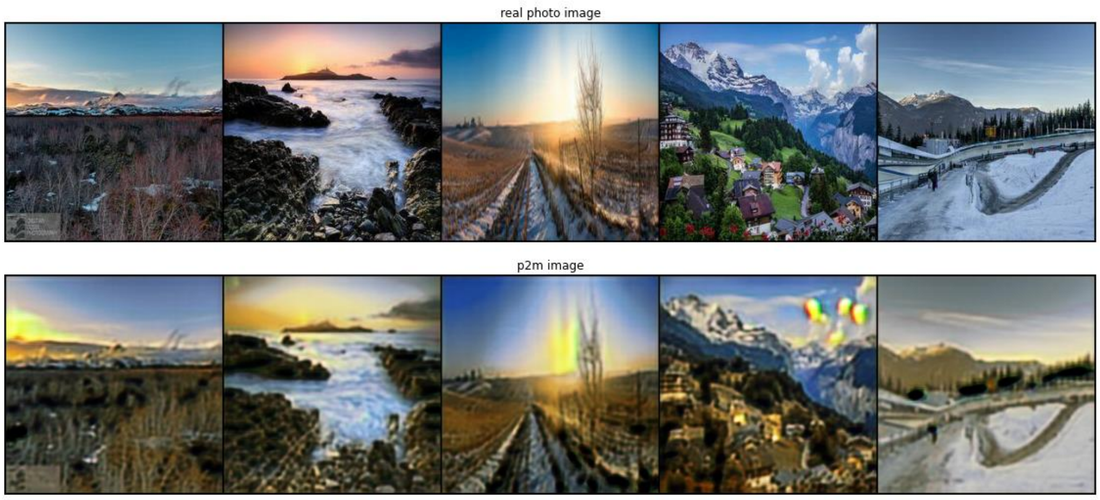


Figure 1.4 Result after training for 10 epochs

After 50 epochs(final):

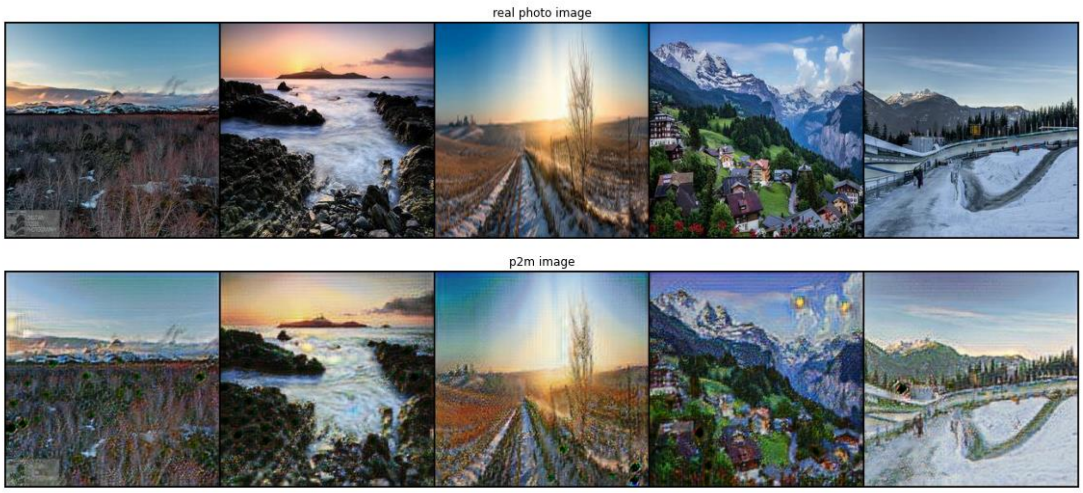
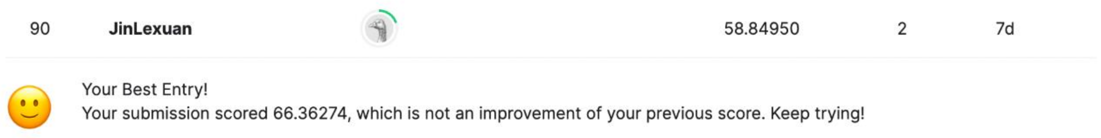


Figure 1.5 Final result after training for 50 epochs

Our best FID score is 58.84950



| | | | | |
|---|---|---|---|---|
| 90 | **JinLexuan** | 58.84950 | 2 | 7d |

Your Best Entry!
Your submission scored 66.36274, which is not an improvement of your previous score. Keep trying!

## 2. Saliency via Backprop Algorithm

### 2.1 Introduction of algorithm analysis

In recent years, deep Convolutional Networks have been typical choices for computer vision tasks due to considerate good performance. Therefore, researchers increasingly urgently want to better understand this "black box" for further refinement of CNN models. When it comes to the visualization of image classification models, Karen Simonyan[3] and his team implemented visualization of deep Convolutional Networks on ImageNet by proposing a computational method for saliency maps for certain image with given class label via back propagation. This proves the probability of generating understandable visualization through numerical optimization of input images, and shows the crucial role of gradients playing in building effective CNNs.

For a certain pre-trained classification model, we can generate the visualization figure of certain class with the following formula:

$$\arg\max_{I} S_c(I) - \lambda \|I\|_2^2,$$

where $\lambda$ is the regularization parameter for image $I$, and we want to obtain the maximum output value for class $c$. In contrast to the common function of back propagation in CNNs to update weights, we continually update the image to find local optimal image with fixed weights via back propagation. Note that $S_c$ here is the score before the processing of Softmax.

When it comes to a certain image with a given class label, we compute corresponding saliency map based on the $S_c$ mentioned above. Considering the non-linearity, we can approximate $S_c(I)$ with the first-order Taylor expansion:

$$S_c(I) \approx w^T I + b, w = \frac{\partial S_c}{\partial I}\Big|_{I_0}$$

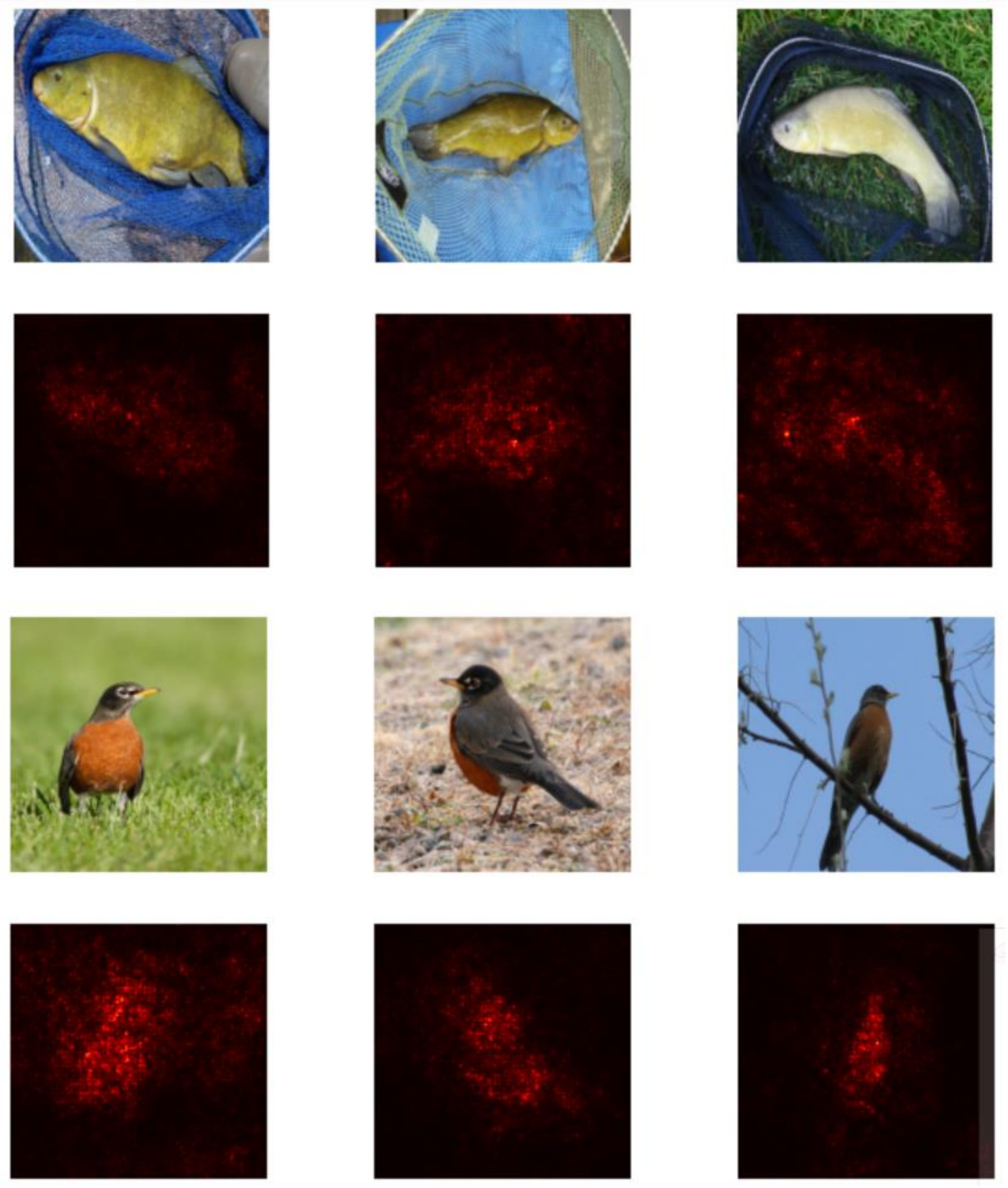
The value of gradient shows the influence of pixel at this position on classification score, then the saliency map indicates the position of target in the given image.

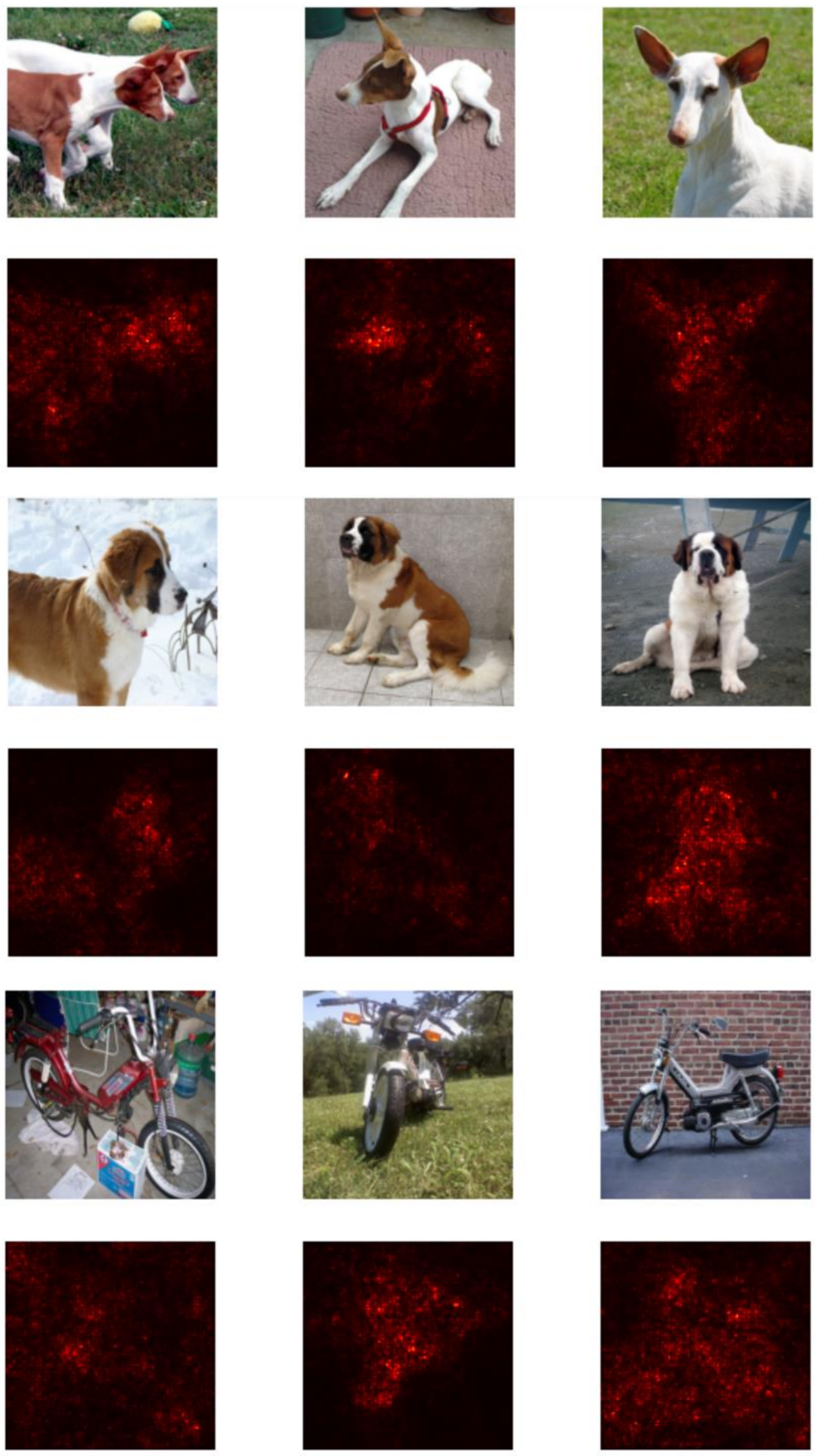
Moreover, the back propagation of gradients is similar to the deconvolution process. For the convolutional layer $X_{n+1} = X_n \star K_n$, the gradient is computed as $\partial f / \partial X_n = \partial f / \partial X_{n+1} \star \widehat{K_n}$ , which is part of the visualization algorithm. When we need to construct the feature map based on the DeconvNet, adopting backprop is thus reasonable. For our classification model SqueezeNet, we chose the classification score in the final fully-connected layer as the basis of visualization.

In our work, we designed our pre-process and post-process methods respectively for input images with certain class label and generated saliency maps considering better performance of our visualization model. With images after style transfer in task 1, I implement comparative visualizaition.

## 2.2 Qualitative saliency results

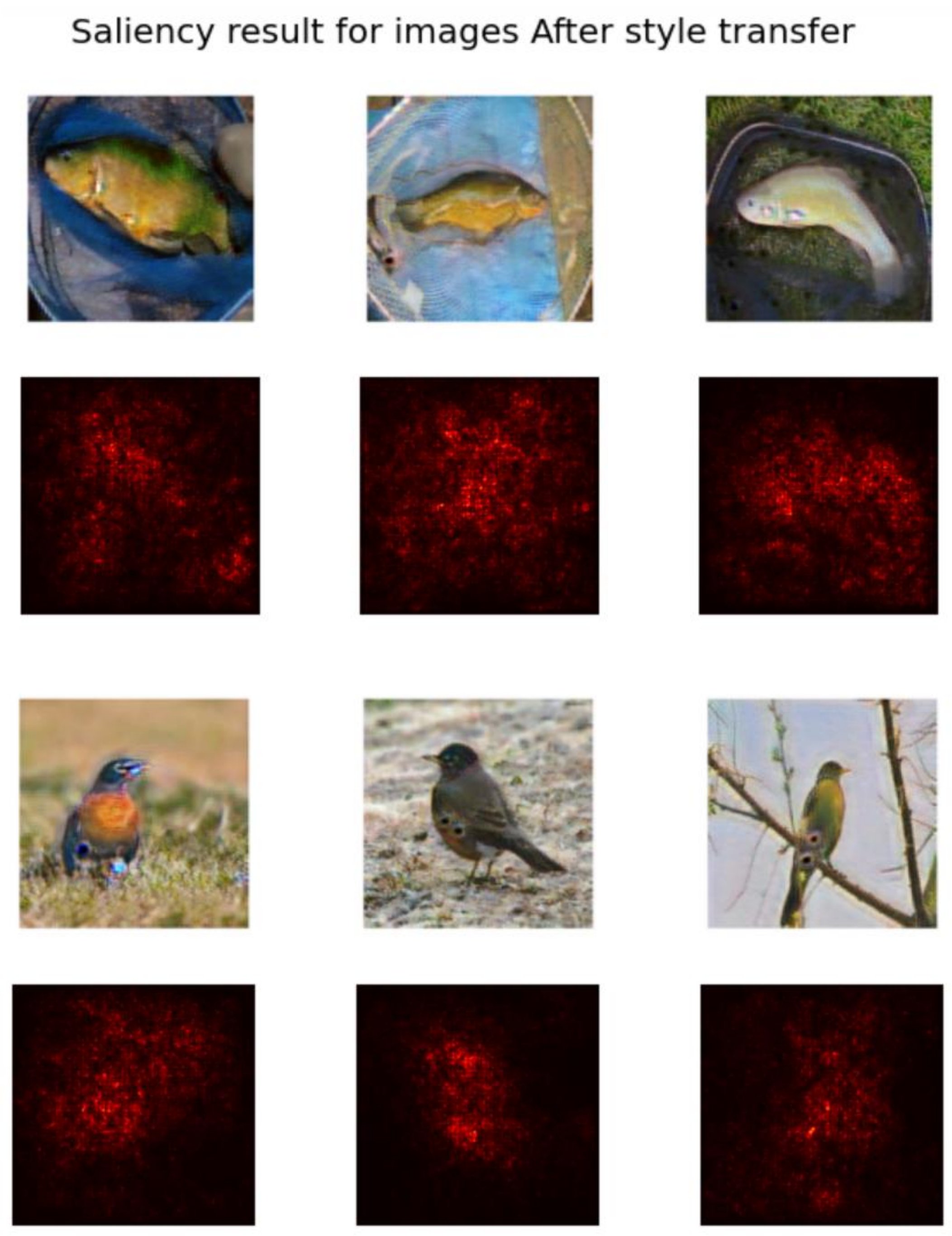
For the 15 input images provided in the zip file, we generate corresponding saliency maps plotted in heatmaps.

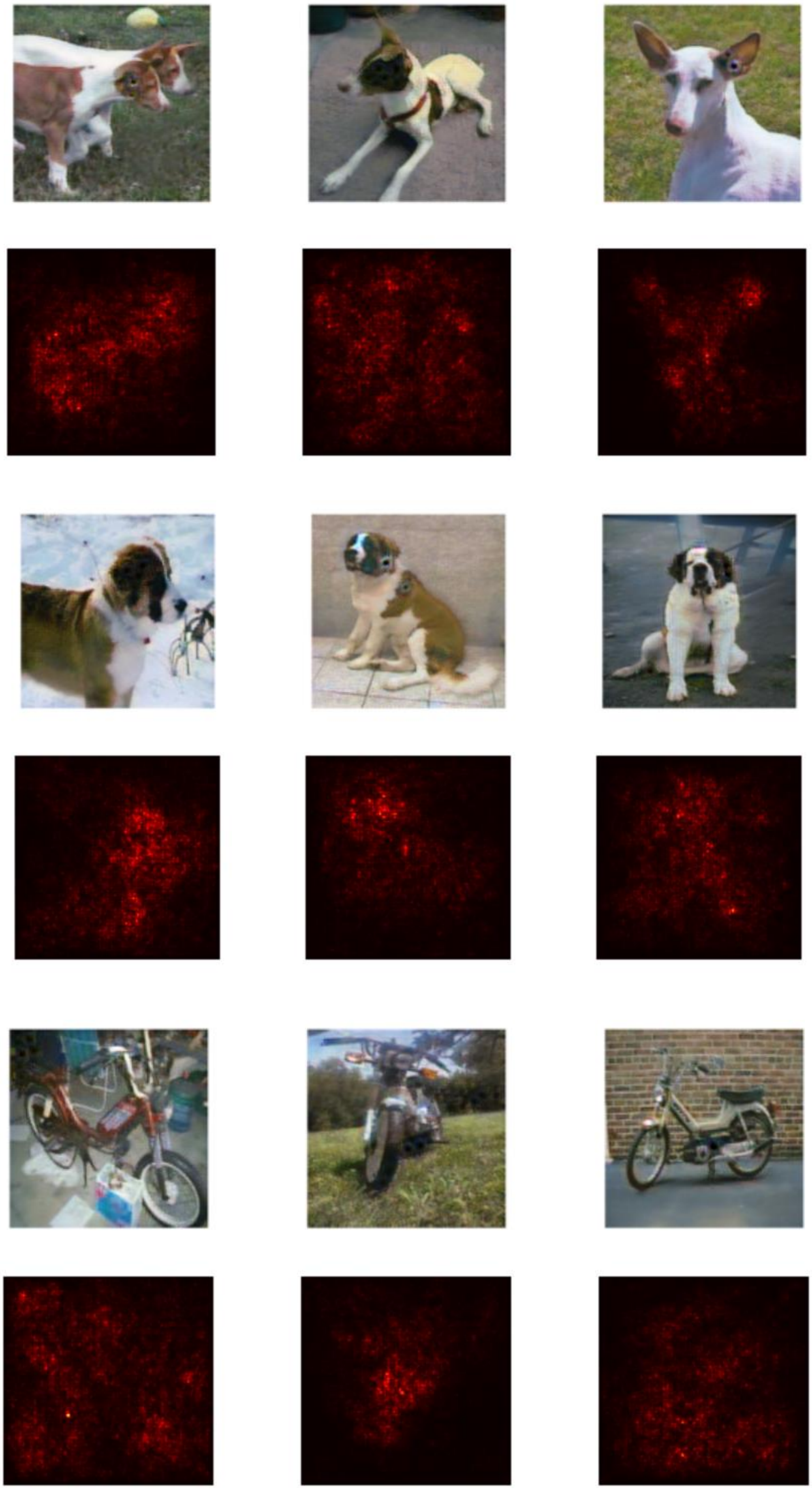




## 2.3 Complete comparison on visualization difference before and after style transfer

For the 15 input images processed by style transfer model in task 1, we load the images from the “transfer_images” folder and generate corresponding saliency maps as follows.





We can find directly from the figures above that the images after style transfer generate “poorer” saliency maps than the original images with the same computing algorithm. That means fewer features of targets in the images are captured and outlines of target can be harder to recognize.

## 3. Responsibility of each team member

For the teamwork of EE5934 DEEP LEARNING Project 2, we carefully analyzed all the tasks together and then determined the task assignment as follows due to high efficiency and reasonable workload for each member:

1) For the Kaggle competition of task 1 and style transfer for images in task 2: Chen Yansong (A0232804M) and Jin Lexuan (A0232696W)

2) For task 2: Zhang Xiaoyu (A0232760L)

## 4. References

[1]. Isola P, Zhu J Y, Zhou T, et al. Image-to-image translation with conditional adversarial networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1125-1134.

[2]. Zhu J Y, Park T, Isola P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2223-2232.

[3] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.