Analysis of the Neural Artistic Style Algorithm

Tianye Wang
Dalhousie University
Halifax, NS
tn610582@dal.ca

ABSTRACT

Deep neural networks have shown astonishing results in object recognition and detection, some of the models are achieved near human-level performance. However, when the task is to generate artistic artifacts, the deep neural network is far behind. This project takes advantage of pre-trained convolutional neural networks and combines famous artistic style references to generate arts and improve the original paper that introduce the algorithm of neural style transfer.

1 INTRODUCTION

Art is an expression of the authors imaginative or technical skills. In 40,800 BP some art drawing was discovered in a cave in Spain called El Castillo[1]. Humans already mastered the skills to create expressions to represent things. However, how a machine can replicate this process through algorithms with similar capabilities is unknown.

Neural style transfer can be considered as an image transformation problem. We want to have a photograph that has the style of painting. Figure 2 is an example of a photograph as a content image and a painting as a style image. Figure 3 is the output image after the neural style transfer process.

There are existing solutions to the neural style transfer problem. The paper by Gatys, Ecker, and Bethge use the representation of an image in a pertained image classification network to obtain the representation of an input image and a style image. The representations are used to generate a combined image which contains the content of the input image and the style of the style image through an optimization technique[2]. The problem of generating styled image through an optimization process is the computation time because optimization is iterative. Another approach to solving this problem to use an untrained image transformation network to generate the best guess at an appealing pastiche image. Then pass the pastiche image, the content image and the style image to a pre-trained image classification network[3]. The papers mentioned above does not provide specific instruction of which layer to use when performing the style transfer task. Thus, it is important to study the different configurations of the selection of the layers to use in the pre-trained network when performing neural style transfer tasks. The focus of this paper is to experimenting with different settings of the algorithm to see if the changes will improve the quality of the generated image. A series of experiments are designed to test the method proposed by Gatys, Ecker, and Bethge.

The remainder of this paper structured as follows. First, we present background information and the algorithm used in Gatys, Ecker and Bethge's paper. Next, we construct test cases for the experiment and discuss the reason why we choose them. Then, the initialization of the algorithm and termination condition are presented such that the experiment is reproducible. Finally, the experiment result, the discussion on the results and future work are discussed.

2 BACKGROUND

2.1 VGG-19

VGG-19 is a convolutional neural network developed by Visual Geometry Group for image classification. It classifies 1000 categories of objects in the ImageNet dataset, such as dog, cat, car, flowers and many other common objects[6]. The network has 19 layers which include convolution, pooling, and fully connected layers. The input of the network is a 224-by-224 image the output is a probability of the most likely object[4]. Figure 1[5] visualize the detailed architecture of the VGG-19 network. In the experiments, we use the activation of different layers in the VGG network to get the style and content representation. The following section will discuss the content and style representation in details.

2.1.1 Content and Style Representation. A trained convolutional neural network for object recognition develop representations of the image in each layer. If we reconstruct the image in each layer, when the layer gets deeper the representation of the image that cares more about the actual content of the image rather than the exact pixel value of the image[]. So, the content representation is the feature response in the deeper layer of the network. In addition, the style representation is defined as the correlations between the different filter responses in each layer. We use the gram matrix to obtain those correlations.[A Neural Algorithm of Artistic Style]

2.2 Neural Algorithm for Artistic Style

In the previous section, we discussed that we can use a pre-trained image classification network to get the content representation and the style representation of an image. The feature space of the VGG-19 network is used in this method to calculate the loss during the optimization process. Suppose we have an image as the input to the VGG network, the network gives the filter responses of

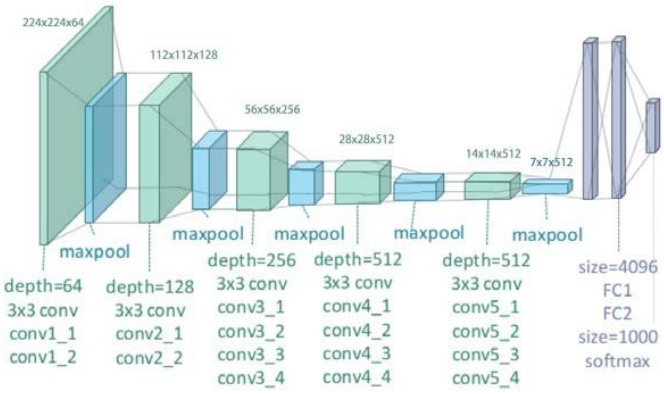


Figure 1: The Architecture of VGG-19 Network for Image Recognition

that image a_{ij}^{l} in each CNN layer where a_{ij}^{l} is the activations of the i^th filter at position j in layer l. Let C be the content image and G be the generated image. The content loss is defined as the square-error loss between the two feature representations:

$$L_{content}(C, G) = \frac{1}{2} \sum_{ij} (a_{ij}^{l\_C} - a_{ij}^{l\_G})^2 \tag{1}$$

it is also the L2 Norm of element wise subtraction between the two activation matrices of the image C and G.

Gram matrix is use to represent the correlations between the different filter response, ${G_{ij}}^{l}$ is the all possible inner products of the activations in layer l:

$${G_{ij}}^{l\_S} = \sum_{k} a_{ik} a_{jk'} \tag{2}$$

$${G_{ij}}^{l\_G} = \sum_{k} a_{ik} a_{jk'} \tag{3}$$

where ${G_{ij}}^{l\_S}$ is the gram matrix of the content image C, ${G_{ij}}^{l\_G}$ is the gram matrix of generated image G and k and k' are the pixel positions. Similar to the content loss function, we take the L2 norm of the two gram matrix. Then we define the contribution E_l in each layer l to be:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{ij} ({G_{ij}}^{l\_S} - {G_{ij}}^{l\_G})^2 \tag{4}$$

where N_l is the number of filters in layer l and M_l is the height times the width of that filter in layer l.

The total style loss is the weight sum of all the contribution in layer l of the loss network:

$$L_{style}(S, G) = \sum_{l} w_l E_l \quad (5)$$

where w_l is the weighting factor of layer l.

To generate the image that has the content of a photograph and the style of a painting we define the total loss to be the weight sum of the content loss and the style loss:

$$L_{total}(S, C, G) = \alpha L_{content}(C, G) + \beta L_{style}(S, G) \quad (6)$$

where α and β are the weighting factors of the content and style loss function. Finally, we use gradient descent to minimize this loss function by this updating rule:

$$G = G - \frac{\partial}{\partial G} L_{total}(S, C, G) \quad (7)$$

2.3 Total Variation Denoising

Noise can be introduced in an image during its creation, transmission or encoding. It makes the image difficult to interoperate. In the case of neural style transfer, noise in the generated image is inevitable, because the initialization of the resulting image is based on a randomly generated noise image. Total variation denoising is a method to remove the presence of noise in the image. The algorithm was originally proposed in 1992 by Rudin, Osher, and Fatemi[7]. The method is a constrained optimization type of numerical algorithm. The idea behind this algorithm is that when there are excessive or spurious details in the image the total variation of that image is high. High total variation means that when computing the integral of the absolute gradient the value of the image is large. The total variation loss L_{TV} is the sum of the absolute differences for neighboring pixel values in the input images:

$$L_{TV} = \sum_{i,j} |p_{i+1,j} - p_{i,j}| + |p_{i,j+1} - p_{i,j}| \quad (8)$$

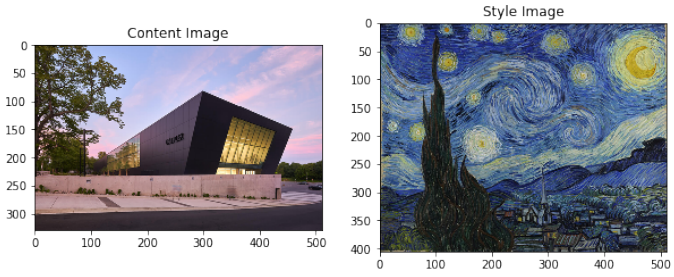


Figure 2: Content Image Dalhousie Gym, Style image Van Gogh Starry Night

where $p_{i+1,j}$ and $p_{i,j+1}$ are the neighbouring pixel of $p_{i,j}$.

3 EXPERIMENT SETTING UP

The programming environment of this experiment is in Python 3.6, running on a RTX 2070 GPU. The implementation is based on tensor-flow with eager execution[reference]. The hyper parameters of the implementation are set as follows: number of iterations =5000, content weight = 1e3, style weight = 1e-2. For the Adam optimizer the learning rate = 5, beta1 = 0.99, epsilon = 1e-1. In addition, the layer used for content loss is ['block5_conv1'], and the layers used for style loss are ['block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1', 'block5_conv1 Also, the generated image is initialized randomly with the same size of the content image using numpy random integer generation function. Finally, the error graphs are generated using log-scale on the y-axis.

4 EXPERIMENT RESULTS

4.1 Experiment 1:

The original paper[2] on neural style transfer use a random initialization of the generated image to optimize the loss function. In this experiment, the content image is used instead of using a randomly generated image as the initial image to optimize. The experiment shows that with the same number of iteration performed on the generated image, the

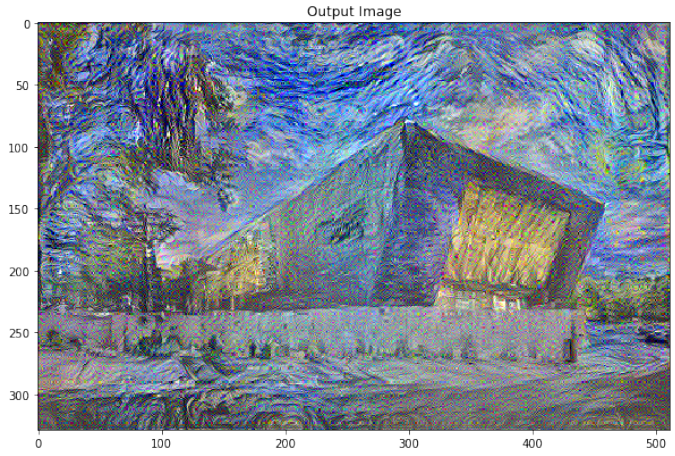


Figure 3: Output image base line

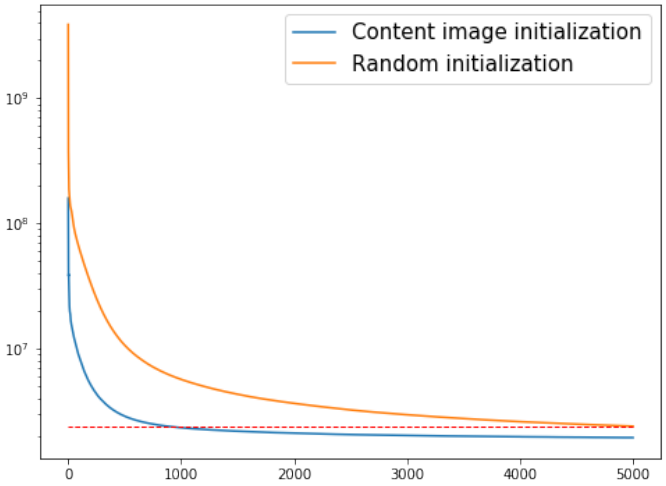
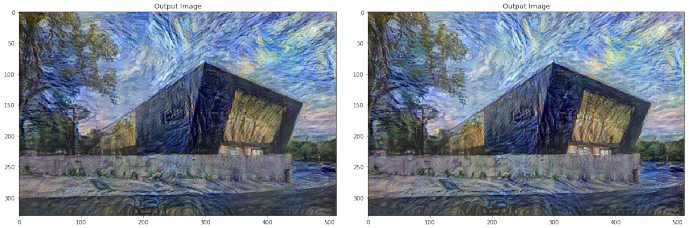


Figure 4: Total loss graph



(a) 5000 iteration **(b) 1000 iteration**

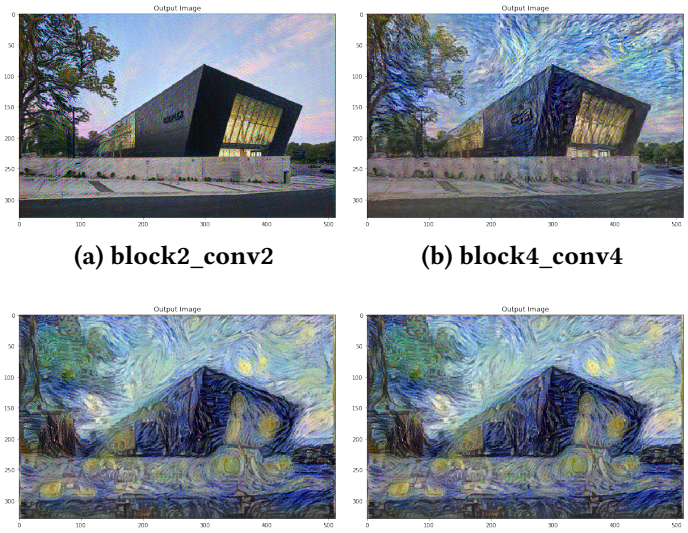
Figure 5: 2 Figures side by side

one which uses the content image as the initialization image gives a more visually appealing result faster. Faster means if we want to generate a similar level of content and style details of the image, the image that uses the content image as the initialization image requires less iteration to compute as shown in Figure 4. Figure 5 is a comparison of the output image of 5000 iterations and 1000 iterations. Besides, after comparing the baseline image and the image generated from the content image initialization have less noise in the output image. Therefore, to use the content image as the initialization image to optimize the output images gives a similar result in the loss function yet produce less noise image and require less computation time.

4.2 Experiment 2:

Empirical result of the original neural style transfer[artistic style] shows that deeper layers of the network gives a better mix of the content image and the style image. The experiment in the original paper only test on the first layer in the five blocks of the VGG network, namely ['block1_conv1'], ['block2_conv1'], ['block3_conv1'], ['block4_conv1'] and ['block5_conv1']. However, the performance of other intermediate layer and pooling layers of the network are unknown. In this experiment, initially we test the algorithm on ['block2_conv2'], ['block3_conv3'], ['block4_conv4'] and ['block5_conv4'] of the VGG network as the content representation layer to see it argument still holds. Figure 6 (a), (b), and (c) are examples of the output image. As expected, the layer goes deeper the better the mix of the two input image, In addition, the image generated use content representation in ['block5_conv4'] gives the best result compare to the original baseline image generated from ['block5_conv1'] and the rest of the generated image in both visually appealing and total loss side.

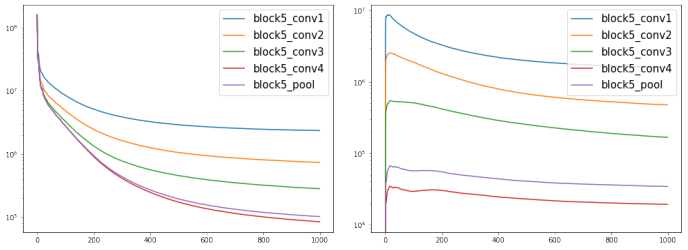
Later in the experiment, we test all the layers in block5. The result is shown in Figure 7. An interesting finding is that the image generated using the final pooling layer have higher total loss than its previous layer, but the resulting image has less noise. To sum up, the 4th layer in the 5th bock of the VGG-19 network gives a better result than the



(a) block2_conv2 (b) block4_conv4

(c) block5_conv4 (d) bolck5_pool

Figure 6: 2 Figures side by side



(a) total loss (b) content loss

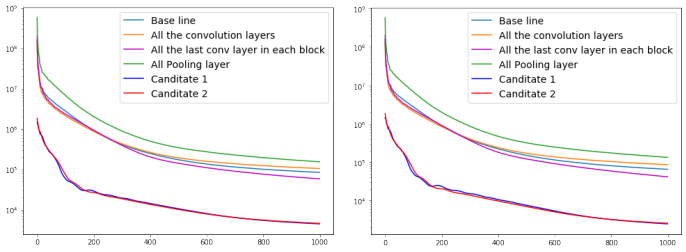
Figure 7: Loss comparison of all layers in block 5

original paper proposed layer. Also, as we move from lower layers to the deeper layers the color of the original image fades away the edges of the building are less sharp less original content is captured, but the overall content is well preserved. Thus the original argument still holds with one exception. The exception is using the pooling layer of the 5th block in the network increase the total loss, but reduce the noise of the output image.

4.3 Experiment 3:

In this experiment, we test the algorithm using different combinations of the gram matrix from different layers for the style loss. We use the best output image from the previous experiment in Figure 6 (c) as our baseline image. The experiment results are shown in Figure 8 and 9. We start with the hypothesis that by increases the number of the layer in the gram matrix the style loss will decrease and the algorithm will output image which contains more information about the style image. After the test, the result shows that compares to the original configuration of the style layer adding more layers will not decrease the style loos. The reason for that is because adding more layers will result in larger gram matrix and increase the computation time, so in 1000 iteration the total loss and style loss are increased. Next, we test the algorithm using deeper layers in the 5 blocks. We use ['block1_conv2', 'block2_conv2', 'block3_conv4', 'block4_conv4', 'block5_conv4'] as the layers to build the gram matrix. The output image is improved as shown in Figure 10, the circular shape brushstroke of the sky of the style image is not shown on the sidewalk of the output image using deeper style layers.

There is two candidate configuration that improves the baseline image by %96 on style loss and %94 on a total loss. From the content loss in Figure 9, an interesting finding is that the content loss of the candidate image is decreased, but the content representation layer was not changed. The first candidate (Figure 11) uses style layer: 'block1_conv1', 'block1_conv2', 'block2_conv1', 'block2_conv2', 'block3_conv1', 'block4_conv4', 'block5_conv4'. The second candidate (Figure 12) uses style layer: ['block1_conv1', 'block1_conv2', 'block2_conv1', 'block2_conv2', 'block3_conv2', 'block4_conv4', 'block5_conv4']. One explanation of why this configuration works well is that the number of layers selected well cover the key style of the image. As mentioned in paper[6] the shallow layer detects edge and color, so by adding the correlation of all the layers in the first two blocks of the VGG-network gives the best color of the style image. And based on the comparison of the baseline image(Figure 6 (c)) and Figure 11 and 12, the first



(a) Total loss (b) Style loss

Figure 8: Loss comparison of all configurations

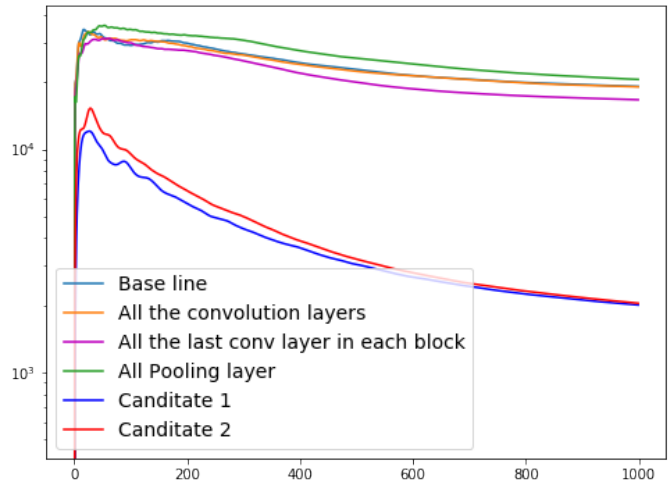
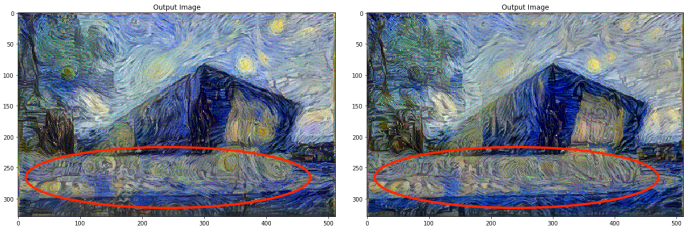


Figure 9: Experiment 3 Content Loss

two-layer in the third block of the VGG network gives the right balance between the correlation of the shallow layer and the deeper layer of the network. The two candidate configuration also reduces noise compares to the baseline image. To sum up, a key finding in this experiment is that the first two blocks in the VGG-19 network controls the color of the brush strokes of the image, the third layer controls the sharpness of the strokes of the image the rest of the layers controls the general shape of the brush stroke from the style image. The candidate configurations have %94 of improvements on the baseline image and produce less noise.



(a) Base line image (b) Deeper style layers

Figure 10: Loss comparison of all layers in block 5

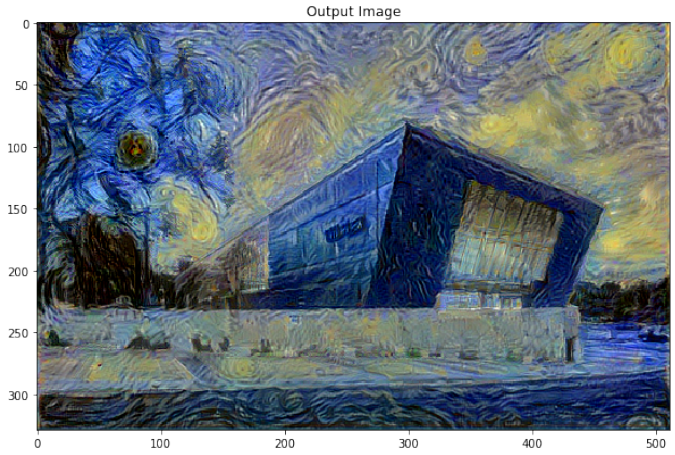


Figure 11: Candidate image 1

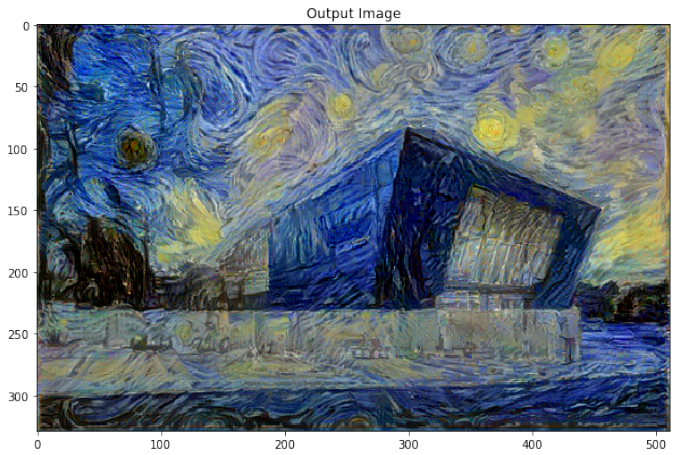


Figure 12: Candidate image 2

5 DISCUSSION AND FUTURE WORK

To summarize, during the experiment we found that use the content image as the initialization image instead of using random initialization to optimize the output image will reduce the computation time significantly. Also, when choosing the style representation layer of the VGG-19 network, layer 'bolck5_conv4' gives the best result compares to the other layers, but layer 'block5_pool' will generate a much smoother image without losing too much of the content information based on the content loss. Next, when choosing the style representation, we should find the right balance between the lower layer that represents the color and deeper layer that represents the overall brush stroke and structure of the style image. Finally, the best candidate image(Figure 12) shows significant improvements over the image generated from the original neural style transfer paper. The candidate image is more visually appealing and has a much lower total loss value.

Unfortunately, the total variation loss is not tested in this experiment. In the future, we will test the total variation loss against all of our experiment above. Also, a different method of neural style transfer should be explored, for example, take the results of our experiment and test it on the real-time neural style transfer. Finally, a more sophistic experiment should be designed and more robust validation of the generated image should be used to evaluate the resulting image instead of just focus on the loss graph.

6 REFERENCE

[1] El Castillo and Las Monedas https://cuevas.culturadecantabria.com/el-castillo-2/

[2] L. Gatys, A. S. Ecker, and M. Bethge A Neural Algorithm of Artistic Style, 2 Sept. 2015, arxiv.org/abs/1508.06576v2.

[3] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision, pages 694–711. Springer, 2016.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[5] N.A Transfer Learning in Tensorflow 7. June 2018 https://mc.ai/transfer-learning-in-tensorflow/

[6] Mahendran, A. & Vedaldi, A. Understanding Deep Image Representations by Inverting Them. arXiv:1412.0035 [cs] (2014). URL http://arxiv.org/abs/1412.0035. ArXiv: 1412.0035.

[7] Rudin, L. I.; Osher, S.; Fatemi, E. (1992). "Nonlinear total variation based noise removal algorithms". Physica D. 60 (1–4): 259–268. doi:10.1016/0167-2789(92)90242-f.