Deep Learning02: Feature Inversion

School of Computer Science and Engineering

17373240 Wanru Zhao

2020.5

Backgrounds

Learn the principle of representation inverting [1]. Approximate an image with its feature extracted from deep neural networks.

The representation inverting problem is formulated as finding an image x whose representation, e.g. extracted features best match the one given.

The representation inverting problem is formulated as finding an image whose representation, e.g. extracted features, best matches the given target， as $x = argmin_{x \in \mathbb{R}^{H \times W \times C}} L(\Phi(x) - \Phi_0 + \lambda R(x)$ where the loss L compares the representation of image x and that of the target Φ_0 and R serves as the regularization term.

- In this experiment we generate the representation using a pretrained vgg16 model with batch normalization added after each convolution layer from the onnx model zoo.
- We use L2 loss to compare the difference between representations.
- During the training process, the image x gets tuned and saved per 10 epochs. Check the stored images and find out how well this image approximates the target.

Tasks

Q1: How learning rate helps?

> *The baseline takes constant learning rate 1e-3. Modify the learning rate curve and pick one that achieves better convergence. You may use this configuration as your baseline.*

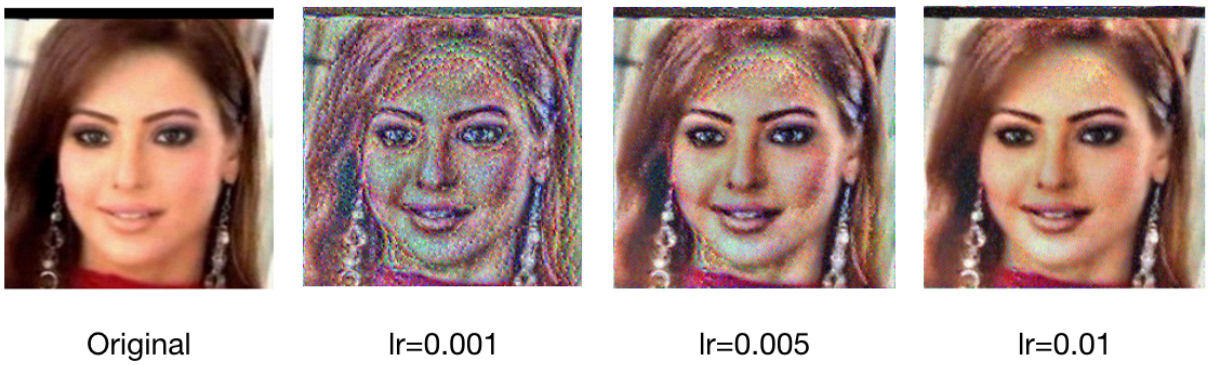


Fig1. Different performance by different learning rate

When we take constant learning rate 1e-2, it achieves the best convergence.

Q2: How regularization term helps?

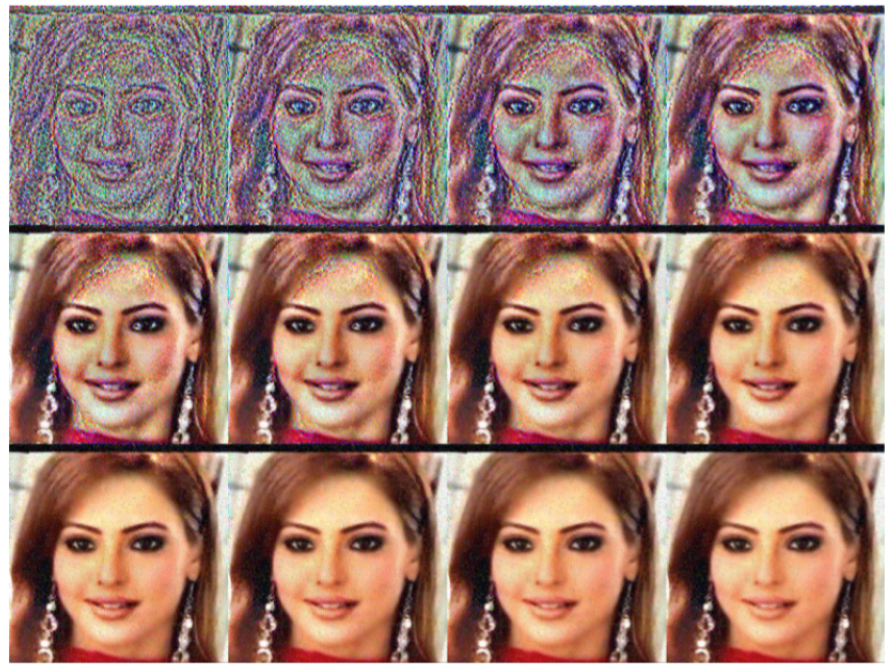
Total variation is defined as $\sum_{i,j} (y_{i+1,j} - y_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2$ where i,j denotes row and column index of a pixel in the image. Total variation loss encourages images to be made up of piece-wise constant patches.

> *Add total variation loss in the loss term and compare the generated images with that in Q1.*

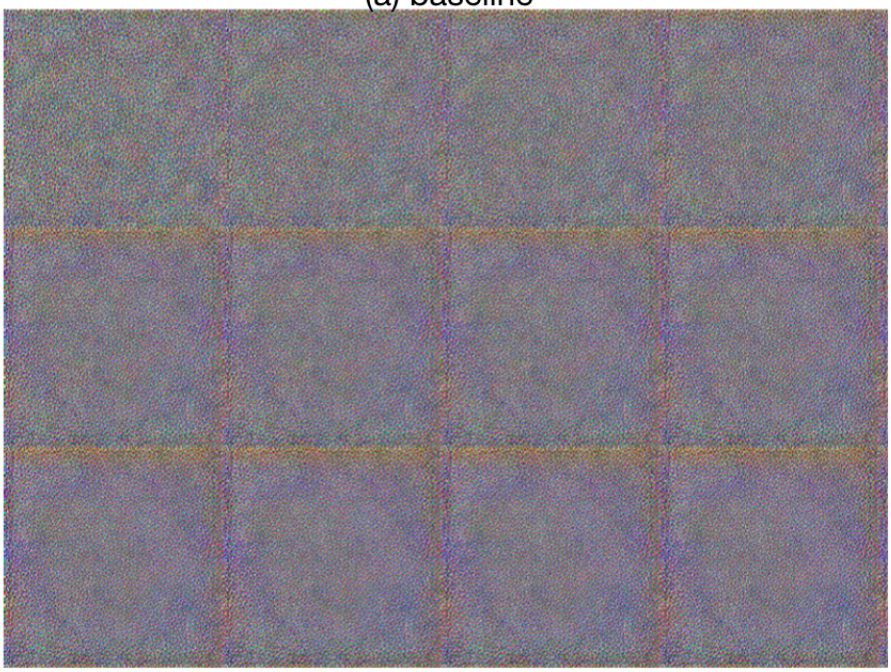
Codes

```
    tv_loss = tf.reduce_sum(tf.image.total_variation(noise_layer))
    total_loss = loss + tv_loss
```

Result



(a) baseline



(b) add total variation loss

Fig2. Generated images in Q2.1

Increase the portion of total variation loss in the loss term and rerun the training process.

Codes

```
    total_loss = loss + 10 * tv_loss
```

Result

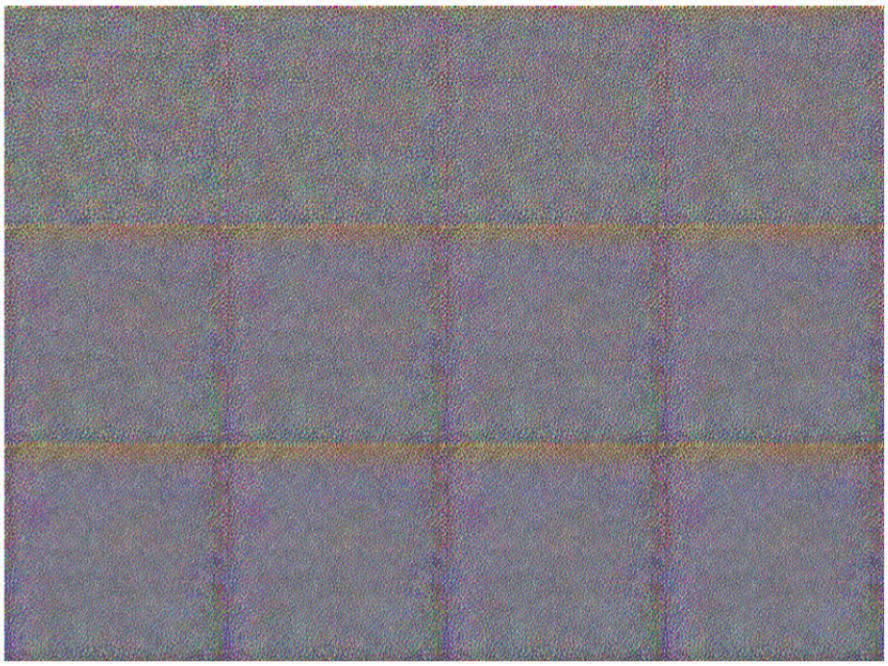


Fig3. Generated images in Q2.2

> *Increase the portion of feature/representation loss and rerun the training process.*

Codes

```
    total_loss = loss + 0.1 * tv_loss
```

Result

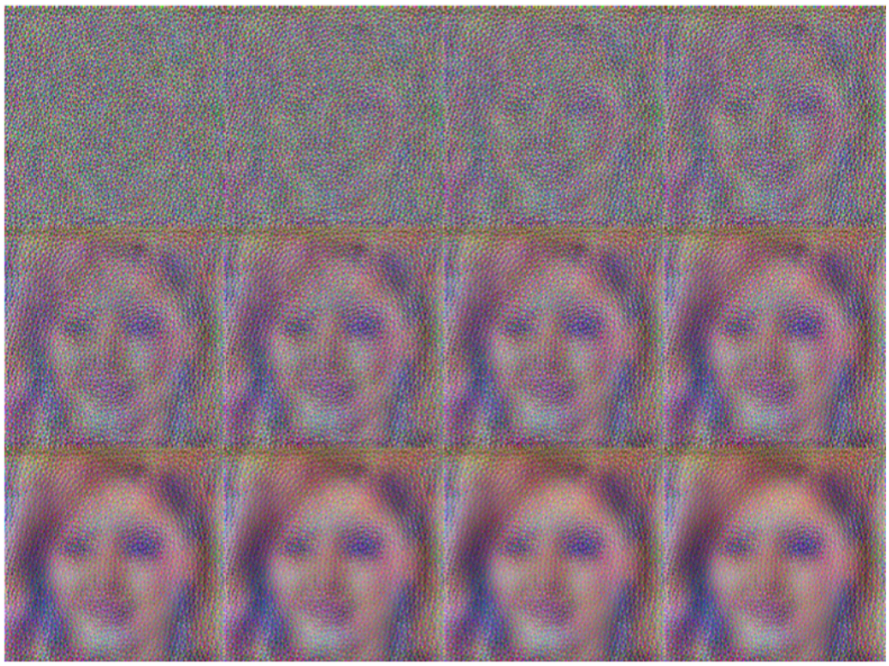


Fig4. Generated images in Q2.3

Q3: Which is more helpful, shallow or deep representation?

> *Use the features generated from 'conv1_1' and construct the representation loss. Rerun the training process.*

Codes

```
    FEATURE_LAYERS = ['conv1_1']
```

Result

Fig5. Generated images in Q3.1

> *Use the features generated from 'pool5' and construct the representation loss. Rerun the training process.*

Codes

```
    FEATURE_LAYERS = ['pool5']
```

Result

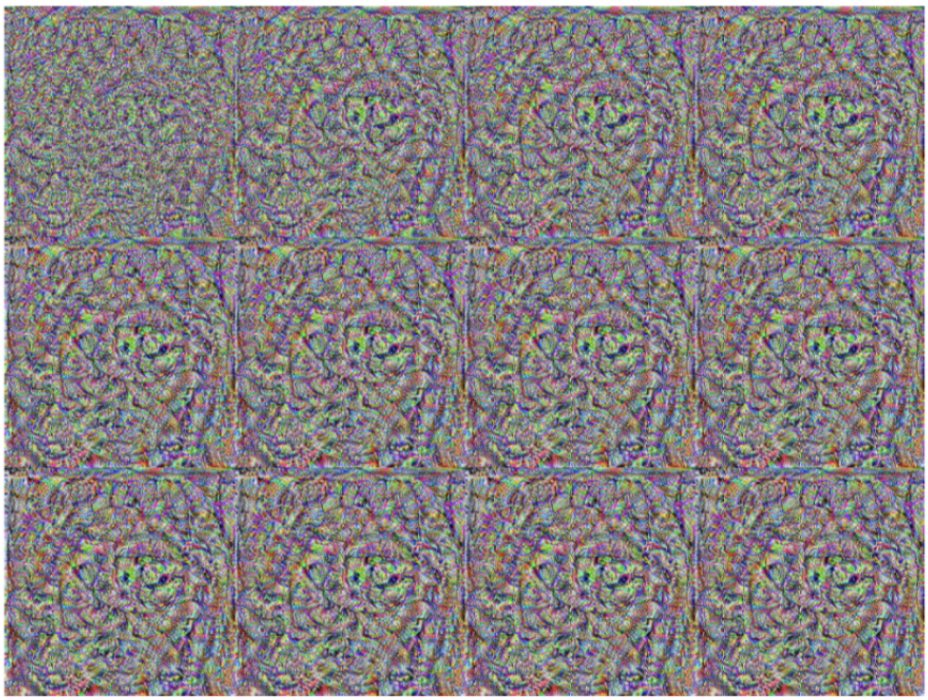


Fig6. Generated images in Q3.2

> *Construct representation loss from features from multiple layers. e.g. features from 'conv3_1' and 'fc6' and rerun the training process.*

Codes

```
    FEATURE_LAYERS = ['conv3_1', 'fc6']
```

Result

Fig7. Generated images in Q3.3

Analysis

After selecting a multi-layer representation, the overall result will be between the two representations.

Q4: What does different feature extraction model bring?

> *Change the feature extraction model from pretrained VGG16 model to Resnet18 model. Use the feature generated from second residual block , i.e. 'res2' to construct representation loss and rerun the training process. You may also play with other layes, e.g. the first convolution layer in each residual block, e.g.'middle1'.*

Codes

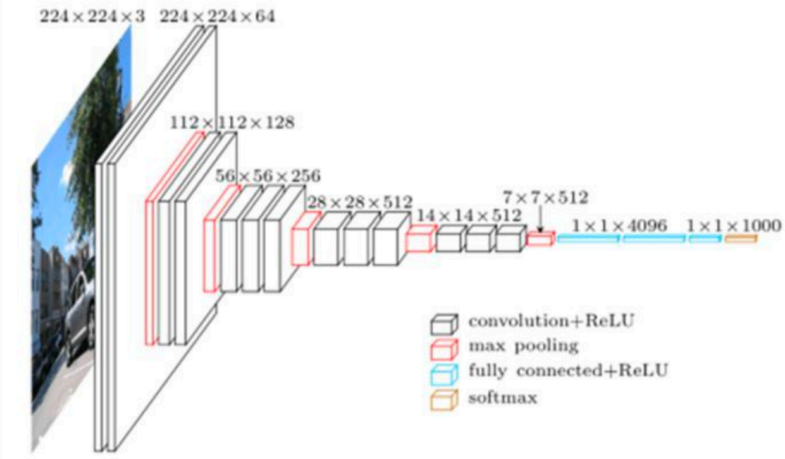


Fig8. Network structure of VGG16

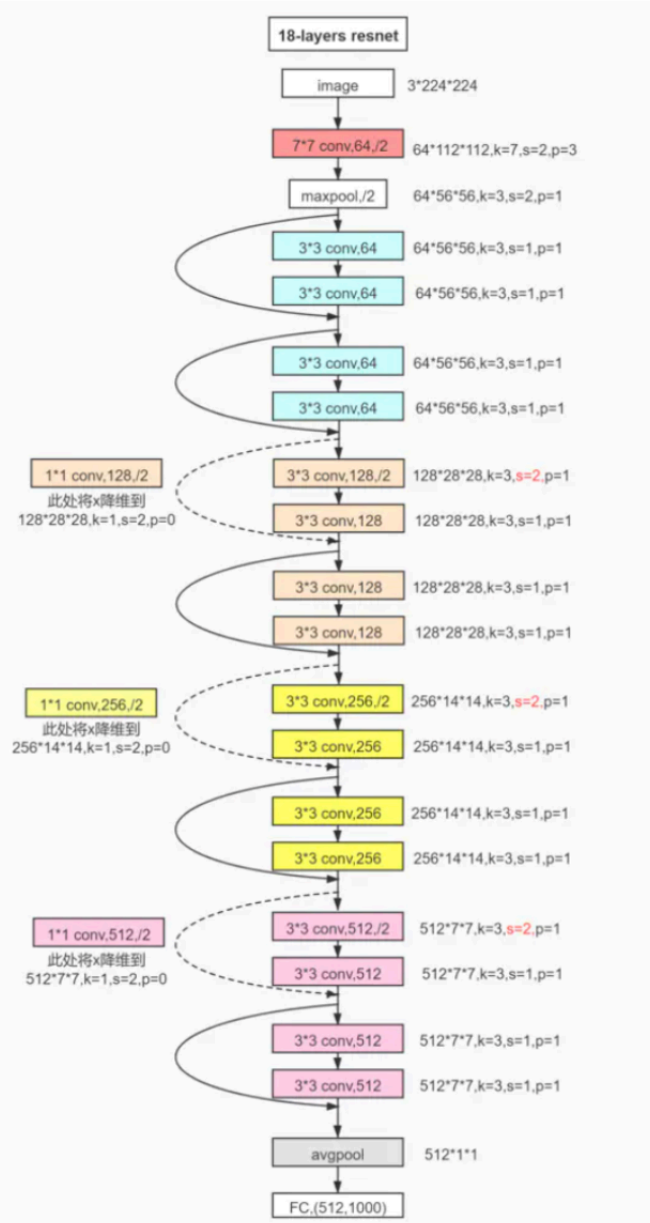


Fig9. Network structure of Resnet18

```
    import custom_resnet18 as resnet18

    FEATURE_LAYERS = ['res2']
#   FEATURE_LAYERS = ['middle1']
```

```
    '''get representation of the target image, which the noise image will approximate'''
    with tf.name_scope('resnet_src'):
        image_model = resnet18.Resnet18()
        image_model.build(image)

    '''get representation of the noise image'''
    with tf.name_scope('resnet_noise'):
        noise_model = resnet18.Resnet18()
        noise_model.build(noise)
```

Result

Fig10. Generated images in Q4

Analysis

- From the aspect of **running speed**, the running speed is greatly accelerated, mainly from the simplified parameters of Resnet.
- From the aspect of **performance**, Resnet18 model can better preserve the representation features than VGG16 model.

References

[1] Mahendran A, Vedaldi A. Understanding deep image representations by inverting them[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 5188-5196.