风格迁移

陈思聪 10195101402 王晨晖 10194304435

摘要： 近几年风格迁移出现了各种各样的实现方案。本文是数字图像课程的期末项目报告，同时也作为课程项目的数字图像处理系统的一个功能。本文的模型改进了课程提供的代码，从将 vgg16 改成 vgg19，ResidualBlock 改成 ResNextBlock，mse loss 改成 gram 三个角度改进模型，使其在小数据集上的效果超过原模型。同时使用的不同种类的风格图和内容图进行迁移，比较效果。另外，本文还进行了消融实验，探究了改进的这三个部分在风格迁移中起了什么作用。

关键字： 风格迁移，卷积神经网络

1 介绍

风格迁移是一个出现了很久的问题，它研究一个将一张图片的风格应用到另一张图片上的问题。从问题提出到现在，大量研究者提出了各种各样的模型和解决方法。比如，Rethinking Style Transfer: From Pixels to Parameterized Brushstrokes [1] 中提出需要将笔触考虑在内，能更好地还原风格图像的风格。StyleGAN3 [2] 则更加强大，可以实现表情、年龄、性别、肤色等等的风格迁移。DALL-E2 [3] 更为惊艳，基于 CLIP 实现了将文字转换成图片的功能。用文字对场景进行描述即可生成一张符合描述的图片，用文字描述一种风格来完成类似风格迁移的功能也非常令人惊喜。

本文是数字图像课程的期末项目报告，不涉及这些前沿的研究，主要对提供的论文和代码进行改进。本文涉及到的训练均在笔记本 cpu 上完成，因此后文提到的训练时间都特指在这个 cpu 上训练的时间。

2 相关工作

在 A neural algorithm of artistic style [4] 中，提出用图像识别的思路提取内容图和风格图的特征，使用 Gram 矩阵求风格的 loss。Gram 矩阵是对 feature map 做内积，就好像是高维空间的两个向量，向量之间的夹角 θ 越小，这两个向量就越相似，感觉类似 Word Embedding。总的 loss 是内容的 loss 和风格的 loss 的和，当这个 loss 变小，说明生成的图片和内容图、风格图都有相似之处，达成风格迁移的目标。但由于这个做法没有训练出模型，所以当内容图或风格图发生改变时，都要重新进行训练，生成一张图片要近 30 分钟，速度过于缓慢。将这样的算法嵌入期末项目的系统中显然是不合适的。

Perceptual losses for real-time style transfer and super-resolution [5] 提出了一种固定风格任意内容的风格迁移方法。通过在 COCO2014 上训练 TransformNet，得到某个风格图对应的模型。将这个模型应用在任意的内容图上都可以得到对应的迁移图像。然而，如果想要换一个风格，就要重新训练一个模型。如果想要实现用户上传两张图片，进行风格迁移的操作的话，这样的算法是不能满足需求的。对于用户上传的风格图，系统不知道该用哪一个模型进行风格迁移。因此，这个方法也是不合适的。

Meta networks for neural style transfer [6] 提出了任意风格任意内容的风格迁移方法。通过在 COCO2014 和 WikiArt 上训练一个 MetaNet，然后将训练出来的权值赋给 TransformNet，完成风格迁移。在训练的过程中，每次迭代都将 MetaNet 训练得到的权值赋值给 TransformNet，然后用 TransformNet 对图像进行操作，得到转换后的图像。用这个图像和内容图、风格图比较，求出内容

loss 和风格 loss。这个方法是满足系统的需求的，因此本文对这个模型进行改进。本文中，这篇文章中提出的模型称作原模型，本文的模型称作改进模型。改进的方向是在数据集比较小的情况下，如何让这个模型的表现超过原来的模型。

3 改进方法

由于论文使用数据集非常大，COCO2014 有 82783 张内容图，WikiArt 中也有 79433 张风格图。另外，目前设备的 GPU 显存只有 2G，运行论文的代码会爆显存，所以只能在 cpu 上训练。但是直接运行代码发现一个 epoch 就要 80 小时左右，22 个 epoch 也就是 73 天，这显然是不能接受的。所以本文从内容、风格数据集中分别提取出 500 张和 1000 张图片进行训练，缩短训练时间。每次取图片的 seed 均为 42。本文“大小为 500 的数据集”指内容图 500 张和风格图 500 张。“大小为 1000 的数据集”指内容图 1000 张和风格图 1000 张

3.1 替换 backbone

原来的网络中使用的是 vgg16 来提取图像特征，所以想到改用 vgg19。原先模型使用的是 vgg16 的 3, 8, 5, 22 层，根据图 1，这些层都是 maxpooling 的前一层。所以换成 vgg19 后，应该相应地使用 3, 8, 17, 26 层。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

图 1: vgg [7]

3.2 替换 residual block

根据图 2，原来的 TransformNet 在上采样和下采样之间有 5 个 ResidualBlock，所以想到 ResNet 的升级版 ResNeXt [8]，用一个 ResNeXtBlock 代替原来的 ResidualBlock。ResNeXtBlock 的入通

道和出通道保持相同，内部有 32 个相同的分支。没有采用 5 个 ResNeXtBlock 的原因是这样的结构非常消耗计算资源，一个 ResNextBlock 带来的时间开销已经大幅超过了 5 个 ResidualBlock。原先的模型训练 1 个 epoch 只需要约 25 分钟，修改后的模型则需要约 35 分钟。

这也是合理的，相当于用 32 个分支代替了 5 个串行的结构，训练量增加了。但本文没有将 ResNextBlock 所有的参数都传入 TransformNet 中，中间 32 个分支的参数没有存入 TransformNet，因为这些分支参数量相当大，没有必要放入 TransformNet，而且放入的话内存不够，所以舍弃。

layer	activation size
input	$3 \times 256 \times 256$
Reflection Padding (40×40)	$3 \times 336 \times 336$
$8 \times 9 \times 9$ conv, stride 1	$8 \times 336 \times 336$
$16 \times 3 \times 3$ conv, stride 2	$16 \times 168 \times 168$
$32 \times 3 \times 3$ conv, stride 2	$32 \times 84 \times 84$
Residual block, 32 filters	$32 \times 80 \times 80$
Residual block, 32 filters	$32 \times 76 \times 76$
Residual block, 32 filters	$32 \times 72 \times 72$
Residual block, 32 filters	$32 \times 68 \times 68$
Residual block, 32 filters	$32 \times 64 \times 64$
$16 \times 3 \times 3$ deconv, stride 2	$16 \times 128 \times 128$
$8 \times 3 \times 3$ deconv, stride 2	$8 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

图 2: TransformNet [6]

3.3 替换 style loss

A neural algorithm of artistic style [4] Gram 矩阵适合用来比较图像的相似度，所以这里用 Gram 矩阵代替原来训练用的 MSE loss。由于修改了 loss 计算方法，所以 style loss 的 weight 也需要做调整。为了让 style loss 和 content loss 的值相近，weight 选择 3e5。

4 实验结果

训练时间显著增长。表 1 展示的是两个模型大致的训练时间。

表 1: 训练时间

	500	**1000**
原模型	22min	50min
改进模型	35min	80min

可以看出数据集翻倍后，训练花费的时间近似于翻倍。改进模型花费的时间约为原模型的 1.6 倍。相应的，用模型进行风格迁移的时间也变长了。由于原模型风格迁移的过程是先指定风格图片，训练 20 个 batch，得到一个类似 Perceptual losses for real-time style transfer and super-resolution [5] 中的固定风格任意内容的风格迁移模型，然后将该模型应用到大量图片上，所以风格迁移的时间是 20 个 batch 的训练时间（较长）+ 转换时间（与图片数量有关）。因此这里比较的是转换一张图的时间，近似认为是 20 个 batch 的训练时间。原模型大约是 8-10 分钟，改进模型大约是 12-14 分钟。

模型文件缩小。由于改进模型只保存了 ResNeXtBlock 的两层，相比于原模型的五层 ResidualBlock，模型文件大幅缩小。原模型的两个模型文件约为 850 MB，改进模型的两个模型文件约为 647 MB，约为原模型的 $\frac{3}{4}$。

小数据集。相比于原模型的 80K 的训练，改进模型的训练集仅为 $\frac{1}{160}$ 和 $\frac{1}{80}$，大幅减少了训练时间，并且二者在大小为 500 和 1000 的数据集上训练后，进行风格迁移的效果差不多，甚至改进

模型更好，见附录 A 选择的风格图 3 色调上包含红、黄、绿、蓝、黑、白等常见的颜色，内容上包含人像、自然景物、建筑等，绘画流派上包含了抽象派、印象派等。依次为 18993，29052，30925，78717，mosaic，带珍珠耳环的少女，picasso，星月夜。前四个是 COCO2014 中的编号。后文以这些名称指代这些风格图。选择的内容图 4 主要为人像、动物、建筑、景物和文字，其中两张是黑白图。另外，在这 8 张风格图和 8 张内容图中，各有一半不在训练集中。

数据集内的风格图

数据集外的风格图

图 3: 风格图

数据集内的内容图

数据集外的内容图

图 4: 内容图

从附录的图中可以看出，原模型的迁移结果有许多斑点，仿佛绘画的时候颜料撒了，在画布上晕染开来，而且色块颜色和其他地方非常不协调。而改进模型没有这个问题。尤其是在大小 500 的数据集上训练的改进模型，整体颜色都比较协调。在大小 1000 的数据集上训练的改进模型虽然也有一些突出的色块，但色块的颜色不突兀，出现的位置相比原模型也更合理，都是一些原图明亮的、偏白色的位置。但目前并不清楚为什么数据集变大了会出现这样的现象。

权值不停增大的问题依然存在。由于没有在 80K 的数据集上训练，所以无法得出解决权值不停增大的结论。图 5 显示 transform_net.downsampling.1.weight 的权值。可以看出改进模型权值上升的趋势明显，至少在大小 500 和 1000 上权值在不停增大，而不是稳定在一个区间内。另外，改进模型权值的上升速度比原模型快，相同的迭代次数下，改进模型的权值普遍比原模型高，这可能也是改进模型效果更好的原因。

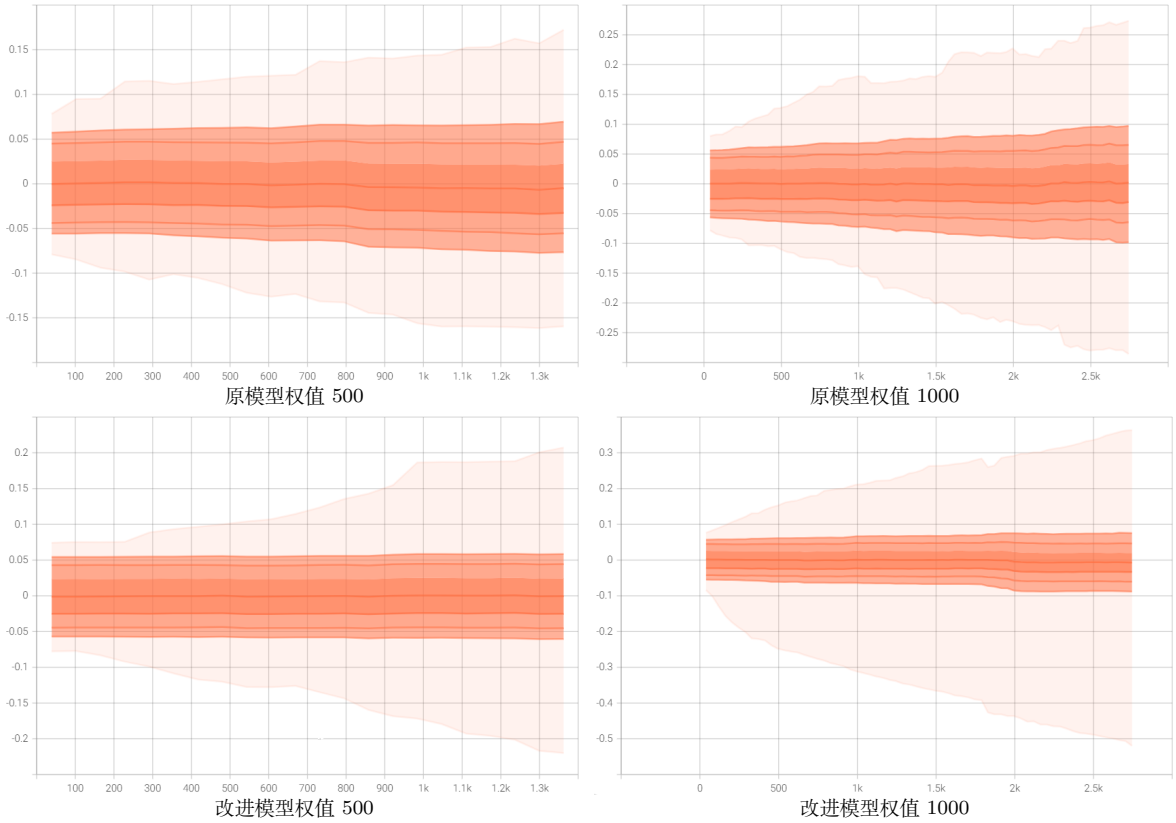


图 5: 模型权值

由于改进模型修改了原模型的三个地方，所以本文做了消融实验探究各个地方的改进分别对风格迁移起了什么作用，见附录 B。这部分的实验都是在大小为 500 的数据集上进行。

从图中可以看出，vgg16+resnext+gram 处理白色的效果不好。虽然在大小为 1000 的数据集上训练的改进模型在原图明亮的位置也用了和周围不同的颜色，但饱和度低，和周围的颜色是相近、协调的。而 vgg16+resnext+gram 的版本用了饱和度很高、鲜艳的颜色代替原图中明亮的位置，非常突兀。vgg19+resnet+gram 处理黑色的效果不好。在原图黑色或偏暗的位置出现了饱和度很高的颜色，如红色、绿色、白色，并且这些颜色出现的位置不连续。vgg19+resnext+mse 处理阴影的效果不好。在人物影子、水面倒影、云层阴影的位置都用了和周围相差很大、不协调、饱和度很低的颜色。

所以推测 vgg19 能更好地理解图像，能更正确地按语义分割图像，从而在合适的位置替换合适的颜色。resnext 和 gram 则是优化了黑色和阴影的转换，是图像的颜色更协调、线条更平滑。

5 总结

本文通过分析三种风格迁移的方案，根据方案的可拓展性和运行时间确定可以在数字图像处理系统中应用的方案，并对该方案的模型进行改进。本文从将 vgg16 改成 vgg19，ResidualBlock 改成 ResNextBlock，mse loss 改成 gram 三个角度改进模型，使其在小数据集上的效果超过原模型。同时使用的不同种类的风格图和内容图进行迁移，比较效果。可以看出改进模型的风格迁移效果要优

于原模型，但代价是更多的训练时间和迁移时间。另外，本文还进行了消融实验，探究了改进的这三个部分在风格迁移中起了什么作用，得出修改的每一部分对模型的效果都起到了作用。遗憾的一点是受制于设备，无法让其在 80K 的数据集上进行训练，所以也无法得知在大数据集上，改进模型的效果是否优于原模型。

而且，即使在《18993》，《29052》，《78717》，《星月夜》等风格图上取得了不错的效果，但在转换一些特定的图片上，模型的效果并不好。比如《30925》或《戴珍珠耳环的少女》。这两张图都有占比较少的、使画面丰富的颜色，如亮绿色和淡蓝色。可以看出原模型尝试在图像中引入这些颜色，虽然这些颜色出现的位置非常奇怪。但改进模型迁移的结果中完全没有这些颜色，取而代之的是占据大部分画面的黄色和黑色。虽然改进模型的结果更自然，但很难让人从迁移结果中联想出风格图的内容。

另外，原模型和改进模型都有图像不清晰的问题。因此模型还有许多地方有待改进，但受制于时间等因素，本文没能继续进行改进。

参考文献

[1] D. Kotovenko, M. Wright, A. Heimbrecht, and B. Ommer, "Rethinking style transfer: From pixels to parameterized brushstrokes," *CVPR*, 2021.

[2] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Proc. NeurIPS*, 2021.

[3] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[4] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[5] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[6] F. Shen, S. Yan, and G. Zeng, "Meta networks for neural style transfer," *arXiv preprint arXiv:1709.04111*, 2017.

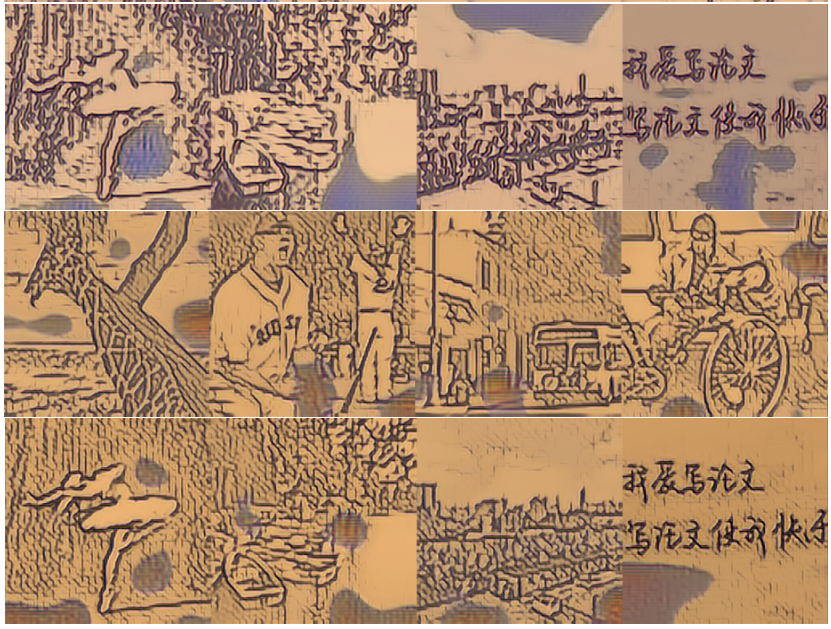
[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[8] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

A 效果对比

原模型 500

原模型 1000



改进模型 500

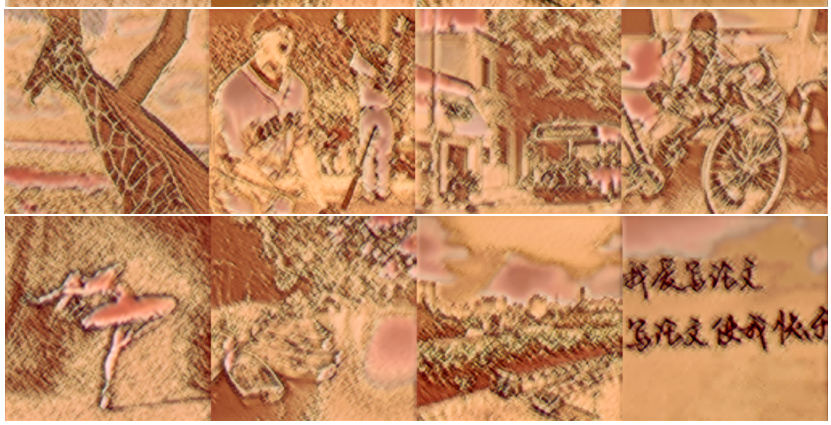
改进模型 1000

原模型 500

原模型 1000

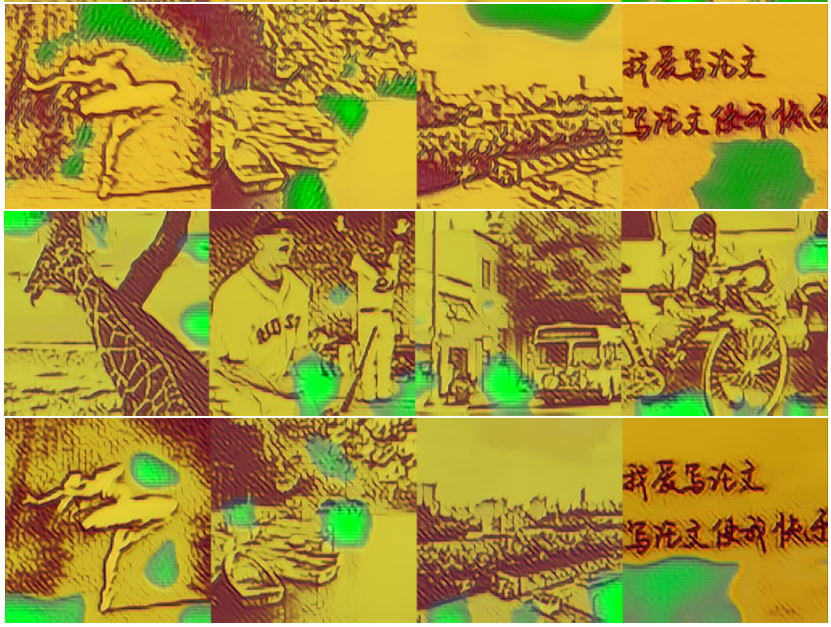
改进模型 500

改进模型 1000



原模型 500

原模型 1000



改进模型 500

改进模型 1000

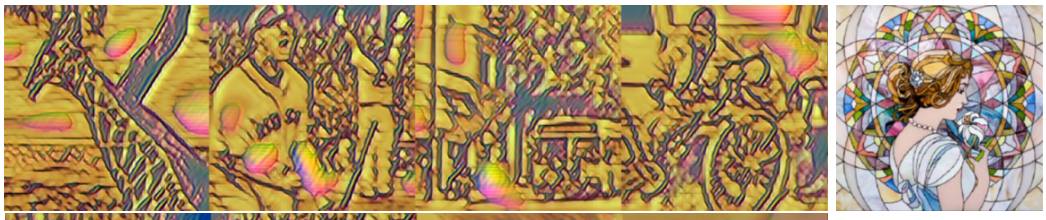
原模型 500

原模型 1000

改进模型 500

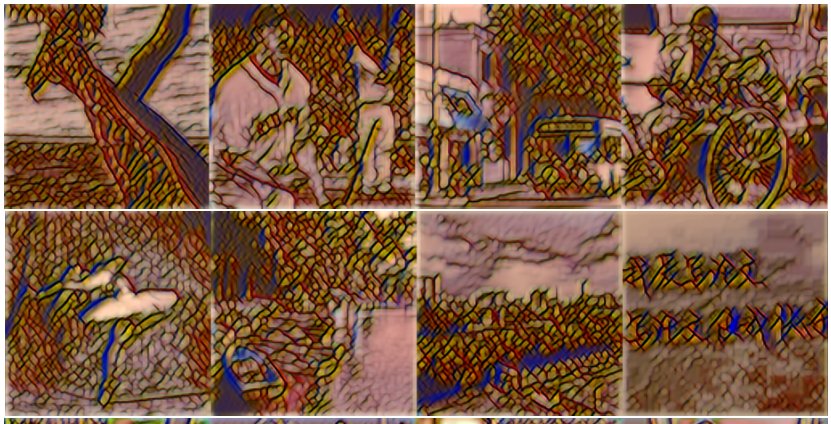
改进模型 1000

原模型 500



原模型 1000

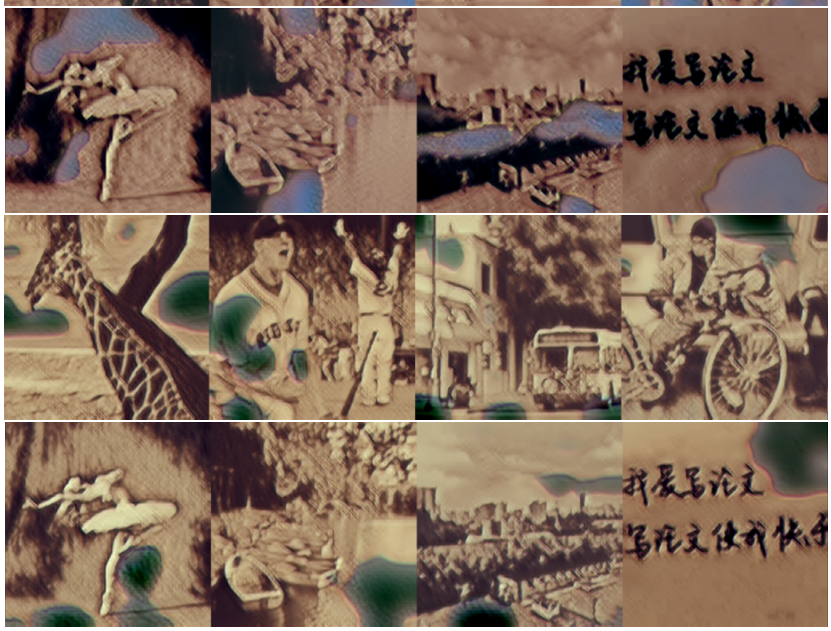
改进模型 500



改进模型 1000

原模型 500

原模型 1000



改进模型 500

改进模型 1000

原模型 500

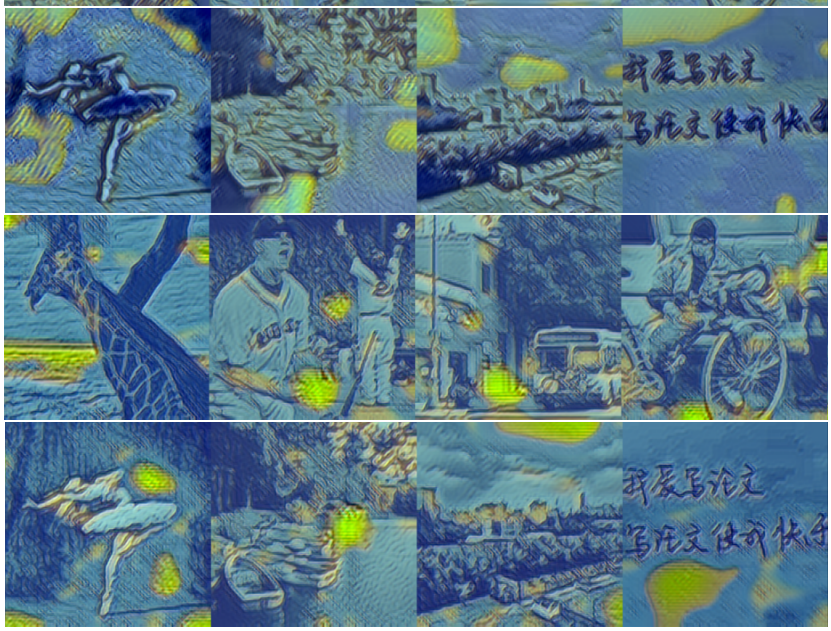
原模型 1000

改进模型 500

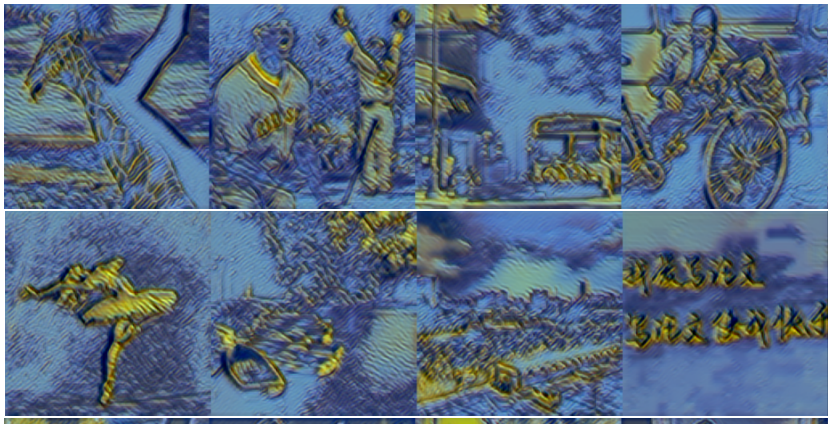
改进模型 1000

原模型 500

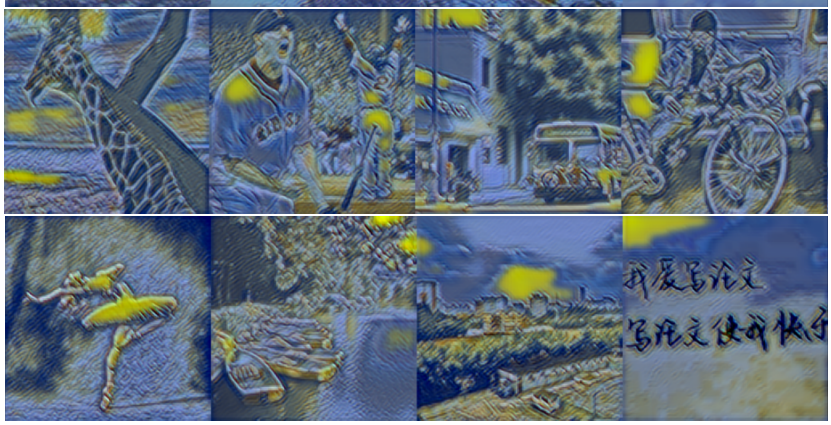
原模型 1000



改进模型 500

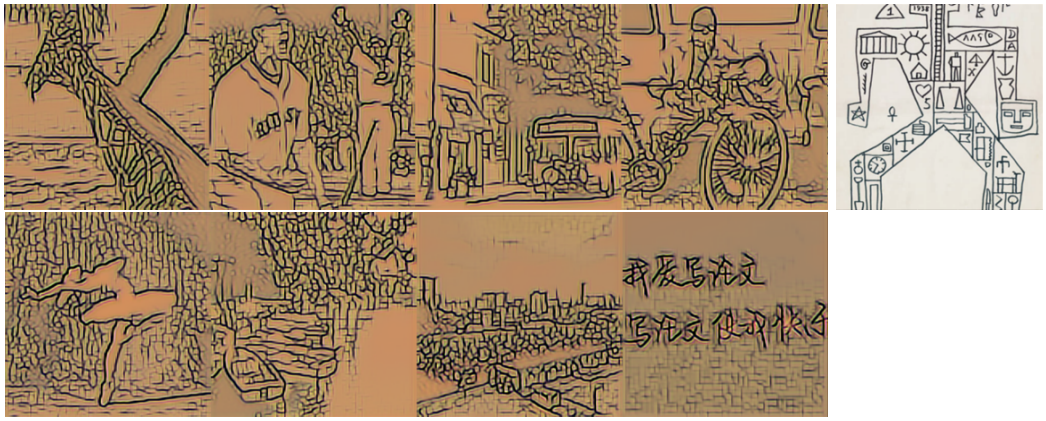


改进模型 1000



B 消融实验结果

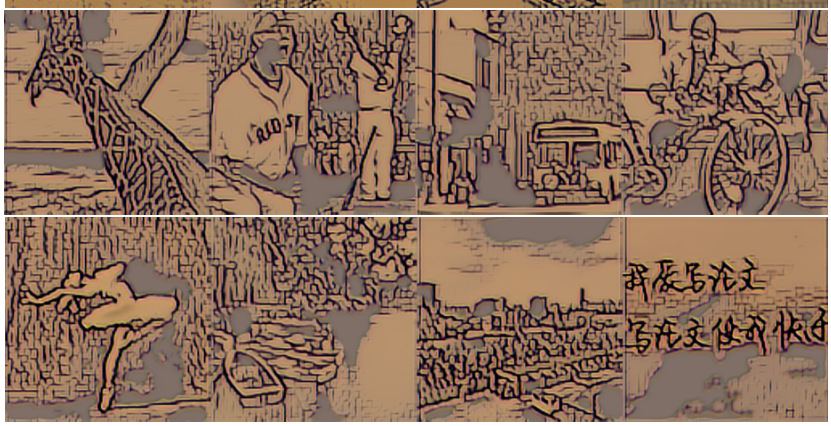
改进模型
vgg19
resnext
gram



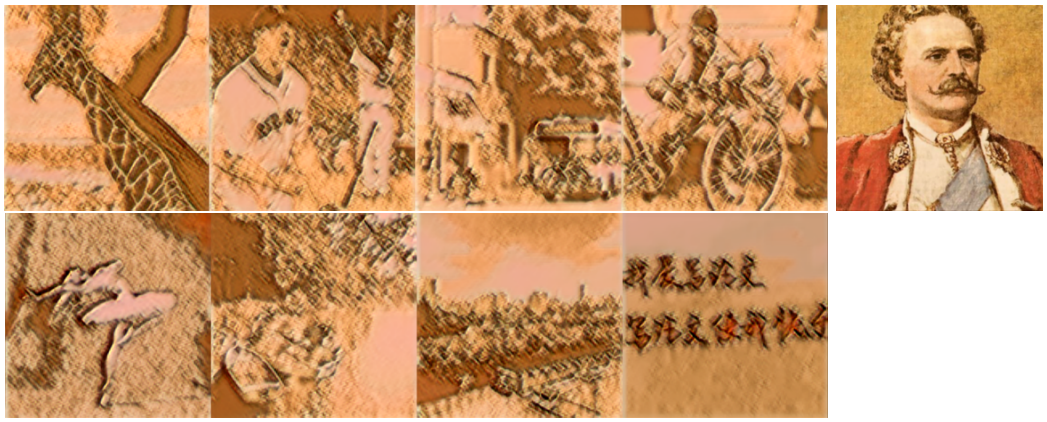
vgg16
resnext
gram

vgg19
resnet
gram

vgg19
resnext
mse



改进模型
vgg19
resnext
gram

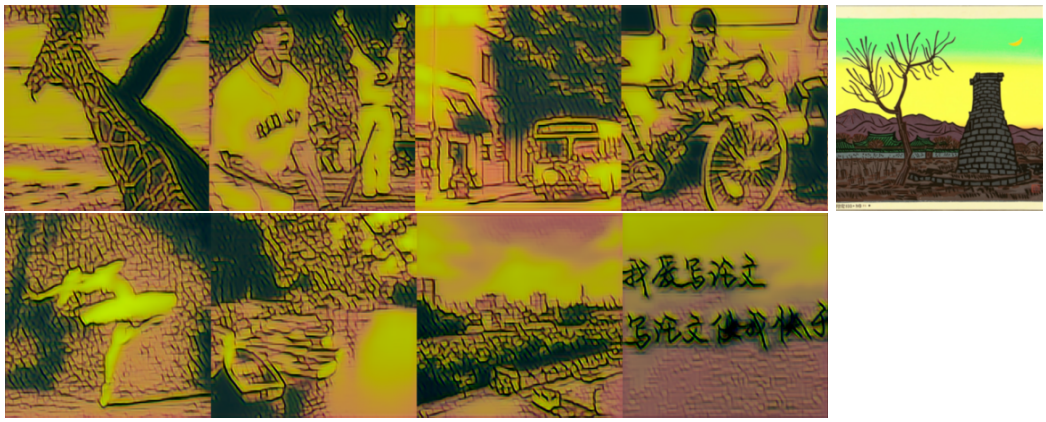


vgg16
resnext
gram

vgg19
resnet
gram

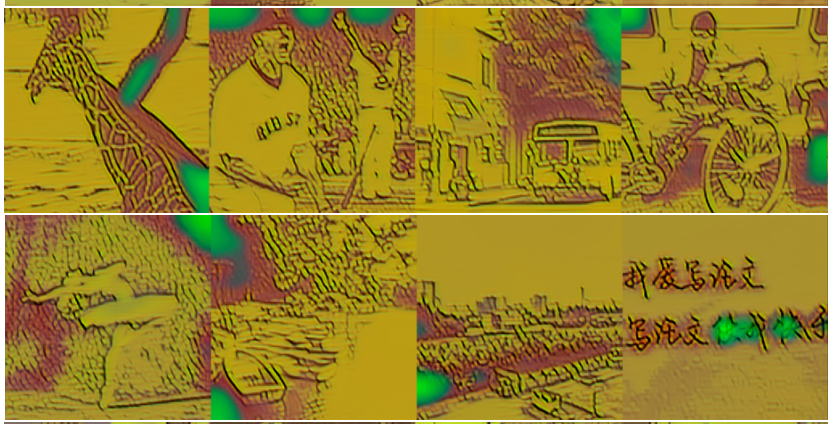
vgg19
resnext
mse

改进模型
vgg19
resnext
gram



vgg16
resnext
gram

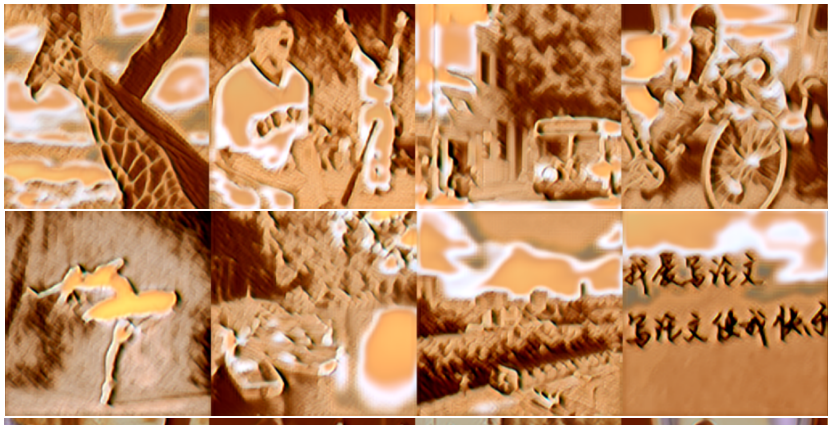
vgg19
resnet
gram



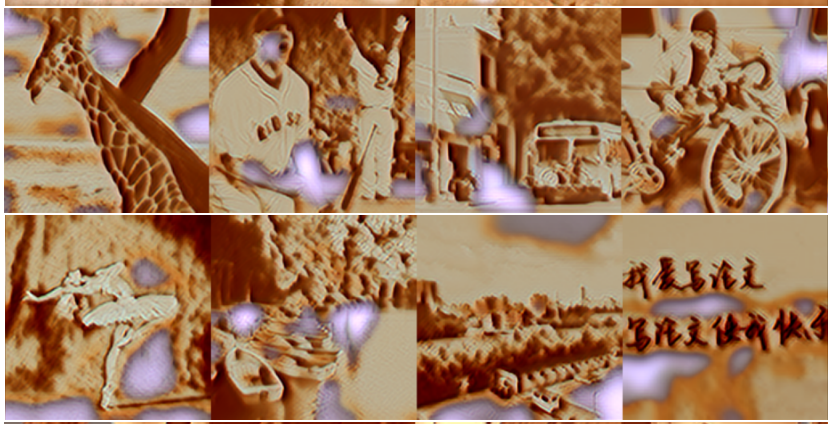
vgg19
resnext
mse

改进模型
vgg19
resnext
gram

vgg16
resnext
gram

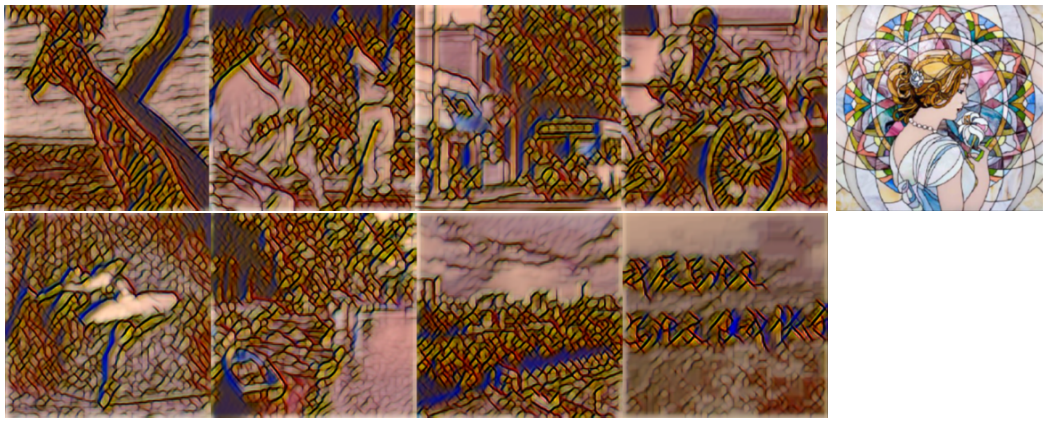


vgg19
resnet
gram



vgg19
resnext
mse

改进模型
vgg19
resnext
gram



vgg16
resnext
gram

vgg19
resnet
gram

vgg19
resnext
mse

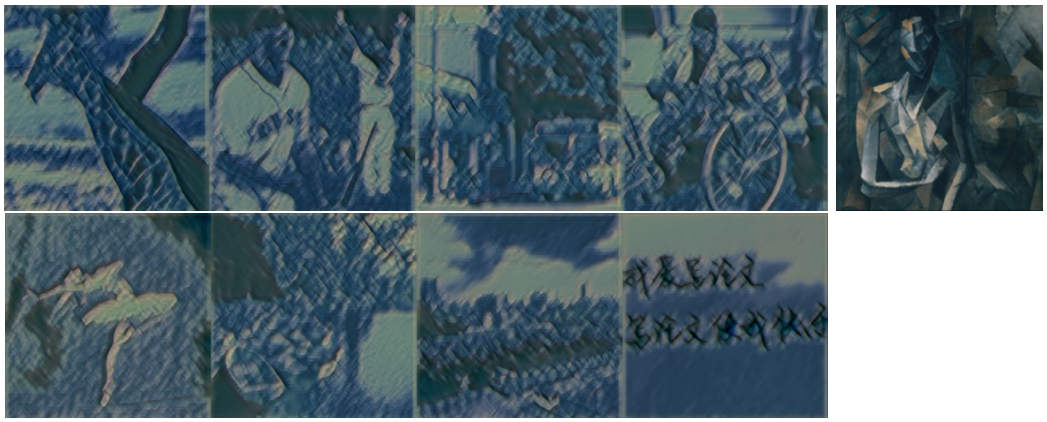
改进模型
vgg19
resnext
gram

vgg16
resnext
gram

vgg19
resnet
gram

vgg19
resnext
mse

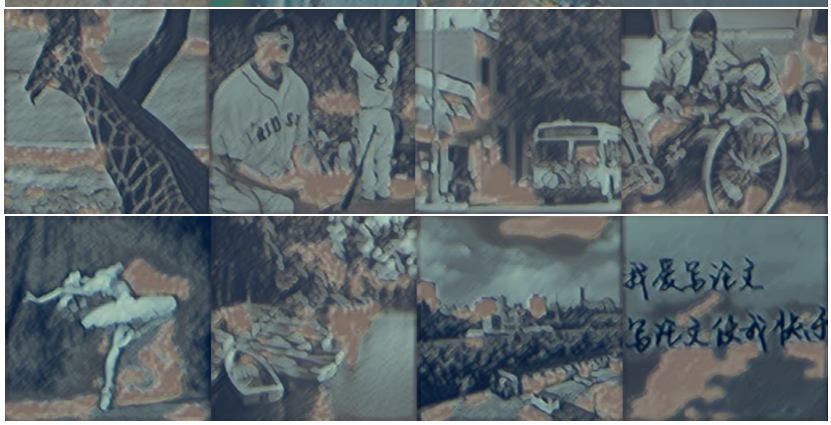
改进模型
vgg19
resnext
gram



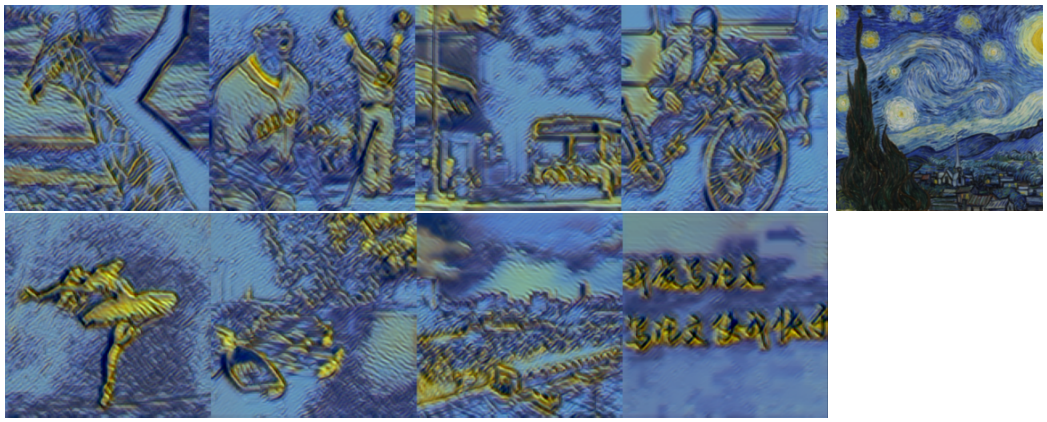
vgg16
resnext
gram

vgg19
resnet
gram

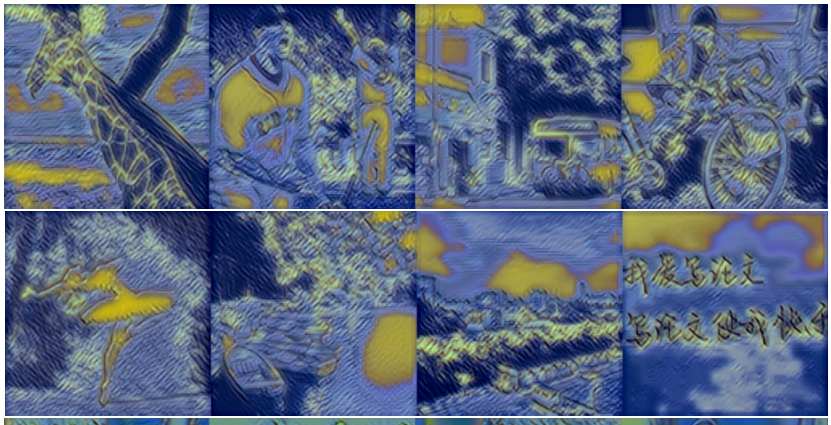
vgg19
resnext
mse



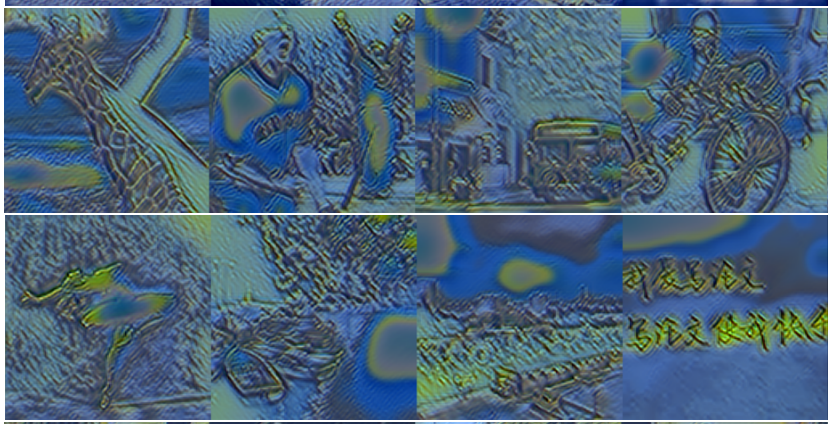
改进模型
vgg19
resnext
gram



vgg16
resnext
gram



vgg19
resnet
gram



vgg19
resnext
mse