edge-SR: Super-Resolution For The Masses (WACV 2022)

王健平

摘要

经典的图像缩放 (例如：bicubic) 可以看作是一个卷积层和一个放大滤波器。这种实现常见于几乎所有显示设备和图像处理软件。并且在过去的十年中，深度学习被引入到图像超分辨率（SR）任务中，使用多个卷积层和滤波器。这里面许多方法已经取代了原先放大任务中图像质量的基准。是否有可能在显示屏、平板电脑、笔记本电脑等边缘设备上应用深度学习架构来取代传统的上采样？一方面，边缘 AI 芯片的迅猛发展趋势显示了这方面的广阔前景，可以高效深度学习的硬件快速发展。另一方面，在图像 SR 中，只有较少的模型可以实现在极小尺寸上的边缘设备上实现实时的运行。文章探索了这个问题可能的解决方案，目的是填补传统放大方法和小型深度学习配置之间的差距。文章还提出了一组可解释性的上采样单层架构的边缘 SR(eSR)，作为从传统上采样到深度学习的过渡。当然，一层的架构无法达到大多数深度学习的质量，但在推理速度方面 eSR 可以在权衡图像质量和运行性能之间表现优异。并且文章提出的可解释性系统，可以揭示解决问题的内在策略，并指导我们未来对更大网络的改进和理解。

关键词：实时超分辨率；可解释性网络；

1 引言

基于当前边缘 AI 芯片市场的快速发展，越来越多深度学习系统能够在生活中落地和实现。这是一个巨大的市场，包括了手机、平板、显示器、甚至更小的移动端设备等等[1]。对于这些设备，通常会有图像方面的应用，例如：图像分类或对象检测，而在这些应用中输入的图像一般不会很大 (例如：256x256) 并且输出数据是低维的 (例如：标签和边框)，但有时候对于别的应用程序例如：从低分辨率图像恢复成高分辨率图像，也就是图像超分辨率 (SR) 时，就会产生大量的输入输出数据。这时，深度学习轻量化落地就会有着巨大的前景。例如，考虑在电视显示器中将图像从全高清分辨率提升到 4K 分辨率。输入层需要处理 200 万像素，输出层需要以每秒至少 24 帧的速率传送 800 万像素。有趣的是，使用小特征 (例如 2×) 进行升级是网络最容易解决的问题，通常需要较少的参数来学习，同时也是最难部署的解决方案。后者是由于显示设备具有固定的输出分辨率。对于较小的放大因子，输入图像仍然较大，与输入图像越来越小的较高放大因子相比，需要更高的输入吞吐量。因此，考虑到端设备的性能限制，图像 SR 的问题变得更加具有挑战性。

2 相关工作

复现文章中的 eSR-MAX、eSR-TR、eSR-TM、eSR-CNN 模块[2]，对比 bicubic、FSRCNN、espcn 等实时超分网络并进行评估。最后进行对网络的改进，其中包括了对网络结构的优化和轻量化。

2.1 SR 的历史

标准缩放算法，例如线性或双三次插值算法，通过应用低通滤波器在低分辨率的相邻像素之间插入零而创建的高分辨率图像[3]。现代张量处理框架 (例如 Pytorch、Tensorflow 等) 使用转置卷积层来实

现这个过程，每个输入通道都有一个滤波器。而更高级的缩放算法遵循几何原理来提高图像质量。例如，边缘定向插值使用自适应滤波器来改善边缘平滑度，或带通方法同时使用自适应上采样和滤波。后来，机器学习已经能够使用原始高分辨率图像的示例从低分辨率图像学习映射。

深度学习和卷积网络在图像分类任务中的兴起迅速带来了一系列重要的改进。这些改进中的许多都是随着图像分类网络架构的进步而来的，例如，SRCNN 中应用的 CNN、EDSR[4]中应用了 ResNets[5]、RDN 中使用的 DenseNets、RCAN 中使用了注意力、RNAN 中应用非局部注意力以及 SwinIR 中应用 swin transformer[6]。

2.2 实时 SR

为图像 SR 提出的第一个深度学习系统，即 SRCNN，使用了相对较少的参数 (60k)，成为边缘设备的合适候选。不久之后，FSRCNN 意识到，通过以低分辨率执行计算，可以显著提高质量和性能。他们提出了一个简短的配置，在 4 个卷积层的序列中使用 4k 个参数，再加上最后一个转置卷积来执行放大，从而达到小分辨率的实时性能。

在实时应用方面的下一个重大进展是 ESPCN，它使像素混淆的应用流行了起来，多路复用多个网络信道以形成更高分辨率的输出。他们提出了一种使用 20k 参数和 3 个卷积层的配置，所有计算都以低分辨率进行。FSRCNN[7]和 ESPCN[8]都为未来的图像 SR 研究打下了基础，这些研究通常以低分辨率执行计算，并使用像素混淆。后来，研究转向了规模更大、质量更好的网络。这些包含数百万个参数的大型网络，例如 EDSR(结合 ResNets 和像素混淆)，目前无法达到边缘设备上实时应用所需的吞吐量。一些所谓的轻量级网络已经被提出用于中层应用。典型的轻量级网络使用数十万个参数，仍然超出了边缘设备上实时应用程序的能力。

2.3 受到的挑战

尽管在技术上取得了可喜的进步，但边缘设备的图像 SR 挑战依然没有解决。人们可能会认为 Edge-AI 芯片会变得更快、更便宜，但标准的发展也会使问题变得更加困难 (例如 BT.2020)，因为像素更多、比特深度更高、帧速率更高。因此，AI 芯片能否成功部署图像 SR 技术并进入大规模市场，在很大程度上取决于更好的算法解决方案。

主要的挑战是如何简化网络结构，以达到与传统的非自适应缩放器相当的性能水平。经典的双三次，水平和垂直分辨率加倍，可以使用转置卷积层和使用 121 个参数的单个滤波器来实现，可以认为这是映像 SR 的最简单的网络配置。

一种可以解释的配置，即我们理解插值滤波器值的含义。我们在这里的主要任务是探索经典升级和小型深度学习系统之间的关联，以便为边缘设备中应用的当前状态提供实用解决方案。

3 边缘设备超分辨率

3.1 传统方法

图像升频和降频是指低分辨率 (LR) 图像向高分辨率 (HR) 的转换，反之亦然。这两个过程是密切相关的。将图像从 HR 降级为 LR 的最简单方法被称为池化或下采样。下采样的过程会在水平和垂直方向上均匀地丢弃像素。这种降尺度的问题是，HR 图像的高低频分量组在 LR 时可能会出现在同一个低频分量中，从而导致众所周知的问题：锯齿状失真。为了避免这个问题，经典的线性降频器首先

使用抗混叠低通滤波器去除高频，然后对图像进行降频。该过程在张量处理框架中实现，该框架具有跨卷积层，其中核或权重参数对应于低通滤波器系数。经典的线性升频过程对应于降频线性变换的转置，如图 1所示：

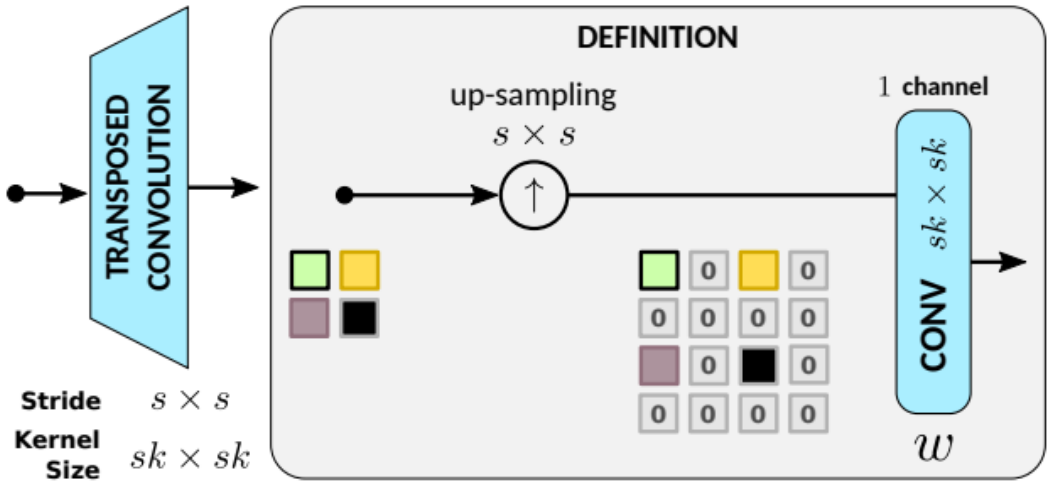

图 1: 传统方法

图 1中的上采样方式显然是低效的，因为上采样引入了许多零，当乘以滤波系数时，这些零浪费了资源。在经典升级器的实际实现中广泛使用的一种众所周知的优化是将图 1中大小为 $sk \times sk$ 的内插滤波器拆分或多路分解为 s^2，即在 LR 下工作的大小为 $k \times k$ 的高效滤波器。

然后，S^2 滤波器的输出通过像素混淆被多路复用，以获得放大的图像，如图 2 所示。设 $\widetilde{w}_i \in \mathbb{R}^{k \times k}$，其中 $i = 1, \cdots s^2$，为有效滤波器的系数。然后，可以通过将有效系数多路复用回其原始位置来恢复内插滤波器。如下所示：

$$w = Pixel - Shuffle_{s \times s}\left(\widetilde{w}_i, i = 1 \cdots, s^2\right). \tag{1}$$

在实验中，我们将比较不同的体系结构，包括双三次插值法。证明了两种实现有相同的效果和精度，我们使用图 2中的高效实现实现了放大器。

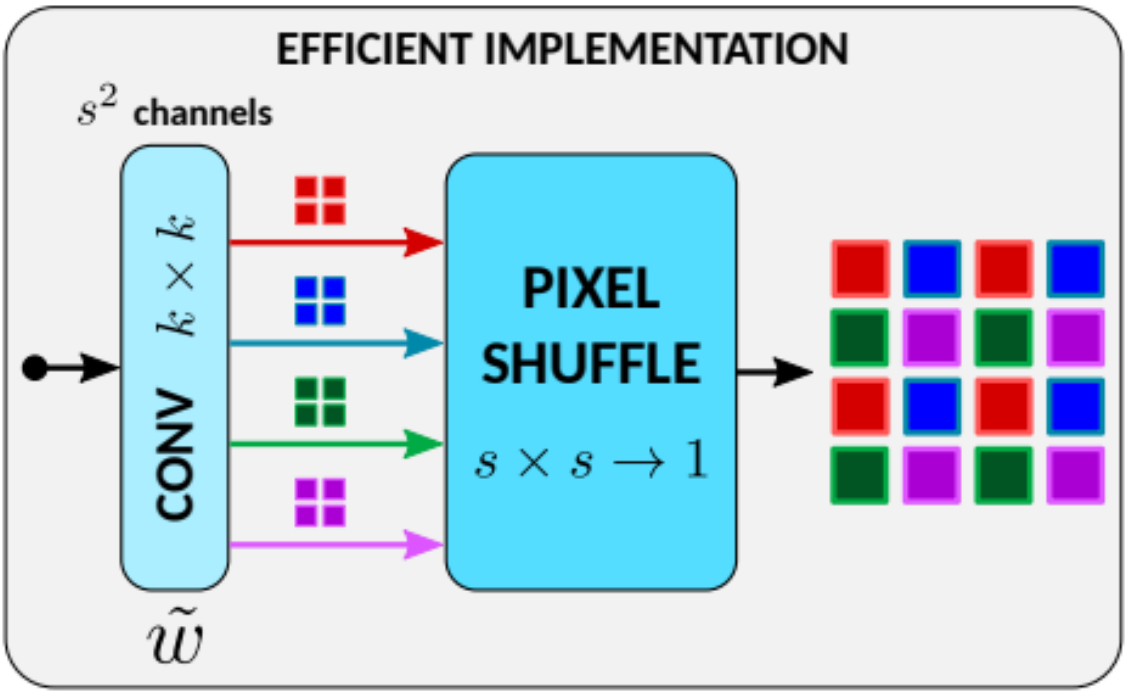

图 2: 高效的改进方法

3.2 Maxout

第一个方案是 edge-SR 最大值 (eSR-MAX)。从输出几个放大的候选对象的单个卷积层获得最大的解。如图 3所示，通过选择所有通道的最大值即可快速输出结果。

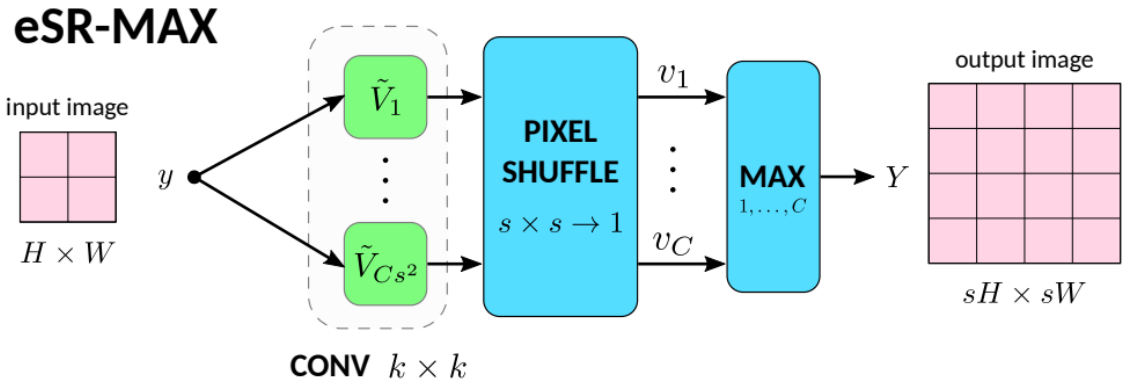

图 3: eSR-MAX

3.3 自注意力

第二个方案是遵循半经典策略的 EDGE-SR 模板匹配 (ESR-TM)，如图 4所示：

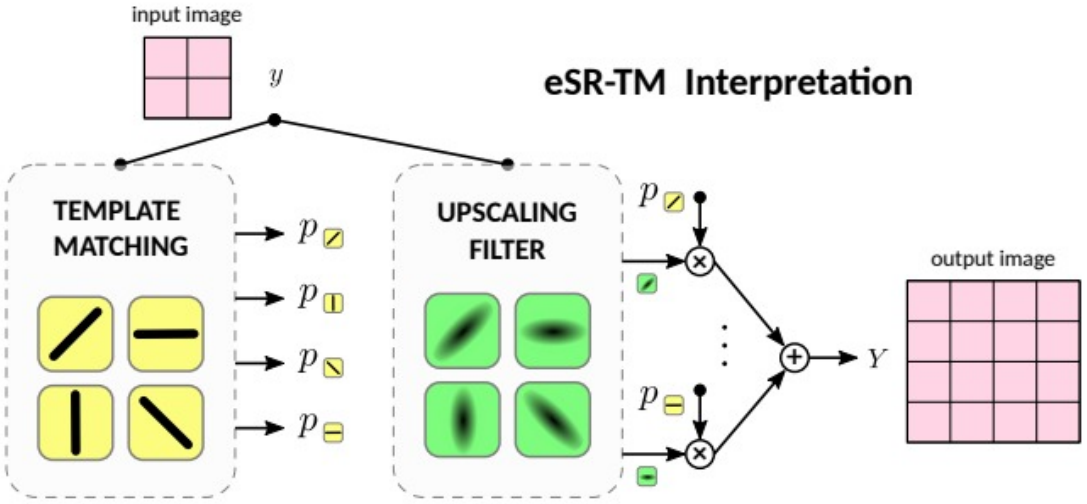

图 4: eSR-TM

图 4通过模板匹配模块计算找到符合示例学习的模板图像特征之一的概率。使用时也通过示例学习的一组不同的放大器来放大输入 LR 图像，输出是由放大后图像和模板概率的加权平均值计算的期望值。而模板匹配模块检测模式 (例如边缘方向)，给出每个模式的概率，是通过以下方式实现的：第一，使用与图案相似的匹配滤波系数，第二，对通道上的像素值进行归一化，以表示每个模板的概率。同时为每个图案计算一组 HR 图像。对输出图像的最优预测是所有模板的期望值。因此，概率被用来通过加权不同升级器的解来计算期望值，不同放大器组合在一起给出最终输出，如图 6.a 所示。eSR-TM 系统本质上是一个自注意力模块，除了最后阶段的像素混淆和所有通道的总和。这两个步骤是很重要的，因为：第一，它们将放大过程嵌入到注意力模块中，第二，它们明确使用概率来计算预期值，从而为这个模块提供了一个清晰的解释。

第三个方案是 edge-SR-Transformer（eSR-TR），它使用了现在流行的转化器自注意力模块。图 6.b 展示了模块的实现。eSR-TM 的匹配过滤器被两组查询（Q）和关键（K）过滤器取代，以估计概率。改变了 eSR-TM 的模板匹配解释，使用一个带有 Q 和 K 过滤器的，而不是一个单一的模板匹配过滤

器。这个架构的目的是为了测试自注意力对边缘 SR 的影响。

3.4 深度学习

我们认为 FSRCNN 和 ESPCN 这些用于边缘设备上图像 SR 的候选深度学习架构。图 5显示了 FSRCNN 和 ESPCN 网络架构的详细结构。

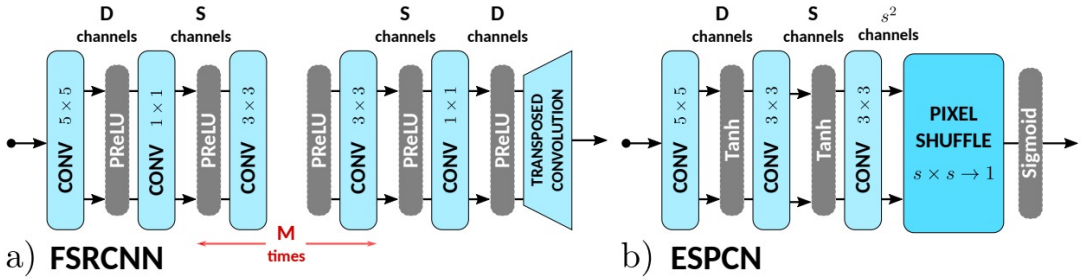

图 5: FSRCNN 和 ESPCN 架构图

相比之下，FSRCNN 使用了更多的层 (至少 5 层)，每层的通道数量比 ESPCN 小。另一个区别是放大策略，FSRCNN 使用的是分层转置卷积，ESPCN 使用的是像素混淆。根据经典的插值理论，这两种方法是等价的，如图 2所示，但实现方式可能不同。张量处理框架通常使用卷积层的梯度来实现转置卷积，基于线性变换梯度的矢量计算属性，不同的方法可能会导致性能上的差异。

最后，还提出了图 5.c 中的 edge-SR-CNN(eSR-CNN) 架构。这只是将 eSR-TM 中的单一卷积层扩展为一个与 ESPN 相同的多层结构。目的是测试和 ESPCN 与 FSRCNN 相比，是否取得了更好的结果，是否可以通过使用自我注意模块来提高规模。

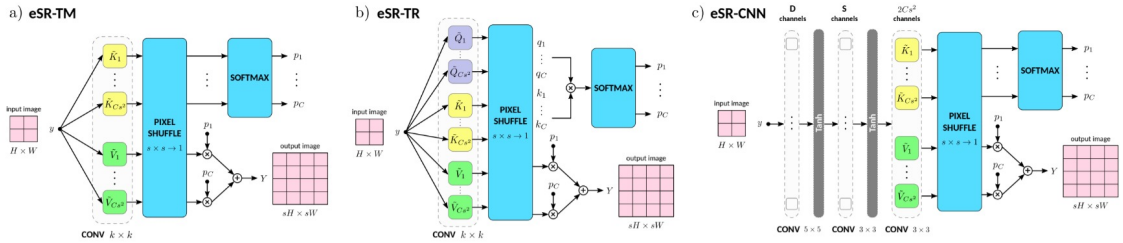

图 6: 使用自注意力的 edge-SR 架构图

4 复现细节

4.1 与已有开源代码对比

论文提供了测试代码，独立复现了训练及模块并对网络做出了一定改进。在前几层中替换了论文中原有的普通卷积层，改为 ECB 块，通过嵌入 ECB[9]块在损失了极小 fps 的情况下提升了网络性能。

ECB 块具体参数如图 7.b 所示：

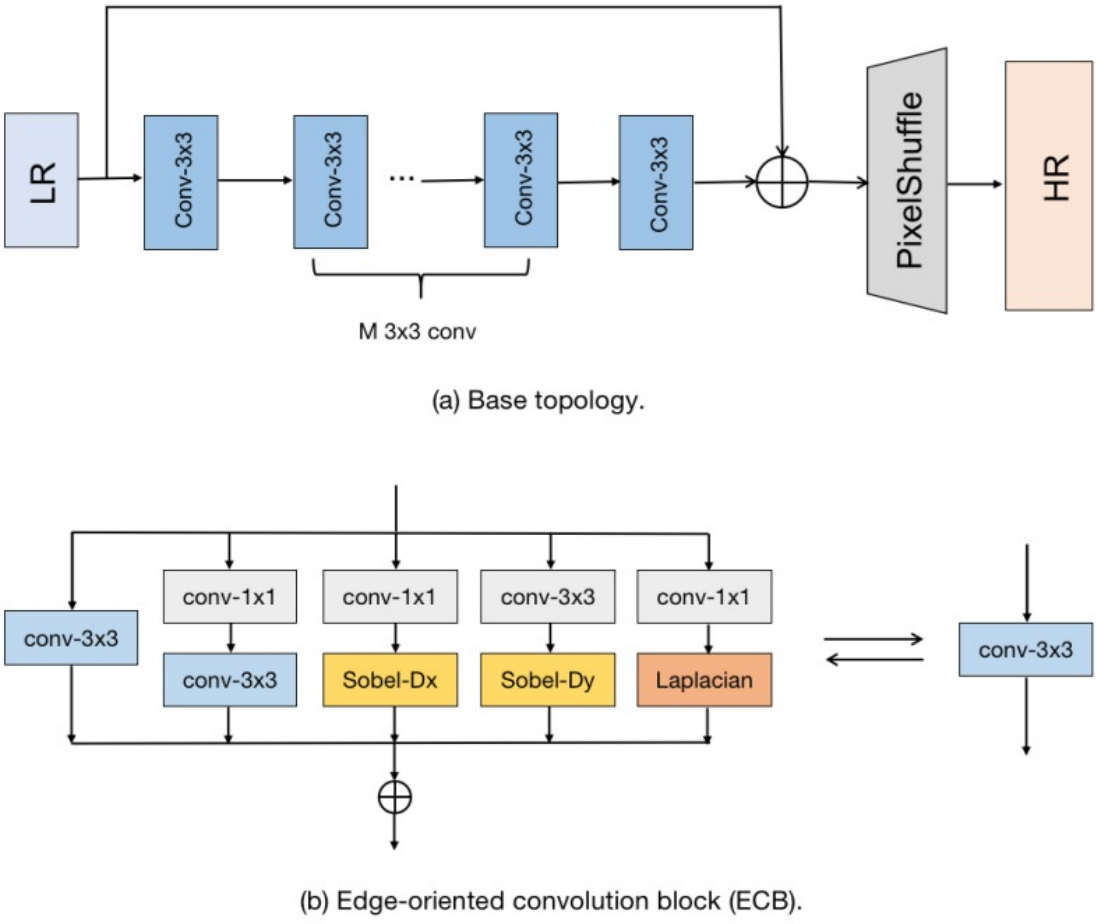

图 7: ECB 架构图

通过在训练阶段将串联卷积层更改为可修改为相同大小的 3×3 卷积层，并且这些包含了普通算子、索贝尔算子以及拉普拉斯算子的卷积层有着更好的边缘提取效果，并且通过卷积核求和的方式进行替换，获得测试时卷积核。实现了具有更好效果的实时 SR 网络。

4.2 实验环境搭建

PyTorch 1.12.0

GPU GTX3090

tqdm 4.64.0

numpy 1.32.2

h5py 3.7.0

4.3 使用说明

预处理阶段：

生成测试集：

```
python prepare.py # 训练集图片所在文件夹 \
                  --images-dir "dataset_train/eSR"  \
                  # 训练集h5文件输出文件夹 \
                  --output-path "h5_file/91-image/train_x3.h5" \
                  --scale 3  # 放大倍数
```

生成测试集：

```
python prepare.py # 测试集图片所在文件夹 \
                  --images-dir "dataset_eval/eSR" \
                  # 测试集h5文件输出文件夹 \
                  --output-path "h5_file/91-image/eval_x3.h5" \
                  --scale 3 # 放大倍数 \
                  --eval # 测试集标志
```

训练阶段：

```
python train.py --net "edgeSR_TR_ECBSR" # 选用的网络模型 \
                # h5格式训练集所在位置 \
                --train-file "h5_file/91-image/train_x3.h5" \
                # h5格式测试集所在位置 \
                --eval-file "h5_file/91-image/eval_x3.h5" \
                # 输出保存模型参数的文件夹 \
                --outputs-dir "outputs" \
                # 选填 \
                --gpu-id 0 # 使用的gpu \
                --scale 3 # 放大的倍数 \
                --lr 1e-3 # 学习率 \
                --batch-size 16 # 每次处理的批次大小 \
                --num-epochs 20  # 训练轮次 \
                --num-workers 16 # 预加载的线程个数 \
                --seed 123  # 随机数种子 \
                --group-conv 1 # 是否使用分组卷积
```

测试和评估阶段：

```
python test.py --net "edgeSR_MAX" # 模型网络 \
               --outputs-dir "outputs/edgeSR_MAX_x3" # 输出文件夹 \
               --weights-file "outputs/edgeSR_MAX_x3/best.pth" # 模型参数
               --gpu-id 0 # gpu \
               --image-dir-test "dataset/Set5" # 推理图片文件夹 \
               --image-dir-eval "dataset/Set5" # 评估图片文件夹 \
               --scale 3 # 放大的倍数
```

4.4 创新点

将 low-level 的领域特征与结构重参数思想进行了结合。在 low-level 领域，图像的梯度是非常重要的一个关注点，如何有效提升生成图像的边缘锐利度一直是业界的关注点。ECBSR 就巧妙的将一

阶梯度与二阶段梯度进行了结合，在超分任务上达到了比 DBB、RepVGG、ACNet 等重参数模块更优秀的性能。通过结合了 ECB 块和 eSR 块，将 eSR-CNN 前面提取特征的卷积串替换为 ECB 块，达到了更好的边缘优化的效果。

5 实验结果分析

训练结果如图 8所示，使用 GTX 3090 训练，学习率为 0.001，放大倍数为 3，batch-size 大小为 16 进行训练，训练集使用 91image 作为训练集和验证集。可以发现 eSR-MAX 在 psnr 表现不佳，尽管它的速度是最快的，但也是最简单的。加入了模板匹配和自注意力模块的 eSR-TM 和 eSR-CNN 表现相差不大但都比 FSCNN 表现更好 psnr 提高了 1 到 2。而改进后的 eSR-TR-ECBSR 表现最好，在训练时可以比 sota 级别的 ESPCN 高 0.5 左右的 psnr。

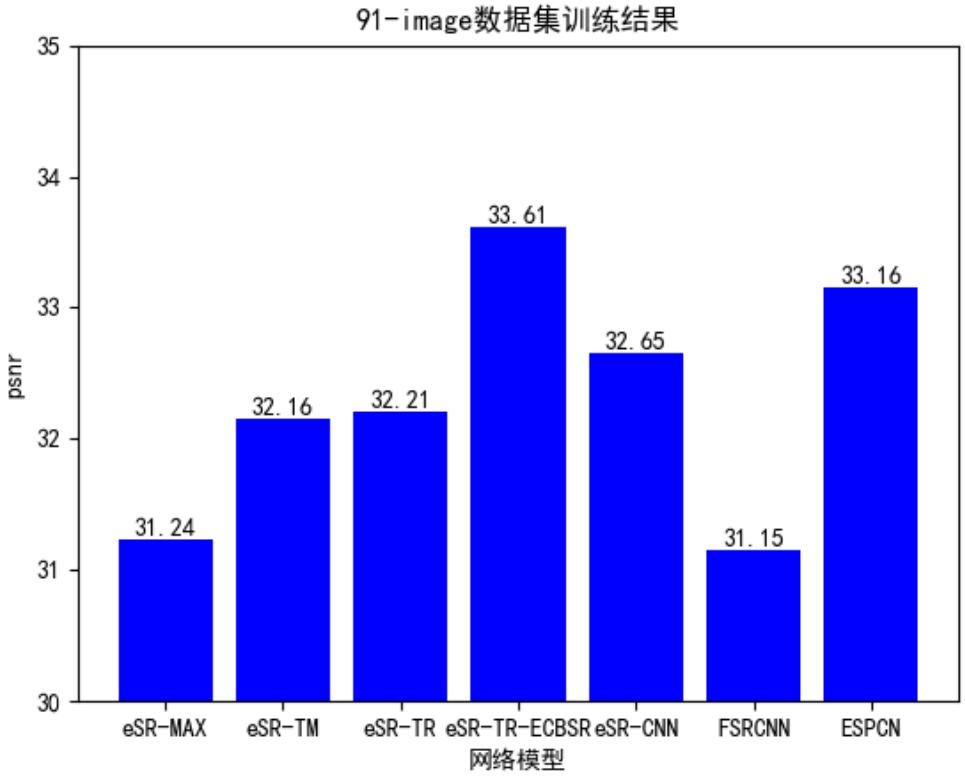

图 8: 91image 训练集结果

在各个数据集测试结果如表 1所示，放大倍数为 3。eSR-MAX 虽然图像质量上有所欠缺但是在速度上比其他网络具有优势，同时极少的参数量也是它的优点。而采用模板匹配的 eSR-TM 和自注意力机制的 eSR-TR 表现并没有特别优异，主要问题还是串联卷积层和大核卷积带来的速度降低。模板匹配给我们带来了许多思考，无论是对注意力机制的解释，还是他的性能表现。通过类似自注意力的机制带来不错的质量提升，并且因为自注意力的并行性质并不会增加太多的推理时间。我们改进后的 eSR-TR-ECBSR 网络不但在速度上比 ESPCN 快，图像质量上也是略高于 ESPCN。

在 Set5 数据集上的 butterfly 图片的推理对比如图 9所示，放大倍数为 3，截取边缘比较明显的部分。比较明显的是 eSR-MAX 和 eSR-TM 对比传统的 bicubic 有着不错的效果，但是没有带自注意力块的 eSR-TM 和 eSR-CNN 效果好。特别地，在加入了 ECBSR 模块后 eSR-TR-ECBSR 和 ESPCN 效果相当都有着不错的边缘锐化效果并且 eSR-TR-ECBSR 有着更高的 fps 性能。

Model	Set5	Set14	BSDS100	BSDS200	urban100	DIV2K
	PSNR(db)/SSIM	PSNR(db)/SSIM	PSNR(db)/SSIM/Time(ms)	PSNR(db)/SSIM/Time(ms)	PSNR(db)/SSIM/Time(ms)	PSNR(db)/SSIM/Time(ms)
eSR-MAX	30.91/0.94	27.97/0.88	27.70/0.86/79	28.14/0.87/110	25.24/0.84/249	29.67/0.90/368
eSR-TM	31.83/0.95	28.65/0.90	28.20/0.87/84	28.66/0.88/128	25.86/0.86/337	30.29/0.91/408
eSR-TR	31.87/0.95	28.62/0.90	28.22/0.87/90	28.68/0.88/131	25.90/0.86/373	30.29/0.91/472
eSR-CNN	32.33/0.95	28.98/0.90	28.42/0.87/81	28.87/0.89/150	26.24/0.87/842	30.55/0.92/1457
FSRCNN	30.89/0.93	28.14/0.88	27.76/0.86/126	28.13/0.87/216	25.58/0.85/1231	29.40/0.90/4575
ESPCN	32.84/0.96	29.34/0.90	28.65/0.88/92	29.10/0.89/146	26.72/0.88/1052	30.81/0.92/2083
eSR-TR-ECBSR(ours)	33.31/0.96	29.52/0.90	28.86/0.88/127	29.30/0.89/222	27.14/0.89/1338	31.09/0.92/1913

表 1: 不同模型在不同数据集下的评估结果，放大倍数为 3,红色为最优的结果,蓝色为次优结果。

图 9: 推理结果对比

6 总结与展望

边缘 AI 芯片的迅猛发展趋势为大规模部署高效 AI 解决方案提供了机会，但这些解决方案对图像 SR 的性能要求非常高。文章提出了 edge-SR 架构，旨在填补经典和深度学习升级之间的差距。、在一千多个不同的模型中进行了详尽的搜索，探讨了经典升级和深度学习解决方案之间的差距。文章使用单个卷积层的 edge-SR 配置显示了很有希望落地的结果，可以填补小规模图像缩放的空白。模型的简单性也使其具有可解释性，并且在文章基础上，加入了 ECBSR 模块。通过不同算子对水平垂直方向特征的提取，使用拉普拉斯滤波器提取二阶空间导数，该滤波器对于边缘信息提取更稳定，对噪声更鲁棒性。在最后的表现中也十分不错，有着不错的进步同时，计算性能也没有降低特别多。对于模型未来的发展，可以增加对参数的量化甚至二值化达到参数量进一步减小更适合未来边缘 AI 计算能接受的大小。

参考文献

[1] Jiasi, CHEN X, Ran. Deep learning with edge computing: A review[J]. IEEE, 2019, 07(8): 1655-1674.

[2] MICHELINI P N, LU Y, JIANG X. edge-SR: Super-Resolution For The Masses[J]., 2021.

[3] J.G., PROAKIS D, Manolakis. Digital Signal Processing[J]. Digital Signal Processing.Prentice Hall international editions, 2007.

[4] Bee, LIM S, SON H, et al. Enhanced deep residual networks for single image super−resolution[J]. In The

IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2017.

[5] Kaiming, HE X, ZHANG J, Shaoqing amd Ren, et al. Deep residual learning for image recognition[J]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 770-778.

[6] Jingyun, LIANG J, CAO G, et al. SwinIR: Image restoration using swin transformer[J]. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021: 1833-1844.

[7] CHAO D, CHEN C L, TANG X. Accelerating the Super-Resolution Convolutional Neural Network[C]//European Conference on Computer Vision. 2016.

[8] P A, AITKEN R, BISHOP D, et al. Accelerating the super‐resolution convolutional neural network [J]. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, 2016: 1874-1883.

[9] Xindong, ZHANG H, ZENG L, et al. Edge-oriented Convolution Block for Real-time Super Resolution on Mobile Devices[J]. ACM Multimedia, 2021: 4034-4043.