1 模型

1.1 模型简介

通常来说，视频超分需要聚合视频序列中的多个高度相关但未对齐的帧之间的信息。因此，视频超分除了基于图像超分之外，往往还需要借助时间信息。本质上来说 XLSR 是图像超分模型，但是由于其运行速度快，超分效果稳定，因此在不依赖于时间信息的情况下，也是可以胜任视频超分任务的。

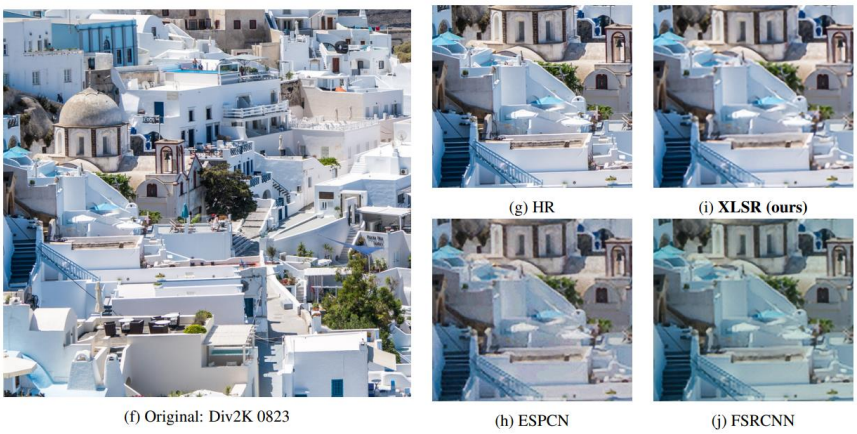


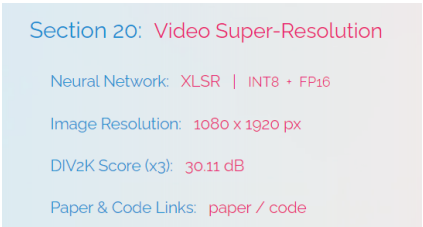
图1-1 XLSR 超分示例（from paper）

XLSR

paper：https://arxiv.org/abs/2105.10288

XLSR 的论文全名为 Extremely Lightweight Quantization Robust Real-Time Single-Image Super Resolution for Mobile，名义上获得了 2021 年 Real-Time Single Image Super Resolution Challenge 比赛的冠军。XLSR 模型将特征按通道平均拆分为 4 份，并采用 3x3 以及 1x1Conv 进行特征提取，再通过 Concat 算子对拆分特征提取后的通道特征融合，运行效率高，对 NPU 友好，可用于 1080p 视频超分场景。官方认为该模型运行效率更高，对 NPU 也更友好。

在 AI-Benchmark 的基准测试页面中，第 20 项任务视频超分采用的就是 XLSR：



从参数量上来说，XLSR 比 VDSR 参数量要少 30 多倍，但在 DIV2K 验证集上的指标要更好。作者基于 ResNext block，去除了 channel shuffle 和 skip connection。此外，避免了所有的加法和乘法算子，选择在必要时使用 Concat 算子，输入没有进行归一化处理，而是在输出添加一个 Clip ReLU 模块。

下图是论文提出的结构图：

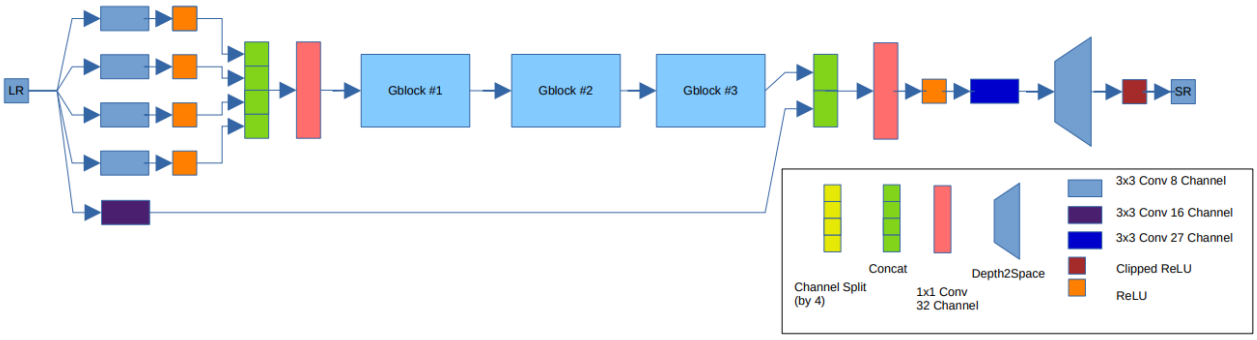


图1-2 XLSR 模型结构（from paper）

1.2 模型下载

模型 xlsr_quant.tflite 可以通过下载 AI_Benchmark_v5.0.3.apk，并使用常规的解压缩工具解压下载的 apk 文件即可，解压后模型位于 assets_v5.0.1\models\xlsr_quant.tflite

xlsr_quant.tflite 实现 3 倍超分至 1080p，详情如下表：

输入 input	uint8 [1,360,640,3]
权重	int8
输出 output	uint8 [1,1080,1920,3]

2 数据

2.1 数据集下载

这里数据集采用 DIV2K 数据集的 val 验证集

下载命令如下：

```
wget http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_valid_HR.zip --no-check-certificate
```



```
wget http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_valid_LR_bicubic_X3.zip --no-check-certificate
```

2.2 数据集简介

DIV2K 官网：https://data.vision.ee.ethz.ch/cvl/DIV2K/

DIV2K 数据集是一个高分辨率图像数据集，用于超分辨率重建算法的研究，来自于不同的场景，如城市、自然风景、动物等。每张高分辨率图像都有对应的低分辨率版本，可以用于训练和评估超分辨率重建算法。由于其高质量和多样性，DIV2K 数据集在图像超分辨率领域变得非常流行。

DIV2K 数据集有 1000 张高清图(2K 分辨率)，其中 800 张作为训练，100 张作为验证，100 张作为测试。这里我们实现的是 3 倍超分，使用的高分辨率和低分辨率图片文件夹结构如下：

高分辨率图片文件夹：

```
----- DIV2K/DIV2K_valid_HR/
  |-----0801.png
  |-----0802.png
            ...
  |-----0900.png
```

低分辨率图片文件夹：DIV2K_valid_LR_bicubic/X3/

```
----- DIV2K_valid_LR_bicubic/X3/
  |-----0801x3.png
  |-----0802x3.png
            ...
  |-----0900x3.png
```

其中 0801.png 的宽高分别是 0801x3.png 宽高的 3 倍。

3 服务器端测试

3.1 前处理

已经量化的 xlsr_quant.tflite 模型无前处理，直接输入原图 uint8 格式进行推理即可

完整服务器端测试代码见附录 test_PSNR.py

3.2 后处理

无后处理，模型本身包含 Minimum 和 Relu 以及 Mul 算子做后处理，因此输出即为 uint8 格式图片

3.3 评估指标

PSNR 全称 Peak signal-to-noise ratio，峰值信噪比，是一个表示信号最大可能功率和影响它的表示精度的破坏性噪声功率的比值的工程术语。由于许多信号都有非常宽的动态范围，峰值信噪比常用对数分贝单位来表示。

计算 PSNR 要先知道 MSE(均方误差)的计算。两个 m×n 单色图像 I 和 K，如果一个为另外一个的噪声近似，那么它们的的均方误差定义为：

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

PSNR 的定义为：

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

其中 MAXI 是图像点颜色的最大数值。一般地，针对 uint8 数据，最大像素值为 255；针对浮点型数据，最大像素值为 1。

服务器端测得的 TFlite 的 PSNR 为 29.328。

使用传统的 Nearest 插值算法与 XLSR 超分至 1080p 的对比图如下：

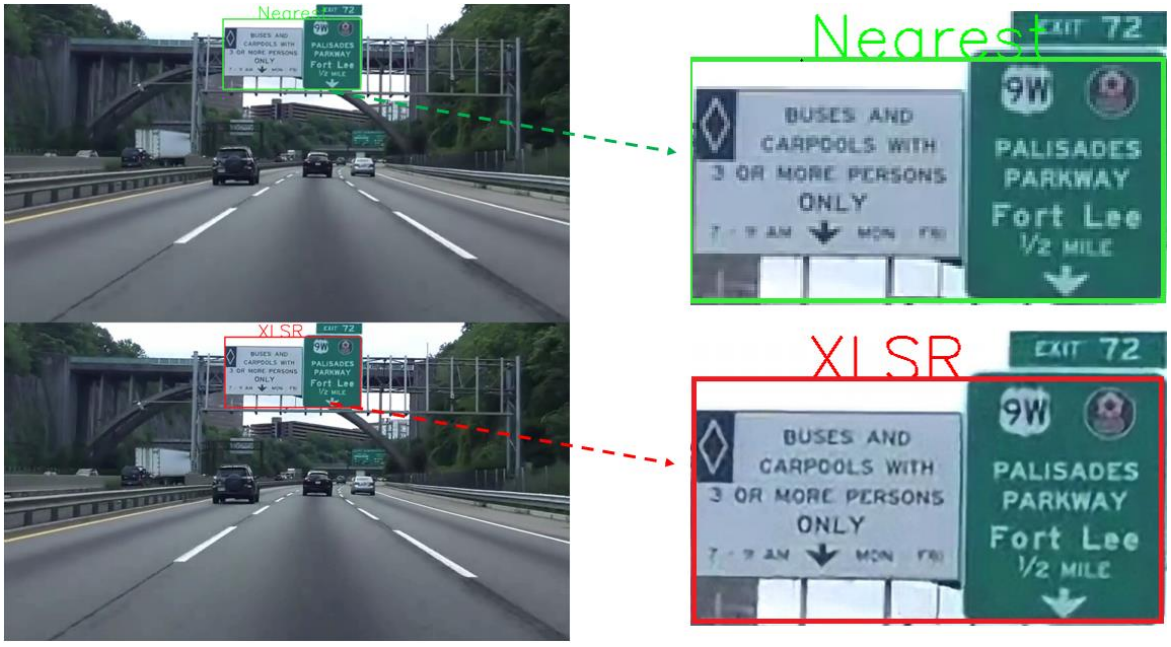


图3-1 Nearest 与 XLSR 细节对比

4 安卓端测试

注意 **本节测试环境手机 Android 版本为 A13**

4.1 离线

利用 UniAI-SDK v1.0.0 工具，将模型编译为 uir 文件，然后在手机上进行离线推理测试.

将 8bit 量化的 TFlite 模型转换为 uir 格式，转换命令如下：

```
./model_convertor_x86 -f tflite -p ./xlsr_quant.tflite -o xlsr_quant.uir
```

为获取最佳测试性能，需要使用 shell 进行手机环境调整，完整 set_env_for_perf_v4.sh 见附录。运行 set_env_for_perf_v4.sh 将 DDR、CPU 以及 NPU 频率调整到系统最高。set_env_for_perf_v4.sh 使用方式如下：

```
adb root
adb push set_env_for_perf_v4.sh /data/local/tmp
adb shell "chmod +x /data/local/tmp/set_env_for_perf_v4.sh"
adb shell "sh /data/local/tmp/set_env_for_perf_v4.sh"
```

这里可以使用我方提供的 classification_demo 进行离线推理演示，其文件结构如下：

```
-----xlsr_demo
    -----images
    -----models
    -----runtime
    -----uniai_inference
```

其中 images 文件夹中存放待测的图片数据，models 文件夹存放我们转换的 uir 模型，runtime 文件夹预先存放了所需的 runtime 依赖库，而 uniai_inference 是 demo 的可执行程序。

xlsr_demo 的使用教程如下：

将 xlsr_demo 文件夹导入手机的/data/local/tmp 目录，然后使用 adb 进入手机环境，cd 至 /data/local/tmp/xlsr_demo 目录，给 uniai_inference 赋予执行权限，使用./uniai_inference -h 查看命令行参数：

```
adb push xlsr_demo /data/local/tmp
```



```
adb shell
cd /data/local/tmp/xlsr_demo
chmod +x uniai_inference
export LD_LIBRARY_PATH=./runtime:$LD_LIBRARY_PATH
./uniai_inference -h
```

./uniai_inference -h 输出结果：

```
Usage: UniAi SDK
--model_path, -m: default: models/xlsr_quant.uir
--input_dir, -i: A directory include images, default: ./images
--input_shape, -s: input shape, default: 1,360,640,3
--backends, -b: backends, default: NPU,CPU, optional: NPU,CPU / CPU / NPU
--output_dir, -o: output `SR` image result directory, default:
```

接着使用如下命令即可进行离线推理测试：

```
./uniai_inference
```

4.1.1 离线性能

版本：T790 A13　NPU Frequence 1000Mhz	
模型名	**UniAI-SDK 离线速度 ms**
XLSR	**116.472**

4.1.2 离线准确率

模型名	服务器端 Tflite PSRN	UniAI-SDK 离线 PSNR
XLSR	**29.328**	**29.328**

4.2 在线 NNAPI

利用 benchmark 工具，直接测试 tflite 模型，可参考相关 http://isupport.unisoc.com/isu-dcc 文档.

文档编号	文档名
102728	**在 T770 上对 Tensorflow lite 模型进行 Android NNAPI 性能测试示例教程.pdf**

性能评估数据使用 Tensorflow 官方性能评估工具得到，更多细节可参考 github 教程

4.2.1 在线性能

测试命令示例：

adb shell ./data/local/tmp/benchmark_model --graph=/data/local/tmp/xlsr_quant.tflite --num_threads=1 --num_runs=50 --enable_op_profiling=true --nnapi_accelerator_name=unisoc-npu --use_nnapi=true

注意 **Android 版本 A11：** --nnapi_accelerator_name=img-nna

Android 版本 A13： --nnapi_accelerator_name=unisoc-npu

版本：T790 A13　NPU Frequence 1000Mhz	
模型名	NNAPI 在线速度 ms
XLSR	27.97

4.2.2 在线精度

模型名	服务器端 Tflite PSRN	NNAPI 在线 PSNR
XLSR	29.328	29.328

