
RealSR ncnn Vulkan



ncnn implementation of Real-World Super-Resolution via Kernel Estimation and Noise Injection super resolution.

realsr-ncnn-vulkan uses ncnn project as the universal neural network inference framework.

Download

Download Windows/Linux/MacOS Executable for Intel/AMD/Nvidia GPU

<https://github.com/nihui/realsr-ncnn-vulkan/releases>

This package includes all the binaries and models required. It is portable, so no CUDA or Caffe runtime environment is needed :)

About RealSR

Real-World Super-Resolution via Kernel Estimation and Noise Injection (CVPRW 2020)

<https://github.com/jixiaozhong/RealSR>

Xiaozhong Ji, Yun Cao, Ying Tai, Chengjie Wang, Jilin Li, and Feiyue Huang

Tencent YouTu Lab

Our solution is the **winner of CVPR NTIRE 2020 Challenge on Real-World Super-Resolution** in both tracks.

<https://arxiv.org/abs/2005.01996>

Usages

Example Command

```
1 realsr-ncnn-vulkan.exe -i input.jpg -o output.png -s 4
```

Full Usages

```

1 Usage: realsr-ncnn-vulkan -i infile -o outfile [options]...
2
3 -h                show this help
4 -v                verbose output
5 -i input-path     input image path (jpg/png/webp) or directory
6 -o output-path    output image path (jpg/png/webp) or directory
7 -s scale          upscale ratio (4, default=4)
8 -t tile-size      tile size (>=32/0=auto, default=0) can be 0,0,0
   for multi-gpu
9 -m model-path     realsr model path (default=models-DF2K_JPEG)
10 -g gpu-id         gpu device to use (-1=cpu, default=0) can be
   0,1,2 for multi-gpu
11 -j load:proc:save thread count for load/proc/save (default=1:2:2)
   can be 1:2,2,2:2 for multi-gpu
12 -x                enable tta mode
13 -f format        output image format (jpg/png/webp, default=ext/
   png)

```

- `input-path` and `output-path` accept either file path or directory path
- `scale` = scale level, 4 = upscale 4x
- `tile-size` = tile size, use smaller value to reduce GPU memory usage, default selects automatically
- `load:proc:save` = thread count for the three stages (image decoding + realsr upscaling + image encoding), using larger values may increase GPU usage and consume more GPU memory. You can tune this configuration with “4:4:4” for many small-size images, and “2:2:2” for large-size images. The default setting usually works fine for most situations. If you find that your GPU is hungry, try increasing thread count to achieve faster processing.
- `format` = the format of the image to be output, png is better supported, however webp generally yields smaller file sizes, both are losslessly encoded

If you encounter crash or error, try to upgrade your GPU driver

- Intel: <https://downloadcenter.intel.com/product/80939/Graphics-Drivers>
- AMD: <https://www.amd.com/en/support>
- NVIDIA: <https://www.nvidia.com/Download/index.aspx>

Build from Source

1. Download and setup the Vulkan SDK from <https://vulkan.lunarg.com/>
 - For Linux distributions, you can either get the essential build requirements from package manager

```
1 dnf install vulkan-headers vulkan-loader-devel
```

```
1 apt-get install libvulkan-dev
```

```
1 pacman -S vulkan-headers vulkan-icd-loader
```

2. Clone this project with all submodules

```
1 git clone https://github.com/nihui/realsr-ncnn-vulkan.git
2 cd realsr-ncnn-vulkan
3 git submodule update --init --recursive
```

3. Build with CMake

- You can pass `-DUSE_STATIC_MOLTENVK=ON` option to avoid linking the vulkan loader library on MacOS

```
1 mkdir build
2 cd build
3 cmake ../src
4 cmake --build . -j 4
```

Sample Images

Original Image



Upscale 4x with ImageMagick Lanczo4 Filter

```
1 convert origin.jpg -resize 400% output.png
```



Upscale 4x with srmd scale=4 noise=-1

```
1 srmd-ncnn-vulkan.exe -i origin.jpg -o 4x.png -s 4 -n -1
```



Upscale 4x with realsr model=DF2K scale=4 tta=1

```
1 realsr-ncnn-vulkan.exe -i origin.jpg -o output.png -s 4 -x -m models-DF2K
```




Original RealSR Project

- <https://github.com/jixiaozhong/RealSR>

Other Open-Source Code Used

- <https://github.com/Tencent/mxnet> for fast neural network inference on ALL PLATFORMS

-
- <https://github.com/webmproject/libwebp> for encoding and decoding Webp images on ALL PLATFORMS
 - <https://github.com/nothings/stb> for decoding and encoding image on Linux / MacOS
 - <https://github.com/tronkko/dirent> for listing files in directory on Windows