# User Documentation: Super resolution and artifact reduction using Swin2SR

## 1. Introduction

This documentation is aimed at the user that would like to use the Swin2SR paper results to scale and remove compression artifacts from a video. This repository contains code for a video processing pipeline using the Swin2SR model. The pipeline includes video downloading, frame extraction, image restoration, and video generation.

## 2. Requirements

The code was tested with Python 3.8.17 and Ubuntu 20.04 as well as Ubuntu 22.04.
At least FFmpeg version 4 should be installed on the system.

## 3. Installation

1) To install FFmpeg on Ubuntu use:
   ```
   sudo apt install ffmpeg
   ```
2) Clone the repository:
   ```
   git clone https://github.com/tendermonster/otcv23
   ```
3) Install required packages:
   ```
   pip install -r requirements.txt
   ```

## 4. Usage

To remove artifacts and scale the video, all you need to do is provide the URL to the video of interest in main.py class.
There are several other variables in main.py that can be adjusted:

```
# Url to the video you want to download
URL_DEMO_VIDEO="https://dlcv2023.s3.eu-north-1.amazonaws.com/demo_small.mp4"
# Path to the folder where the video and frames will be saved
PATH_MEDIA = "media/"
# Path to the folder where the frames will be saved
PATH_VIDEO_FRAMES = "media/video/"
# Path to the model
```

```
PATH_MODEL = "model_zoo/Swin2SR_CompressedSR_X4_48.pth"
# Path to the folder where the results will be saved after running the model
# Results will usually be saved to "results/swin2sr_*" directory
PATH_RESULTS = "results/swin2sr_compressed_sr_x4"
# Name of the output video
OUTPUT_VIDEO_NAME = "demo_output.mp4"
```

If all variables are set correctly, the video of interest will be scaled according to the provided arguments. For the details of the argument description, see `main_test_swin2sr.py`.

## 5. Training

The training of the model is offloaded to the KAIR project. This project contains training and testing codes for USRNet, DnCNN, FFDNet, SRMD, DPSR, MSRResNet, ESRGAN, BSRGAN, SwinIR, VRT, RVRT models.
To start, clone the project:

```
git clone https://github.com/cszn/KAIR
```

To train the model from scratch, fallow the:
https://github.com/cszn/KAIR/blob/master/docs/README_SwinIR.md.
Download the required dataset:
Training- and testing-sets can be downloaded as follows. Please separate them into training- and test-sets respectively.
To train, run the following commands. You may need to change the`dataroot_H`, `dataroot_L`, scale factor, noisel level, JPEG compression level, `G_optimizer_lr`, `G_scheduler_milestones`, etc. in the json file according to your needs.
In this case running:

```
python -m torch.distributed.launch \
    --nproc_per_node=8 \
    --master_port=1234 \
    main_train_psnr.py \
    --opt options/swinir/train_swinir_car_jpeg.json  \
    --dist True
```

You can also train the above models using `DataParallel` as follows, but it will be slower.

```
python main_train_psnr.py --opt options/swinir/train_swinir_car_jpeg.json
```

## 6. Test

To test the model using the KAIR project, use the fallowing command:

```
python main_test_swinir.py \
    --task jpeg_car \
    --jpeg 10 \
    --model_path <PATH/TO/MODEL.pth> \
    --folder_gt <input/ground_truth/test/image/folder>
```

All required datasets can be obtained from:
https://github.com/cszn/KAIR/blob/master/docs/README_SwinIR.md.

## 7. Example

see `main.py` for a complete example of video processing and artifact reduction.

Additionally, the Predictor class can be used to make inference. This class is using `cog` (Containers for machine learning). *Cog* is an open-source tool that lets you package machine learning models in a standard, production-ready container.

```
from predict import Predictor

# Instantiate the predictor
predictor = Predictor()
# Setup the predictor (load models into memory)
predictor.setup()
# Specify the input image and task
input_image = "resources/test_image.jpg"
# Make a prediction
output_image = predictor.predict(input_image, task="compressed_sr")
# Print the path to the output image
print("Output Image:", output_image)
```

## 8. Important method descriptions

The most important methods to get started are described here. For further documentation, refer to the corresponding code file of interest.

`utils/util_processing.py.`:

This file contains all method needed for processing the video

```
def plot_all(images, axis="off", figsize=(16, 8)):
"""Plot all images in a list.

Args:
    images (list): A list of images to plot.
    axis (str, optional): The plot axis. Defaults to "off".
    figsize (tuple, optional): The figure size. Defaults to (16, 8).

Returns:
    None

Examples:
    # Plot a list of images
    images = [image1, image2, image3]
    plot_all(images)

    # Plot a list of images with customized settings
    plot_all(images, axis="on", figsize=(10, 5))"""
```

```
def load_img(filename, debug=False, norm=True, resize=None):
"""
Load an image from a file.

Args:
    filename (str): The path and filename of the image to be loaded.
    debug (bool, optional): If True, print debug information
    about the loaded image. Defaults to False.
    norm (bool, optional): If True, normalize the image values to the range
    [0, 1]. Defaults to True.
    resize (tuple, optional): If provided, resize the image to the
    specified dimensions(width, height). Defaults to None.

Returns:
    numpy.ndarray: The loaded image as a numpy array, with shape
    (height, width, channels).

Notes:
    - The image is loaded using the OpenCV library (cv2)
    in the BGR color space.
    - If norm=True, the image values are normalized to the range [0, 1].
    - If resize is specified, the image is resized to the specified
    dimensions using OpenCV's resize function."""
def video_to_frames(path_video: str, path_frames: str):
"""The video provided is sliced into frames and saved to the path provided.

Args:
    path_video (str): path of the video file
    path_frames (str): path to the frames directory"""
def download_file(url: str, savepath: str) -> str:
"""Download a file from a url and save it to a path.

Args:
    url (str): url to dowonload a file from
    savepath (str): path to save the file to

Returns:
    str: full paths the file was saved"""
```

```
def frames_to_video(frames: str, path_video: str):
"""This method relies on the ffmpeg and codec available on your system.
Please update this method to meet your system requirements if needed.

path_video: str
    path of the video file
frames: str
    path to the frames"""
```

`predict.py` → class Predictor(BasePredictor). :

```
"""A class for making predictions using a pre-trained model.

Inherits from the BasePredictor class.

Attributes:
    models (dict): Dictionary of pre-trained models for different tasks.
    device (str): Device (e.g., "cuda:0") to run the models on.

Methods:
    setup(): Load the models into memory for efficient prediction.
    predict(image: Path, task: str) -> Path: Run a prediction on the
    specified image for the given task."""
```

`models/network_swin2sr.py`.:
This file defines the swin2sr model. It contains the Swin architecture and other parts of the code that were used in the paper [1]. Please refer to the file itself for documentation.
`main_test_swin2sr.py`.:
This is the file that is mainly used to run the inference for the video processing. Please refer to the file itself for documentation.

## 9. License

This project is licensed under the MIT License.

## References

[1] Marcos V Conde, Ui-Jin Choi, Maxime Burchi, and Radu Timofte. “Swin2SR: Swinv2 transformer for compressed image super-resolution and restoration”. In: *arXiv preprint arXiv:2209.11345* (2022).