Warlock-Studio

Technical Documentation and User Guide

Software Version: 4.3

Iván Eduardo Chavez Ayub
@Ivan-Ayub97

November 9, 2025

0. 1

Contents

1. 1

Introduction to Warlock-Studio

Welcome to Warlock-Studio — an AI-powered suite for digital media enhancement. It provides advanced tools for super-resolution, artifact removal, and frame generation through an intuitive interface that delivers professional-quality results with minimal effort.

2. 1

Quick Start Guide

Accelerated Media Enhancement Procedure

Follow these 5 steps to process your first media file in less than a minute.

1. **Load Files:** Click the **"SELECT FILES"** button and choose one or more image or video files for processing.
2. **AI Model Selection:** In the **"AI model"** dropdown menu, select an inference model.
   - For photorealistic images, `BSRGANx4` is recommended for its ability to reconstruct fine textures.
   - For animation or illustrated content, `RealESR_Animex4` is the optimal choice for preserving sharp edges.
   - For video sequences, `RealESR_Gx4` offers an excellent balance between computational performance and perceptual quality.
3. **Adjust Input Resolution:** For a quick first pass, set **"Input resolution"** to 75%. This significantly reduces the computational load with an often imperceptible loss in final quality.
4. **Verify VRAM Limit:** Ensure the value in **"GPU VRAM (GB)"** is equal to or less than your graphics card's dedicated video memory. An initial value of 4 GB is a safe setting for most modern hardware.
5. **Start Processing:** Click **"Make Magic"** to initiate the processing pipeline. The output files will be generated in the specified directory or, by default, in the same location as the source files.

3. 1

Installation and System Architecture

3.1 Installation Process

Warlock-Studio uses a self-contained offline installer, simplifying deployment.

1. **Obtaining the Executable:** Download the 'Warlock-Studio-Setup.exe' file from the official repositories on GitHub or SourceForge.
2. **Run with Elevated Privileges:** Right-click the installer and select "Run as administrator." This step is crucial to ensure the application has the necessary permissions to interact with low-level GPU drivers.
3. **Installation Wizard:** Follow the on-screen instructions. The inclusion of all AI models and dependencies eliminates the need for additional downloads during this process.
4. **Launch the Application:** Once the installation is complete, Warlock-Studio can be launched from the Start Menu or the desktop shortcut.

3.2 System Requirements

Component	Technical Specification
Operating System	Windows 11 or Windows 10 (64-bit architecture required).
RAM	8 GB (minimum), 16 GB (recommended for high-resolution video processing).
Graphics Card (GPU)	**Mandatory Requirement:** GPU with support for the **DirectX 12** API. **NVIDIA (for CUDA):** Maxwell architecture (GTX 900 Series) or newer. Studio drivers are recommended for stability. **AMD/Intel (for DirectML):** Any modern GPU with updated drivers that support DirectX 12 Feature Level 12.0+. **4+ GB of VRAM** is recommended to avoid memory bottlenecks.
Storage	2 GB of free disk space. Using a Solid State Drive (SSD) drastically improves I/O performance during video sequence processing.

Table 1: Hardware and software specifications for optimal performance of Warlock-Studio.

3.3 File Structure and Dependencies

Warlock-Studio operates as a fully self-contained environment. All critical assets, dependencies, and user data are automatically managed by the application, requiring no manual configuration from the user.

- **Core Assets:** The executables `ffmpeg.exe` (for video encoding/decoding) and `exiftool.exe` (for metadata preservation) are stored in the `Assets` directory. Their paths are dynamically resolved through the `find_by_relative_path` function, which accurately locates resources in both development and bundled environments (using the `_MEIPASS` directory created by PyInstaller).
- **AI Models:** All AI inference models are provided in interoperable `.onnx` format and stored within the `AI-onnx` directory. This ensures cross-platform compatibility and seamless hardware acceleration through ONNX Runtime.
- **User Preferences:** The configuration file `USER_PREFERENCE_PATH` is automatically generated in the user's **Documents** folder (`os_path_expanduser('~')`). It stores GUI layout, last-used settings, and model selections between sessions.

- **Diagnostic Logs:** Log files (`MAIN_LOG_FILENAME` and `ERROR_LOG_FILENAME`) are created in a dedicated subfolder inside **Documents**. These serve as vital diagnostic resources, recording operational events, system validation reports, and runtime exceptions for troubleshooting.

4. 1

Detailed Analysis of Inference Models

Selecting the appropriate AI inference model is the most critical decision in Warlock-Studio's enhancement pipeline. It directly determines the balance between visual quality, computational efficiency, and hardware resource usage. Each model implements distinct neural architectures and loss functions, optimized for different types of input data and output fidelity targets.
The following section provides a comprehensive technical breakdown and comparative analysis to support informed model selection.

4.1 Model Comparison Matrix

Model	Primary Function	Scale	VRAM (GB)	Use Case and Technical Considerations
		Denoising Models		
`IRCNN_Mx1`	Denoise	x1	4.0	Moderate-level noise reduction. Ideal for suppressing JPEG compression artifacts and low-ISO sensor noise in old photographs.
`IRCNN_Lx1`	Denoise	x1	4.0	Intensive noise reduction algorithm. Optimal for severely degraded images with pronounced luminance and chrominance noise.
		High-Fidelity Upscaling Models (Computationally Intensive)		
`BSRGANx4`	Upscale	x4	0.6	Generative Adversarial Network optimized for realistic texture synthesis. It is the model of choice for portraits, nature photography, and where the preservation of fine details is paramount.
`BSRGANx2`	Upscale	x2	0.7	A variant with a reduced scaling factor. It offers a perceptual quality similar to the x4 version but with lower computational cost, ideal for moderate upscaling needs.
`RealESRGANx4`	Upscale	x4	0.6	A robust general-purpose model. It excels in reconstructing a wide variety of content, including textures, landscapes, and architectural elements.

Model	Primary Function	Scale	VRAM (GB)	Use Case and Technical Considerations
`RealESRNetx4`	Upscale	x4	2.2	An alternative to RealESRGAN, often producing results with fewer "hallucinatory" artifacts. It can offer a better balance between speed and fidelity on certain GPU architectures.
				High-Speed Upscaling Models (Lightweight)
`RealESR_Gx4`	Upscale	x4	2.2	A lightweight and fast model, optimized for real-time or near-real-time video processing. It offers an excellent compromise between performance and visual quality.
`RealESR_Animex4`	Upscale	x4	2.2	Specialized for non-photorealistic content. It preserves sharp lines and flat colors, avoiding the smoothing artifacts common in models trained on natural data.
				Facial Restoration Models
`GFPGAN`	Restore	x1	1.8	Generative Adversarial Network with a facial prior. It not only upscales but also reconstructs and enhances damaged or low-resolution facial features in photographs.
				Frame Interpolation Models (Video Only)
`RIFE`	Interpolate	N/A	~1.5	High-quality algorithm for motion interpolation. It generates intermediate frames to increase the fluidity of a video (e.g., from 30 to 60 FPS).
`RIFE_Lite`	Interpolate	N/A	~1.2	An optimized variant of RIFE, designed for GPUs with limited VRAM. It offers faster processing at the cost of a slight reduction in interpolation accuracy.

Table 2: Technical guide for the selection of AI models. VRAM values are base estimates derived from the `VRAM_model_usage` dictionary in the source code and may vary.

5. 1

Performance Optimization and Best Practices

5.1 Critical Performance Parameters

- **Input Resolution %:** This is the most influential parameter on processing speed. A value between **50% and 75%** implements an initial downsampling before upscaling, drastically reducing the computational load with minimal impact on the final perceptual quality.
- **GPU VRAM Limiter (GB):** Defines the video memory budget. This value (from

`selected_VRAM_limiter.get()`) is used in a precise formula to calculate the internal `tiles_resolution` parameter:

```
vram_factor = VRAM_model_usage[model] * VRAM_GB_input
tiles_resolution = int(vram_factor * 100)
```

This `tiles_resolution` (e.g., 880px for a 4 GB card using `RealESR_Gx4`) is then used as the maximum tile dimension for image processing, ensuring full stability and preventing **Out Of Memory (OOM)** errors during inference.

- **AI Multithreading:** (Video only) Allows for the parallel processing of multiple frames via a `ThreadPool` . It significantly increases performance on systems with multi-core CPUs, but at the cost of higher VRAM and CPU consumption.
- **AI Blending:** Mitigates visual artifacts. This value (from `selected_blending_factor`) is passed to the `blend_images_and_save` function, which uses OpenCV's `addWeighted` to perform an alpha blend between the original upscaled image and the AI-processed image.
- **System Validation:** Before starting, the `validate_system_requirements` function automatically checks for the presence of `ffmpeg.exe` , available disk space, and available system RAM (using the `psutil` library).

5.2 Tips for Maximum Quality Results

Expert Recommendations for Optimal Output

Maximizing Visual Fidelity

To achieve the highest possible perceptual quality, use the `BSRGANx4` or `RealESRGANx4` models with an **Input Resolution** of **100%**. Although computationally demanding, this configuration ensures maximum reconstruction of micro-textures and reduces information loss, especially on portrait or landscape imagery.

Workflow for Video Restoration

A robust two-stage restoration pipeline for archival or degraded footage involves:

1. **Stage 1 — Denoising:** Apply a denoising model such as `IRCNN\_Mx1` to clean sensor noise and compression artifacts.
2. **Stage 2 — Upscaling:** Use an upscaling model such as `RealESR\_Gx4` on the cleaned output to reconstruct fine detail while maintaining temporal coherence.

Impact of SSD Storage

A Solid State Drive (SSD) significantly improves throughput during video restoration. The extraction (`extract\_video\_frames`), writing (`save\_frames\_on\_disk`), and reassembly (`video\_encoding`) of thousands of frames represent the main I/O bottlenecks; SSDs mitigate these delays and improve system responsiveness.

Frame Persistence for Experimentation

For testing different encoding settings (e.g., codecs or bitrates), enable the **"Keep frames"** option (`selected\_keep\_frames = True`). This preserves the processed frames on disk, allowing video re-encoding without repeating the AI inference phase — saving time and extending the experimental workflow.

6. 1

Diagnostics and Troubleshooting

⚠ Primary Cause of Errors

The **#1 cause** of processing failures is **non-standard characters** in file paths and filenames. Avoid using: `'`, `"`, `@`, `#`, `$`, `%`, `&`, `*`, `[`, `]`, `?`, `etc.`.

⃠ Error: "FFmpeg encoding failed..."

Diagnosis: The `video_encoding` function, which uses `subprocess_run`, has failed. The code specifically checks for several causes:

- **"Invalid argument"**: Most commonly caused by special characters in file paths.
- **"Unknown encoder"**: The selected video codec (e.g., `hevc_nvenc`) is not supported by your FFmpeg build or hardware.
- **"Device or resource busy"**: Your GPU's hardware encoder is being used by another application (e.g., OBS, ShadowPlay).

Solution: Rename files to remove special characters. For encoder errors, select a different codec (e.g., `x264` or `x265`).

🚀 Error: "Failed to load model" or Execution Provider Failure

Diagnosis: Failure in the initialization of the selected hardware backend, caught within the `create_onnx_session` function. **Solution:** The system is designed to automatically fall back to a functional provider in the hierarchy (CUDA -> DirectML -> CPU). To resolve the root cause:

1. **For CUDA errors:** Verify the installation of the latest NVIDIA drivers (Game Ready or Studio).
2. **For DirectML errors:** Ensure that the Windows operating system is fully updated and that you have the latest drivers for your GPU (NVIDIA, AMD, or Intel).

🧠 Error: "out of memory" (OOM) or "allocation" failure

Diagnosis: The GPU's VRAM has been exhausted. This error is caught within the `upscale_video_frames_async` function. **Solution:**

1. Lower the **VRAM Limiter** to a value equal to or less than your GPU's physical VRAM.
2. Decrease the **Input Resolution %** to 75% or less.
3. **Automatic Recovery:** The code will attempt an automatic recovery. It progressively reduces the processing tile size (by lowering the `AI_instance.max_resolution` variable) and retries the failed frame.

Error: "cannot convert float NaN to integer"

Diagnosis: This specific error is caught in the main `upscale_orchestrator`. It indicates a GPU driver timeout (TDR - Timeout Detection and Recovery), often caused by hardware overload or overheating, which returns a "Not a Number" (NaN) value instead of pixel data. **Solution:** Restart the process **without deleting the generated frames folder**. The application will detect the existing frames and resume the task from the point of failure.

7. 1

Software Architecture Analysis

7.1 Inference Engine and Hardware Abstraction

Warlock-Studio is built upon a multi-layered inference engine powered by **ONNX Runtime**. The `create_onnx_session` function abstracts the underlying hardware and prioritizes Execution Providers to maximize performance:

1. **CUDA (`CUDAExecutionProvider`):** The highest performance option, leveraging the parallel computing architecture of NVIDIA GPUs.
2. **DirectML (`DmlExecutionProvider`):** If CUDA is not available, the system falls back to DirectML. This Microsoft API translates neural network operations into native **DirectX 12** calls, ensuring broad hardware compatibility (NVIDIA, AMD, Intel).
3. **CPU (`CPUExecutionProvider`):** This is the last resort provider. If no GPU acceleration is viable, the application runs the model on the CPU, guaranteeing universal functionality at the cost of significantly lower performance.

7.2 Dynamic Tiling and Memory Management

To process high-resolution media, the `AI_upscale_with_tilling` function is invoked. It programmatically subdivides a large frame into smaller "tiles" based on the `max_resolution` variable. This `max_resolution` (confusingly named `tiles_resolution` in the GUI logic) is the pixel dimension calculated directly from the user's **VRAM Limiter** input, ensuring that no single tile exceeds the GPU's memory budget.

7.3 Resume and Checkpointing Functionality

If a video process is interrupted, the frames already processed and written to disk are preserved. Upon restarting the same task, the `check_video_upscaling_resume` function detects these partial files by checking for their existence. It then returns a list of only the *original* frames that still need processing, allowing the `upscale_video` orchestrator to resume the task from the point of failure.

7.4 Asynchronous Frame Writing

During video upscaling, the main processing loop in `upscale_video_frames_async` does not write to disk directly. Instead, it batches processed frames in memory. Once a batch is ready (defaulting to `MULTIPLE_FRAMES_TO_SAVE = 8`), it calls `save_frames_on_disk`, which spawns a new `Thread` from the 'threading' module to run `save_multiple_upscaled_frame_async`. This decouples the GPU-bound inference from the I/O-bound disk writing, maximizing GPU utilization.

8. 1

Processing Pipeline (Diagram)

The following diagram illustrates the high-level processing pipeline.

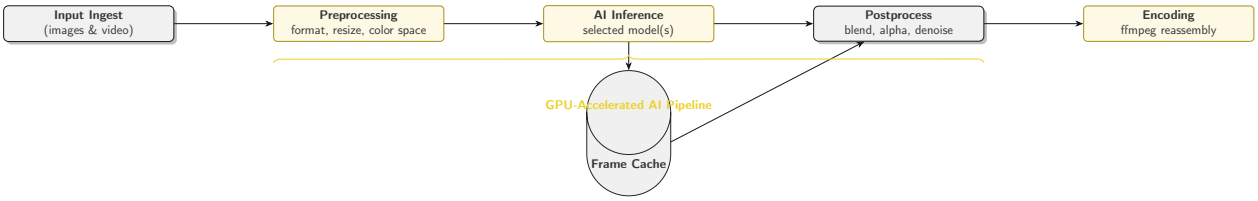

Figure 1: High-level processing pipeline (ingest → inference → encode).

9. 1

Software Architecture (Diagram)

Component-level architecture showing logical components and their relationship to hardware.

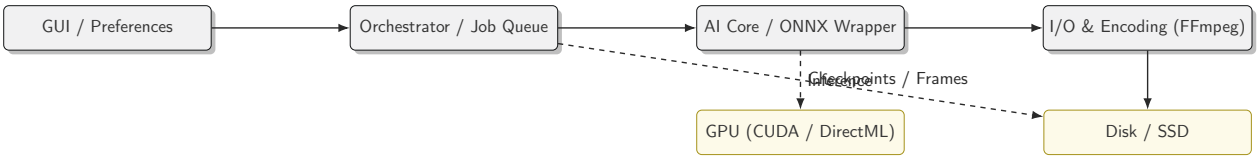

Figure 2: Component-level architecture (logical components and hardware).

10. 1

Glossary

ONNX Runtime
: (Open Neural Network Exchange) A high-performance, cross-platform inference engine for AI models. It allows Warlock-Studio to run models in a hardware-agnostic manner.

CUDA
: (Compute Unified Device Architecture) A parallel computing platform and programming model API created by NVIDIA. It enables general-purpose acceleration on NVIDIA GPUs.

DirectML
: (Direct Machine Learning) A low-level Microsoft API that uses DirectX 12 to provide GPU-accelerated AI on a wide range of DX12-compatible hardware.

VRAM
: (Video RAM) High-speed random-access memory dedicated to a graphics card, used to store textures, framebuffers, and other data critical for rendering and computation on the GPU.

Tiling
: A technique of dividing a large image into smaller "tiles" to be processed individually. It is an essential mechanism for handling resolutions that exceed available VRAM.

Codec

(Coder-Decoder) A software or hardware algorithm/device that compresses and decompresses digital video data. Examples of software codecs are x264 (H.264) and x265 (HEVC). Hardware codecs (NVENC, AMF, QSV) use dedicated encoders on the GPU to accelerate this process.

11. 1

Support and Community

- **Reporting Issues:** If you encounter a reproducible bug, please open an "Issue" on the GitHub repository. It is essential to attach the `error_log.txt` file (located in your **Documents** folder) for effective diagnosis.
- **Code Contributions:** Contributions to the source code are welcome. It is recommended to follow the standard workflow: fork the repository, create a new branch for the feature or fix, and submit a "Pull Request" for review.
- **Direct Contact:** For general inquiries or technical support that does not constitute a software bug, you can contact the author at negroayub97@gmail.com.

Thank you for using Warlock-Studio.