AI Photo Enhancer - Complete Documentation

A powerful command-line tool for enhancing photos using AI, with special focus on natural portrait enhancement. Similar to Remini app but runs locally on your computer.

📦 Complete Package List & Requirements File

All Required Packages

Create a file named `requirements.txt` with this content:

```text
opencv-python>=4.5.0
torch>=2.0.0
torchvision>=0.15.0
numpy>=1.21.0
pillow>=9.0.0
```

Note: scipy is NOT required. The tool works perfectly with just these 5 packages.

Then install with:

```bash
pip install -r requirements.txt --break-system-packages
```

Or Single Command Installation

```bash
pip install opencv-python torch torchvision numpy pillow --break-system-packages
```

📋 Table of Contents

✨ Features

- **AI-Powered Enhancement**: Uses Real-ESRGAN for professional photo upscaling
- **Face Detection & Enhancement**: Automatically detects and enhances faces naturally
- **Batch Processing**: Process entire folders of photos at once
- **Natural Results**: No over-sharpening or artificial-looking artifacts
- **Multiple Modes**: Auto, face, photo, and anime enhancement modes
- **GPU Acceleration**: Faster processing with NVIDIA GPUs (optional)
- **Flexible Scaling**: 1x (no upscale), 2x, or 4x upscaling options

---

🔧 Prerequisites

System Requirements

Minimum:

- Python 3.7 or higher
- 4GB RAM
- 2GB free disk space (for AI models)

Recommended:

- Python 3.8+
- 8GB RAM
- NVIDIA GPU with CUDA support (for faster processing)
- 5GB free disk space

Check Python Version

```bash
python3 --version
# or
python --version
```

If you don't have Python installed:

- **Linux (Ubuntu/Debian)**: `sudo apt install python3 python3-pip`
- **macOS**: `brew install python3`
- **Windows**: Download from python.org

📦 Complete Installation Guide

Quick Installation (Copy-Paste)

```bash
# Complete installation in one command
pip install opencv-python torch torchvision numpy pillow --break-system-packages

# Verify installation
python3 -c "import cv2, torch, torchvision, numpy, PIL; print('✓ All packages ready!')"

# Test the tool
python3 photo_enhancer.py --help
```

Package Details & Troubleshooting

1. OpenCV (opencv-python)

Purpose: Image processing, face detection
Size: ~90 MB

If installation fails:

```bash
```

```
# Try minimal version
pip install opencv-python-headless --break-system-packages
```

2. PyTorch (torch)

Purpose: AI neural network framework
Size: ~2 GB
Note: Largest package, be patient during download

If installation fails:

```bash
# CPU-only version (smaller, faster download)
pip install torch --index-url https://download.pytorch.org/whl/cpu --break-system-packages
```

For GPU support (NVIDIA only):

```bash
# Check CUDA version first
nvidia-smi

# Install CUDA version (example for CUDA 11.8)
pip install torch --index-url https://download.pytorch.org/whl/cu118 --break-system-packages
```

3. Torchvision (torchvision)

Purpose: PyTorch vision utilities
Size: ~6 MB

If installation fails:

```bash
pip install torchvision --no-deps --break-system-packages
```

4. NumPy (numpy)

Purpose: Array processing and math
Size: ~50 MB

If installation fails:

```bash
# Usually pre-installed, but if needed:
pip install numpy --break-system-packages
```

5. Pillow (pillow)

Purpose: Image file I/O
Size: ~3 MB

If installation fails:

```bash
# Try PIL (alternative name)
pip install PIL --break-system-packages
# or
pip install pillow --break-system-packages
```

Step 1: Download the Script

Save the `photo_enhancer.py` script to your computer. For example:

```bash
cd ~/Downloads
# (Save the photo_enhancer.py file here)
```

Step 2: Install Required Packages

Required Packages List

The tool requires the following Python packages:

Package	Version	Purpose
`opencv-python`	Latest	Image processing and face detection
`torch`	Latest	PyTorch for AI neural networks
`torchvision`	Latest	Additional PyTorch utilities
`numpy`	Latest	Numerical computing
`pillow`	Latest	Image file handling

Installation Command

Open terminal/command prompt and run:

```bash
pip install opencv-python torch torchvision numpy pillow --break-system-packages
```

This single command installs all 5 required packages.

Alternative Installation Methods

Method 1: Virtual Environment (Recommended for beginners)

```bash
# Create virtual environment
python3 -m venv photo_env

# Activate it
source photo_env/bin/activate  # Linux/Mac
# or
photo_env\Scripts\activate  # Windows

# Install all packages
pip install opencv-python torch torchvision numpy pillow
```

Method 2: Using requirements.txt (Convenient)

```bash
# Create requirements.txt file with this content:
# opencv-python
# torch
# torchvision
# numpy
# pillow

# Then install
pip install -r requirements.txt --break-system-packages
```

Method 3: System Package Manager (Linux only)

```bash
# For Debian/Ubuntu
sudo apt update
sudo apt install python3-opencv python3-torch python3-numpy python3-pil

# Note: torchvision might need pip installation
pip install torchvision --break-system-packages
```

Method 4: One by one (If bulk install fails)

```bash
pip install opencv-python --break-system-packages
pip install torch --break-system-packages
pip install torchvision --break-system-packages
pip install numpy --break-system-packages
pip install pillow --break-system-packages
```

Installation Size

- **Total download size**: ~2-3 GB (mainly PyTorch)
- **Installation time**: 5-15 minutes (depending on internet speed)
- **Disk space required**: ~4-5 GB after installation

Step 3: Verify Installation

Test if all packages are installed correctly:

```bash
# Test all packages
python3 -c "import cv2, torch, torchvision, numpy, PIL; print('✓ All packages installed successfully!')"
```

If you see "✓ All packages installed successfully!" - you're ready to go!

Test individual packages:

```bash
```

```
python3 -c "import cv2; print('OpenCV:', cv2.__version__)"
python3 -c "import torch; print('PyTorch:', torch.__version__)"
python3 -c "import torchvision; print('Torchvision:', torchvision.__version__)"
python3 -c "import numpy; print('NumPy:', numpy.__version__)"
python3 -c "import PIL; print('Pillow:', PIL.__version__)"
```

Verify the script works:

```bash
python3 photo_enhancer.py --help
```

If you see the help message, installation is successful! ✓

🚀 Quick Start

Enhance a Single Photo

```bash
# Basic enhancement (auto-detects faces)
python3 photo_enhancer.py photo.jpg

# Output will be saved as: photo_enhanced.jpg
```

Enhance Multiple Photos (Batch)

```bash
# Enhance all photos in a folder
python3 photo_enhancer.py ./my_photos -b ./enhanced_photos
```

That's it! The tool will:

1. Download AI models on first run (~250MB, one-time)
2. Process your photos
3. Save enhanced versions

📖 Usage Guide

Single Image Mode

Basic Syntax:

```bash
python3 photo_enhancer.py INPUT_FILE [OPTIONS]
```

Examples:

```bash
# Auto-detect and enhance
python3 photo_enhancer.py vacation.jpg

# Specify output name
python3 photo_enhancer.py input.jpg -o beautiful_output.jpg

# Force face enhancement mode
python3 photo_enhancer.py portrait.jpg -m face

# 2x upscaling (faster)
python3 photo_enhancer.py photo.jpg -s 2

# No upscaling, just face enhancement
python3 photo_enhancer.py photo.jpg -s 1 -m face

# With subtle sharpening
python3 photo_enhancer.py photo.jpg --sharpen
```

Batch Processing Mode

Basic Syntax:

```bash
python3 photo_enhancer.py INPUT_FOLDER -b OUTPUT_FOLDER [OPTIONS]
```

Examples:

```bash
```

```
# Basic batch processing
python3 photo_enhancer.py ./photos -b ./enhanced

# Batch with face mode
python3 photo_enhancer.py ~/Pictures/family -b ~/Pictures/family_enhanced -m face

# Fast batch (2x scaling)
python3 photo_enhancer.py ./input -b ./output -s 2

# Batch with sharpening
python3 photo_enhancer.py ./photos -b ./enhanced --sharpen

# Batch for portraits
python3 photo_enhancer.py ./portraits -b ./enhanced_portraits -m face -s 4
```

📝 Command Reference

Required Arguments

Argument	Description
`INPUT`	Input file path or folder path

Optional Arguments

Option	Values	Default	Description
`-b`, `--batch`	folder path	-	Enable batch mode, specify output folder
`-o`, `--output`	file path	input_enhanced.ext	Output file path (single mode only)
`-m`, `--mode`	auto, face, photo, anime	auto	Enhancement mode
`-s`, `--scale`	1, 2, 4	4	Upscaling factor (1=no upscale)
`--sharpen`	flag	disabled	Add subtle sharpening
`--help`	flag	-	Show help message

Enhancement Modes

Mode	Best For	Description
`auto`	Mixed photos	Auto-detects faces and enhances them
`face`	Portraits	Forces face enhancement mode
`photo`	Landscapes, general	General photo enhancement
`anime`	Illustrations	Optimized for anime/drawings

Scaling Options

Scale	Speed	Quality	Use Case
`1`	Fastest	Good	Just enhance, no upscaling
`2`	Fast	Good	Quick enhancement with moderate upscaling
`4`	Slower	Best	Maximum quality, 4x resolution increase

💡 Examples

Example 1: Family Photo Album

Scenario: Enhance all family photos from vacation

```bash
# Create output folder
mkdir ~/Pictures/vacation_enhanced

# Process all photos
python3 photo_enhancer.py ~/Pictures/vacation -b ~/Pictures/vacation_enhanced -m auto
```

Example 2: Old Portrait Restoration

Scenario: Restore an old portrait photo

```bash
```

```
# High-quality face-focused enhancement
python3 photo_enhancer.py old_portrait.jpg -m face -s 4 -o restored_portrait.jpg
```

Example 3: Quick Preview

Scenario: Quick test before processing all photos

```bash
# Fast 2x enhancement for testing
python3 photo_enhancer.py test_photo.jpg -s 2 -o preview.jpg
```

Example 4: Social Media Batch

Scenario: Enhance photos for Instagram

```bash
# Batch enhance with moderate quality
python3 photo_enhancer.py ./instagram_photos -b ./enhanced -s 2 -m auto
```

Example 5: Professional Headshots

Scenario: Enhance professional portraits

```bash
# Maximum quality face enhancement
python3 photo_enhancer.py ./headshots -b ./headshots_enhanced -m face -s 4
```

Example 6: Anime Artwork

Scenario: Upscale anime illustrations

```bash
# Anime mode for drawings
python3 photo_enhancer.py ./anime_art -b ./anime_enhanced -m anime -s 4
```

🎯 Tips & Best Practices

Getting Best Results

1. **For Portraits:**
    - Use `-m face` or `-m auto` (auto-detects)
    - Use `-s 4` for maximum quality
    - Don't use `--sharpen` (already optimized)
2. **For Landscapes:**
    - Use `-m photo`
    - `-s 2` is often sufficient
    - Can add `--sharpen` if needed
3. **For Batch Processing:**
    - Start with `-s 2` to test
    - Then process with `-s 4` if satisfied
    - Use auto mode for mixed content
4. **Performance Tips:**
    - Use `-s 2` for faster processing
    - Close other GPU applications
    - Process in smaller batches if RAM limited

Recommended Workflows

Workflow 1: Test First

```bash
# 1. Test on one image
python3 photo_enhancer.py test.jpg -s 2

# 2. If satisfied, batch process
python3 photo_enhancer.py ./photos -b ./enhanced -s 4
```

Workflow 2: Organize by Type

```bash
```

```
# 1. Separate portraits and landscapes
# 2. Process portraits with face mode
python3 photo_enhancer.py ./portraits -b ./enhanced_portraits -m face

# 3. Process landscapes with photo mode
python3 photo_enhancer.py ./landscapes -b ./enhanced_landscapes -m photo
```

🔍 Troubleshooting

Common Issues

1. "Command not found" Error

Problem: `python3: command not found`

Solution:

```bash
# Try with 'python' instead
python photo_enhancer.py photo.jpg

# Or install Python
sudo apt install python3  # Linux
brew install python3      # macOS
```

2. Package Installation Fails

Problem: Individual packages fail to install

Solutions by Package:

OpenCV fails:

```bash
# Try headless version
pip install opencv-python-headless --break-system-packages
```

PyTorch fails or too slow:

```bash
```

```bash
# CPU-only version (smaller download)
pip install torch torchvision --index-url https://download.pytorch.org/whl/cpu --break-system-packages
```

NumPy/Pillow fails:

```bash
# Install from system package manager first
sudo apt install python3-numpy python3-pil  # Linux
brew install numpy pillow  # macOS

# Then try pip again
pip install numpy pillow --break-system-packages
```

All packages fail:

```bash
# Use virtual environment (cleanest solution)
python3 -m venv photo_env
source photo_env/bin/activate
pip install opencv-python torch torchvision numpy pillow
```

3. Model Download Fails

Problem: AI model download interrupted

Solution:

```bash
# Delete incomplete download
rm -rf ~/.cache/photo_enhancer

# Run again with good internet connection
python3 photo_enhancer.py photo.jpg
```

4. Out of Memory Error

Problem: `RuntimeError: CUDA out of memory` or system freezes

Solution:

```bash
```

```
# Use 2x scaling instead of 4x
python3 photo_enhancer.py photo.jpg -s 2

# Or process without upscaling
python3 photo_enhancer.py photo.jpg -s 1 -m face
```

5. Slow Processing

Problem: Takes too long per image

Solutions:

- Use `-s 2` instead of `-s 4`
- Close other applications
- If you have NVIDIA GPU, ensure CUDA drivers installed
- Process smaller images first

6. "No faces detected"

Problem: Face mode says no faces found

Solution:

- Ensure faces are clearly visible
- Try `-m photo` for general enhancement
- Face must be at least 30x30 pixels
- Better lighting helps detection

Getting Help

If you encounter issues:

1. **Check Error Message:**

```bash
python3 photo_enhancer.py photo.jpg 2>&1 | tee error.log
```

2. **Verify Installation:**

```bash
python3 -c "import cv2, torch, torchvision, numpy, PIL; print('All packages OK')"
```

3. **Check Disk Space:**

```bash
df -h ~  # Linux/Mac
```

4. **Test with Simple Command:**

```bash
python3 photo_enhancer.py photo.jpg -s 1 -m face
```

📊 Supported File Formats

Input Formats

- JPEG (.jpg, .jpeg)
- PNG (.png)
- BMP (.bmp)
- TIFF (.tiff, .tif)
- WebP (.webp)

Output Formats

- Same as input format
- Quality: JPEG (95%), PNG (high compression)

⚙️ System Information

File Locations

AI Models Cache:

- Linux/Mac: `~/.cache/photo_enhancer/`
- Windows: `%USERPROFILE%\.cache\photo_enhancer\`

Model Files:

- `RealESRGAN_x4plus.pth` (~67MB)
- `RealESRGAN_anime.pth` (~17MB)

Resource Usage

CPU Mode:

- RAM: 2-4GB per image
- Speed: ~30-60 seconds per image (4x scale)

GPU Mode (CUDA):

- VRAM: 2-4GB
- Speed: ~5-15 seconds per image (4x scale)

🎓 Learning More

Understanding Modes

Auto Mode (Recommended):

- Detects faces automatically
- Enhances faces naturally
- Keeps background minimal
- Best for mixed content

Face Mode:

- Forces face detection
- Best for portraits
- More aggressive on faces
- Use when auto doesn't detect

Photo Mode:

- General enhancement
- No face detection
- Good for landscapes
- Uniform processing

Anime Mode:

- Optimized for illustrations
- Better for line art
- Preserves drawing style
- Use for cartoons/anime

Enhancement Pipeline

The tool processes images in stages:

1. **Detection**: Finds faces in image
2. **Upscaling**: Uses AI to increase resolution
3. **Face Enhancement**: Special processing for detected faces
4. **Blending**: Seamlessly merges enhanced faces
5. **Final Touches**: Subtle color and contrast adjustment
6. **Save**: High-quality output file

📞 Quick Reference Card

```
┌──────────────────────────────────────────────────────┐
│        AI PHOTO ENHANCER CHEAT SHEET          │
├──────────────────────────────────────────────────────┤
│                                │
│ SINGLE IMAGE:                             │
│   python3 photo_enhancer.py photo.jpg          │
│                                │
│ BATCH FOLDER:                              │
│  python3 photo_enhancer.py ./input -b ./out    │
│                                                 │
│ COMMON OPTIONS:                                 │
│  -m face      Force face mode                   │
│  -m photo     General photos                    │
│  -s 2         2x scaling (fast)                 │
│  -s 4         4x scaling (best)                 │
│  --sharpen    Add sharpening                    │
│                                                 │
│ EXAMPLES:                                       │
│  Portrait:    -m face -s 4                      │
│  Landscape:   -m photo -s 2                     │
│  Quick test:  -s 2                              │
│  Anime:       -m anime -s 4                     │
│                                                 │
└─────────────────────────────────────────────────┘
```

📄 Version & Credits

Version: 1.0
Created: 2024
AI Model: Real-ESRGAN (xinntao et al.)
License: Free for personal use

🙏 Acknowledgments

This tool uses:

- **Real-ESRGAN**: For AI-powered super-resolution
- **OpenCV**: For image processing
- **PyTorch**: For neural network inference
- **Pillow**: For image handling

Happy Enhancing! 📸✨

For questions or issues, check the Troubleshooting section above.