#### ****Use a Combined New Environment in Miniconda (Best Long-Term Solution)****

We need to install two python Packages:

**1. Realesrgan**

**2. Torchvision**

If you want a **clean setup**, create a **new** environment that includes both:

1. **Create a new environment:**

conda create --name upscale\_env python=3.12 -y

1. **Activate the environment:**

conda activate upscale\_env

1. **Install PyTorch & RealESRGAN together:**

pip install realesrgan

pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121

1. **Run your script:**

python /path/to/script/upscale.py

### script on the next page

#!/usr/bin/env python3

import os

import torch

import numpy as np

from PIL import Image

import torchvision.transforms as transforms

from realesrgan.utils import RealESRGANer

from basicsr.archs.rrdbnet\_arch import RRDBNet # Required for initializing the model

# Check for GPU availability

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

# Initialize the RRDBNet model

model = RRDBNet(num\_in\_ch=3, num\_out\_ch=3, num\_feat=64, num\_block=23, num\_grow\_ch=32, scale=4)

# Load Real-ESRGAN

upscaler = RealESRGANer(

scale=4,

model\_path="https://github.com/xinntao/Real-ESRGAN/releases/download/v0.1.0/RealESRGAN\_x4plus.pth",

model=model,

tile=0, # Set tiling for large images, change if needed

tile\_pad=10,

pre\_pad=0,

half=torch.cuda.is\_available(), # Use half-precision if CUDA is available

device=device,

)

# Get list of all JPG files in the current directory

jpg\_files = [f for f in os.listdir() if f.lower().endswith(".jpg")]

if not jpg\_files:

print("No JPG files found in the current directory.")

exit()

# Process each JPG file

for jpg\_file in jpg\_files:

try:

# Open image

img = Image.open(jpg\_file).convert("RGB")

# Upscale the image (Real-ESRGAN outputs a NumPy array, NOT a tensor)

output\_np, \_ = upscaler.enhance(np.array(img), outscale=4)

# Ensure pixel values are in the correct range

output\_np = np.clip(output\_np, 0, 255).astype(np.uint8)

# Convert to PIL Image

upscaled\_img = Image.fromarray(output\_np)

# Save as PNG with the same name

output\_file = f"{os.path.splitext(jpg\_file)[0]}\_upscaled.png"

upscaled\_img.save(output\_file, format="PNG")

print(f"Upscaled: {jpg\_file} → {output\_file}")

except Exception as e:

print(f"Error processing {jpg\_file}: {e}")

print("Upscaling complete!")