Building Executables for Human Blur Tool GUI

This guide explains how to build standalone executables for the Human Blur Tool GUI on Windows, macOS, and Linux.

Table of Contents

Prerequisites

Before building executables, ensure you have:

1. **Python 3.8 or higher** installed
2. **All dependencies** installed:
   ```
   bash
      pip install -r requirements.txt
   ```
3. **PyInstaller** installed:
   ```
   bash
      pip install pyinstaller
   ```

Building on Windows

Method 1: PyInstaller (Recommended)

1. **Open Command Prompt or PowerShell** and navigate to the project directory:
   ```
   cmd
      cd path\to\human-blur-tool
   ```

2. **Create the executable**:
   ```
   cmd
      pyinstaller --onefile --windowed --name "HumanBlurTool" --icon=logo.png --add-data
   "logo.png;." gui.py
   ```

3. **Find your executable**:
   - Location: `dist\HumanBlurTool.exe`
   - The executable will be in the `dist` folder

4. **Optional: Create installer with Inno Setup**:
   - Download Inno Setup (https://jrsoftware.org/isinfo.php)

- Create a setup script to bundle the executable
- This creates a professional installer for Windows users

Explanation of PyInstaller Flags

- `--onefile`: Creates a single executable file
- `--windowed`: Prevents console window from appearing (GUI only)
- `--name`: Sets the executable name
- `--icon`: Sets the application icon (optional)
- `--add-data`: Includes the logo image file
- Format on Windows: `source;destination`
- Format on macOS/Linux: `source:destination`

Testing the Windows Executable

```
cd dist
HumanBlurTool.exe
```

Building on macOS

Method 1: PyInstaller (Recommended)

1. **Open Terminal** and navigate to the project directory:
   `bash`
   `   cd /path/to/human-blur-tool`

2. **Create the app bundle**:
   `bash`
   `   pyinstaller --onefile --windowed --name "HumanBlurTool" --icon=logo.png --add-data "logo.png:." gui.py`

3. **Find your application**:
   - Location: `dist/HumanBlurTool.app`
   - The `.app` bundle will be in the `dist` folder

4. **Optional: Create DMG installer**:
   ```bash
   # Install create-dmg
   brew install create-dmg

# Create DMG
create-dmg \
–volname “Human Blur Tool” \
–window-pos 200 120 \
–window-size 600 400 \
–icon-size 100 \
–app-drop-link 425 120 \
“HumanBlurTool.dmg” \
“dist/HumanBlurTool.app”
```

Code Signing (for Distribution)

If you plan to distribute the macOS app, you'll need to code sign it:

```
# Sign the app
codesign --deep --force --verify --verbose --sign "Developer ID Application: Your
Name" dist/HumanBlurTool.app

# Verify signature
codesign --verify --deep --strict --verbose=2 dist/HumanBlurTool.app
```

Testing the macOS Application

```
open dist/HumanBlurTool.app
```

Building on Linux

Method 1: PyInstaller (Recommended)

1. **Open Terminal** and navigate to the project directory:
   `bash`
   `cd /path/to/human-blur-tool`

2. **Create the executable**:
   `bash`
   `pyinstaller --onefile --windowed --name "HumanBlurTool" --add-data "logo.png:." gui.py`

3. **Find your executable**:
   - Location: `dist/HumanBlurTool`
   - The executable will be in the `dist` folder

4. **Make it executable** (if not already):
   `bash`
   `chmod +x dist/HumanBlurTool`

Method 2: AppImage (Cross-Distribution)

AppImage creates a portable executable that works across Linux distributions:

1. **Install python-appimage**:
   `bash`
   `pip install python-appimage`

2. **Create AppImage**:
   `bash`
   `python-appimage build app -l manylinux2014_x86_64 gui.py`

Method 3: Debian Package (.deb)

For Debian/Ubuntu users:

1. **Install fpm** (Effing Package Management):
   `bash`
   `   gem install fpm`

2. **Create package structure**:
   ```bash
   mkdir -p package/usr/local/bin
   mkdir -p package/usr/share/applications
   mkdir -p package/usr/share/icons

cp dist/HumanBlurTool package/usr/local/bin/
cp logo.png package/usr/share/icons/humanblur.png
```

1. **Create desktop entry** (`package/usr/share/applications/humanblur.desktop`):
   ```
   ini
      [Desktop Entry]
      Name=Human Blur Tool
      Comment=AI-Powered Privacy Protection
      Exec=/usr/local/bin/HumanBlurTool
      Icon=/usr/share/icons/humanblur.png
      Terminal=false
      Type=Application
      Categories=Graphics;Photography;
   ```

2. **Build .deb package**:
   `bash`
   `   fpm -s dir -t deb -n humanblur-tool -v 3.0 -C package`

Testing the Linux Executable

```
./dist/HumanBlurTool
```

Advanced PyInstaller Configuration

For more control, create a `.spec` file:

```
pyinstaller --name "HumanBlurTool" gui.py --windowed --onefile
```

This creates `HumanBlurTool.spec`. Edit it for advanced configuration:

```python
# -*- mode: python ; coding: utf-8 -*-

block_cipher = None

a = Analysis(
    ['gui.py'],
    pathex=[],
    binaries=[],
    datas=[('logo.png', '.')],
    hiddenimports=[
        'PIL._tkinter_finder',
        'cv2',
        'numpy',
        'ultralytics',
        'torch',
    ],
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=block_cipher,
    noarchive=False,
)

pyz = PYZ(a.pure, a.zipped_data, cipher=block_cipher)

exe = EXE(
    pyz,
    a.scripts,
    a.binaries,
    a.zipfiles,
    a.datas,
    [],
    name='HumanBlurTool',
    debug=False,
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
    upx_exclude=[],
    runtime_tmpdir=None,
    console=False,
    disable_windowed_traceback=False,
    argv_emulation=False,
    target_arch=None,
    codesign_identity=None,
    entitlements_file=None,
    icon='logo.png'
)

# For macOS .app bundle
app = BUNDLE(
    exe,
    name='HumanBlurTool.app',
    icon='logo.png',
    bundle_identifier='org.globalemancipation.humanblur',
    info_plist={
        'NSHighResolutionCapable': 'True',
        'LSMinimumSystemVersion': '10.13.0',
    },
)
```

Build using the spec file:

```
pyinstaller HumanBlurTool.spec
```

Reducing Executable Size

Large executables are common with PyTorch and OpenCV. Here are optimization tips:

1. Exclude Unused Modules

```
pyinstaller --onefile --windowed \
    --exclude-module matplotlib \
    --exclude-module scipy \
    --exclude-module pandas \
    gui.py
```

2. Use UPX Compression

```
# Install UPX
# Windows: Download from https://upx.github.io/
# macOS: brew install upx
# Linux: sudo apt-get install upx

pyinstaller --onefile --windowed --upx-dir=/path/to/upx gui.py
```

3. Strip Debug Symbols (Linux/macOS)

```
pyinstaller --onefile --windowed --strip gui.py
```

Troubleshooting

Issue: “ModuleNotFoundError” in built executable

Solution: Add hidden imports to PyInstaller:

```
pyinstaller --onefile --windowed \
    --hidden-import=PIL._tkinter_finder \
    --hidden-import=cv2 \
    --hidden-import=ultralytics \
    gui.py
```

Issue: Logo image not found

Solution: Ensure logo is bundled correctly:
- Windows: `--add-data "logo.png;."`
- macOS/Linux: `--add-data "logo.png:."`

Fix in code (`gui.py`):

```python
import sys
from pathlib import Path

# Get correct path for bundled resources
if getattr(sys, 'frozen', False):
    # Running as compiled executable
    application_path = Path(sys._MEIPASS)
else:
    # Running as script
    application_path = Path(__file__).parent

logo_path = application_path / "logo.png"
```

Issue: YOLO models not downloading

Solution: Pre-download models and bundle them:

1. Download models first:
   `python`
   ```
   from ultralytics import YOLO
   YOLO('yolov8n-seg.pt')
   YOLO('yolov8s-seg.pt')
   YOLO('yolov8m-seg.pt')
   ```
2. Find model location (usually `~/.cache/ultralytics/`)
3. Bundle with executable:
   `bash`
   ```
   pyinstaller --add-data "models/*:models" gui.py
   ```

Issue: Executable is too large (>500MB)

Cause: PyTorch and dependencies are large

Solutions:
1. Use CPU-only PyTorch:
`bash`
```
pip uninstall torch torchvision
pip install torch torchvision --index-url https://download.pytorch.org/whl/cpu
```

1. Create directory-based build instead of `--onefile`:
   `bash`
   ```
   pyinstaller --windowed gui.py
   ```
   This creates a folder with the executable and dependencies, which is faster and smaller.

Issue: macOS “App is damaged and can’t be opened”

Cause: Gatekeeper security on macOS

Solution:

```
# Remove quarantine attribute
xattr -cr dist/HumanBlurTool.app

# Or tell users to run:
xattr -cr /path/to/HumanBlurTool.app
```

Issue: Slow startup time

Cause: Large dependencies being loaded

Solutions:
1. Use directory build instead of `--onefile`
2. Lazy load heavy modules
3. Pre-compile Python files

Distribution Checklist

Before distributing your executable:

- [ ] Test on a clean machine without Python installed
- [ ] Test all features (file selection, folder processing, help dialog)
- [ ] Verify logo displays correctly
- [ ] Check error messages are user-friendly
- [ ] Include README with system requirements
- [ ] Test on the target OS version
- [ ] For macOS: Code sign if possible
- [ ] For Windows: Consider creating an installer
- [ ] Include LICENSE file
- [ ] Document any third-party dependencies

File Size Expectations

Expected file sizes (approximate):

- **Windows**: 400-800 MB (with PyTorch)
- **macOS**: 450-900 MB (with PyTorch)
- **Linux**: 400-800 MB (with PyTorch)

CPU-only PyTorch builds are typically 50-100 MB smaller.

Additional Resources

- PyInstaller Documentation (https://pyinstaller.org/en/stable/)
- Ultralytics YOLOv8 Documentation (https://docs.ultralytics.com/)
- Python Packaging Guide (https://packaging.python.org/)
- Inno Setup (Windows Installer) (https://jrsoftware.org/isinfo.php)
- create-dmg (macOS DMG Creator) (https://github.com/create-dmg/create-dmg)

Support

For build issues or questions about distribution, please contact:
apps@globalemancipation.ngo