

LogoScale - A Method for Vectorizing Small, Crappy Logos



Abstract

During the height of the Coronavirus pandemic, I found myself doing a ton of work with ESRGAN, an AI-powered image superscaling tool.

ESRGAN was invented to help governments see more detail with older surveillance satellites, but eventually declassified and brought into the open source world. The principle is that a developer creates situation-specific models by showing the AI two sets of images — the original ones, and larger versions of the image. The AI then calculates the patterns of the lower-resolution images, and then corresponds them to areas and textures that it sees in the source imagery.

Tinkerers in the space discovered that someone can create utility models from banks of large, high resolution images, then saving them into highly compressed or highly interlaced states. This gave way to cool stuff like methods for removing JPEG compression from images, or even adding texture *back* to the videos.

I got really absorbed into making my own tools that could aid in some otherwise daily design tasks. **I started thinking about the daily frustrations I had, and being sent a terribly compressed, JPEG or PNG logo instead of a vector format was chief among them.** But, the issue is, oftentimes a client or a stakeholder will send you an inferior format, and then disappear off the face of the planet. **With this in mind, I set out to make a tool that could superscale the low-resolution logos we get sent to something vectorized that can be relied on in a pinch to get a job all the way to completed, without needing to wait around for someone to find a logo file that may not even exist anymore.**

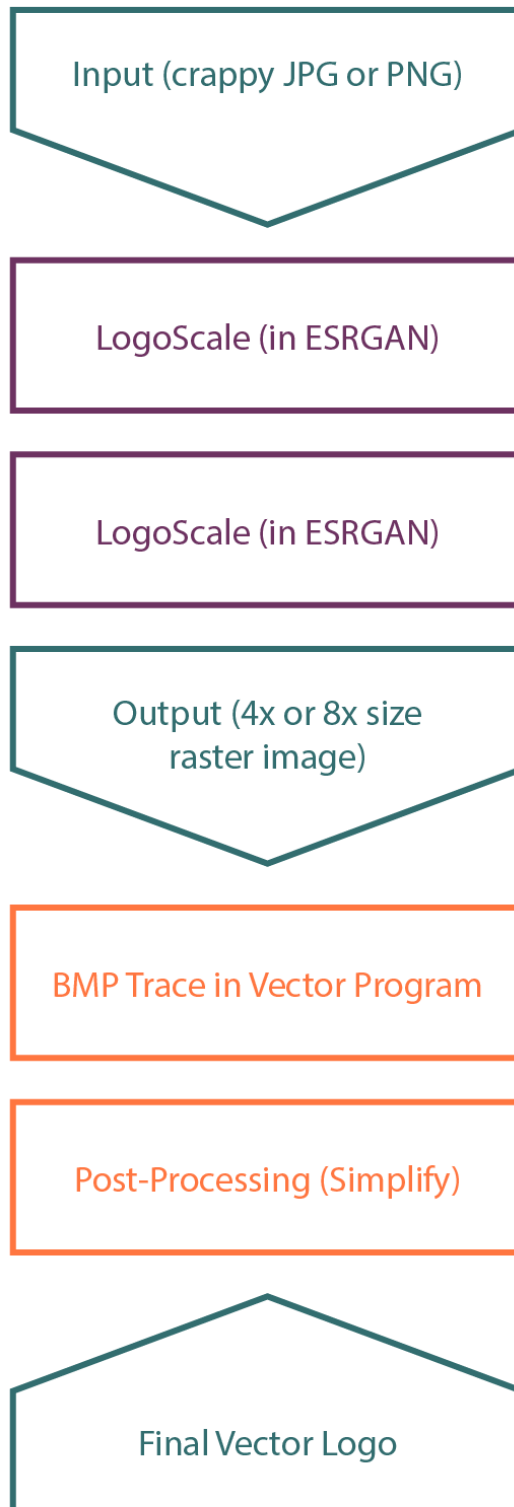
The result is `LogoScale`, a FLOSS upscaling model for use with ESRGAN-compatible tools.

Purpose

LogoScale is designed to be a two-step process — it cannot directly output vector images from raster ones. The input is your crummy logo file, and the output is a raster image that is 4x the size. You can also chain the model's output back into itself to increase the output size by another 4x. The model will attempt to account for JPEG artifacts, or weird jaggy pixels left over from PNG-8's habit of crushing of the alpha channel.

This raster image must then be placed into vector software capable of performing a bitmap trace, such as Adobe Illustrator, or Inkscape. The larger the resolution, the more accurate the live trace will be to the original logo. It is not 100% accurate, but in cases where the original vector imagery was lost or is irrecoverable, this can be a lifesaver. Plus, a little post-processing work can make many logos quite close to the original.

Method



Data

The training data was scraped from WikiMedia using a tool pointed at every file [listed on this page](#). This method is reproducible for specific subsets of logos, such as SVGs of soft drink brand logos, etc.



Examples

LogoScale Examples

Aa Name	🕒 Created	☰ Tags
<u>Tesco</u>	@September 9, 2023 12:34 AM	
<u>Shell</u>	@September 9, 2023 12:34 AM	
<u>Roxy</u>	@September 9, 2023 12:34 AM	
<u>IBM</u>	@September 9, 2023 12:34 AM	
<u>Batman</u>	@September 9, 2023 12:34 AM	
<u>Ford</u>	@September 9, 2023 12:34 AM	

As you can see, logo complexity really is no big deal for this model, as it is not considering the whole composition, only a grid of squares. Its only job is to make sure those squares line up at the edges.

Download

-  [4x LogoScale 125k.pth](#)
-  [4x LogoScale 9k.pth](#)

Recommended Software

- [Cupscale](#), Windows

- [ESRGAN's CLI interface](#), Linux
- [Upscaler](#), Linux

License

This model is entirely *libre* (free as in freedom), and can be used for any commercial or personal purpose. My only request is that you go forth and fill the world with SVGs.