Stanford ENGINEERING
Computer Science

Towards Automatic Icon Design Using Machine Learning (Moses Soh)

Stanford University

The Problem and Our Approach

Icon design is a hard problem. It often involves expert designers handcrafting beautiful icons by choosing weights, shapes and colors. *Yet*, it's easy to find simple outline icons on the internet. Can we learn how a designer would color in and stylize an icon given only its outline?

We model this as a **supervised learning problem**, where we predict RGB pixels from grayscale pixels of the outline icon. Our final architecture draws on Conditional Generative Adversarial Networks (Mirza et. al., 2014) to train a generator that minimizes **L1 loss** and **adversarial loss** from a discriminator.

Model Architecture

Our generator G is the U-Net architecture popularized by the pix2pix paper (Isola et. al. 2016). Define `Ck`, `CDk`, and `CUk` as **k**-layer convolutions followed by Batch Norm and Leaky ReLUs. `CD` and `CU` down and up-sample by 2x respectively. Then **G:** $[0,1]^{128x128}$ (outline) → $[0,1]^{3x128x128}$ (color icon) is defined by:

Encoder. `C32-CD64-C64-CD128-C128-CD256-C256-CD512-C512`
Decoder. `CU512-C256-CU256-C128-CU128-C64-CU64-C32-C3`
Skip-Connections. These connect each `Ck` in encoder to `Ck` in decoder so that information that doesn't make it past the bottleneck can still be used (e.g. the icon outline itself).
Sigmoid layer. This is used to squeeze outputs between [0,1].

Our discriminator D learns to differentiate real icons from generated icons. **D:** $[0,1]^{3x128x128}$ → **[0,1]** can be defined as:

`CD32-C32-CD64-C64-CD128-C128-CD256-DENSE`

Leaky ReLUs with slope = 0.2 mitigate vanishing gradients that plague GAN training. The `DENSE` layer produces a probability.

Loss Functions

Per-pixel L1 loss is used to optimize **G**. In our experiments, we find L1 loss enables **G** quickly learns high-level patterns such as not coloring in the background, coloring within the outline, and rough color distributions. **Adversarial loss** is used to optimize **G** as well. **G** is trained to fool **D** (i.e. **D** should produce a probability close to 1 when fed with **G**'s output). **Binary Cross Entropy loss** is used to optimize **D** to produce 1 when seeing real icons and 0 when seeing generated icons.

Experiments

Q1: How well can purely using L1 loss work?
A: Pretty good but with two problems. Quantitatively, we achieve validation Mean Absolute Error (MAE) of 0.032 per pixel just using L1 loss, and training MAE of 0.010. Qualitatively, **G** learned where to color, and learned to shade the icon edges darker orange for a 3D look as per original icons.
Error analysis reveals that major source of MAE comes from the model (1) being unable to deal with icons of different scales from what it saw in the training set and (2) predicting the color **yellow** instead of **blue** or **green** where it should have. This is the averaging problem that L1 loss is known for since it incentivizes the model to output the average color.
Next step We made a data augmentation pipeline so **G** is invariant to scale, position, noise and sharpness of edges.

Q2: How much did each type of data augmentation help?
A: Data augmentation helped generalization. We achieved validation MAE of 0.0213. We ran **ablative analysis** (Fig 1) and found that "rescaling" and "repositioning" were most helpful.

Base model →	Rescale →	Reposition →	Add Blur →	Noise
0.0320	0.0272	0.0249	0.0225	0.0213

Fig 1. MAE Resultsof Ablative Analysis of Data Augmentation Pipeline

Experiments (cont'd)

At this point, we checked our validation results by running **G** on completely out-of-sample icons to make sure there was no over-fitting (Results in Fig 2). Results were very satisfactory but color repreoduction was wanting.

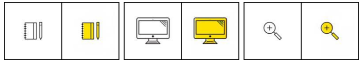

Fig 2. **G** colors in out-of-sample icons very well (qualitatively). However, color reproduction of greens and blues can be improved

Q3: Can adversarial loss improve reproduction of more vibrant but underrepresented colors?
A: Yes it could. We ran the GAN alternating between training **D** and **G**. Fig 3 shows how colorization changed after training with adversarial loss for the same input icons.

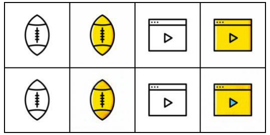

Fig 3. **G** uses dark orange and blue much more as shown by its colorization of the rugby ball and media player above.

Conclusions and next steps

L1 loss and **Adversarial loss** help **G** learn **different aspects of icon design** (structure and color respectively). The **U-Net works** for the multi-modal and ambigious icon design problem. However, **G still struggles on extremely multi-modal problems (e.g. more colors).** The yellow icons above are easier to learn due to there being 4-5 main colors. When adversarial loss fails to overcome the ambiguity of the icon colorization problem, we think the next most promising step is to explore **incorporating user hints** to resolve the multi-modality.