Hand-Crafting Neural Networks for Art-Making

Erik Ulberg[1], Daniel Cardoso Llach[2], Daragh Byrne[2]
Computational Design Laboratory, School of Architecture
Carnegie Mellon University
Pittsburgh, PA 15232 USA

Abstract

A growing number of visual artists use neural networks in their practice. While these networks show promise as an art form, the lack of interpretability limits control to high level decisions based on observations. As an alternative, this research investigates the hand-crafting of network weights coupled with explanatory visualizations as a form of creative control over the internal and lower level processes. Two experimental tools were developed: one for parametrically generating first layer kernels and the second for editing multiple layers. These tools attempt to transform the hand-crafting of features into "crafting" in a richer sense by bringing network weights and visual materials into a tight feedback loop. The first author extensively engaged with these tools and these case studies serve to examine the affordances of internal interaction for art-making. The findings suggest that direct manipulation can be used intentionally and can yield insights into network representations, but that hand-crafting networks of greater sophistication would likely require a hybrid approach integrating data-driven methods.

Introduction

Neural networks show promise as a form of representation in art. However, the available tools for manipulating networks limit creative control to decisions around datasets, algorithms, and hyperparameters (RunwayML). While these tools are valuable for their ease of use, they do not fully leverage artists' fine-grained knowledge of the construction of images. Interaction happens on the exterior of networks based on high level observations of input and output. The internal and lower level processes of networks are overlooked as a potential site of engagement for artists.

AI researcher Christopher Olah asked the provocative question: "What if we treated individual neurons, even individual weights, as being worthy of serious investigation? (Olah et al. 2020b)" Explainable AI (XAI) research has demonstrated that networks often contain a rich world of visual concepts within their intermediate layers (Olah et al. 2020a; Sharif Razavian et al. 2014). This suggests that the right tools could enable artists to closely engage with network interiors and compose with low-level building blocks.

Two experimental tools for hand-crafting network weights were developed to explore Olah's question: one for parametrically generating weights in the first layer of a network and the other for editing multiple layers. These tools do not use machine learning to train network models. Instead, they explore what can be accomplished through close study and the hand-crafting of individual network weights. In the context of neural networks, the term *hand-crafted* refers to features designed by humans rather than learned from data. In creative fields, *hand-crafted* is often associated with workmanship, or the subconscious dexterity emerging from intimate experience with a material (Pye 1968). The tools presented here seek to reposition the crafting of networks to be a form of workmanship. Editing weights and visual materials in a tight feedback loop is intended to create a deeper level of engagement and to unveil new co-creative possibilities.

Background

Understanding and Interacting with AI

Deep neural networks encode visual concepts using tens of millions of weights in ways that are not well understood. To maximize their usability, many AI-powered artistic tools leave network weights as a black box and focus on high level, external controls. Thus, the interaction paradigm for creative AI applications tends to fall under the umbrella of what Zhu et al. (2018) define as Observable AI (OAI). In OAI, users build a mental model of a network's function through observation of inputs and outputs.

OAI stands in contrast to XAI where the goal is to shed light on the inner functioning of networks. Research in XAI often includes interactive visualizations of the flow of weights or renderings of visual concepts within networks (Smilkov et al. 2017; Olah et al. 2020a). These efforts have yielded some insight, leading to the prevailing wisdom that network layers contain progressively higher orders of abstraction.

On close inspection, Olah et al. (2020a) found that the early layers of a network trained on natural images built up recognition from simple concepts such as edge detection, color contrast, and lines to higher level features including corners, patterns, intersections, and shapes. This evidence points to the existence of universal building blocks at intermediate levels in networks. Practitioners often leverage these building blocks through transfer learning to avoid

Proceedings of the 11th International Conference on Computational Creativity (ICCC'20)
ISBN: 978-989-54160-2-8

training from scratch on new datasets (PyTorch). Their existence also provides motivation to find tools for manipulating the low-level, internal processes of networks.

Low-level interaction with network weights is generally viewed as impractical, but forcing a confrontation challenges users to deepen their understanding of networks. Smilkov et al. (2017) argue that rapid, direct manipulation readily provides intuitions on how networks function and sought to demonstrate this through their popular web interface "A Neural Network Playground." Similarly, the hand-crafting tools in this paper are meant to provide intuitions through open-ended interaction and internal visualizations. While explanations of internal processes of networks is less favored in creative practice, this project uses it alongside observation. The tools presented here combine elements of both XAI and OAI by relying on observations of a generative drawing system in tandem with visualization tools for exploring and directly editing the inner weights of networks. The hope is that combining stimulating feedback with the hand-crafting of network weights will lead to fruitful artistic interactions.

Hand-Crafting Networks

Hand-crafting is used sparingly with neural networks because of the relative success of learned features. A number of studies have compared the performance of hand-crafted features to those derived through machine learning. These studies found mixed results showing that hand-crafted features can reduce training time, but struggle to match the accuracy of learned features (Antipov et al. 2015; Zhang et al. 2020). Regardless of these results, quantitative measures of success are less relevant in creative practice. The purpose of this project is not to improve the predictive accuracy of networks through new tools for hand-crafting, but to apply the technique in an art-making context where open-ended interaction and creative control are valued.

Directly crafted weights have generally been overlooked by the creative community. One partial exception that reinforces this observation comes from "Neural Glitch" by artist Mario Klingemann (2018). In his project, Klingemann manually altered the internal weights of a generative adversarial network to probe its inner representations. His results retain some of the appearance of the original output, but with haunting effects. Klingemann's work reveals the delicate balance of weights within networks and shows how even small changes can have a dramatic impact on the learned structures. The name itself, "Neural Glitch," contains the prospects facing an artist attempting to edit network weights by hand. The expectation is that direct manipulation will be a search for interesting glitches rather than a series of intentional choices. This project attempts to challenge that boundary.

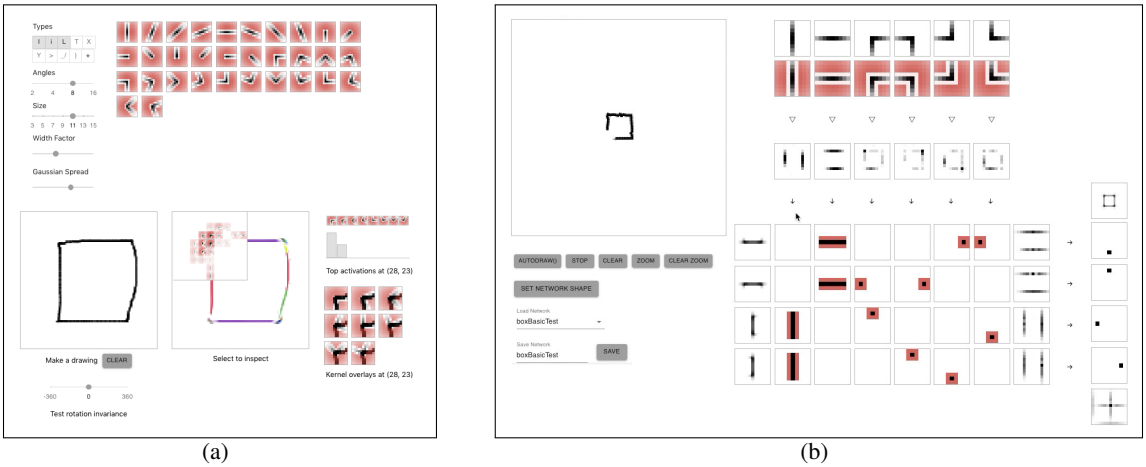
Hand-Crafting Tools

The tools developed as part of this work are called the Kernel Tuner and the Network Builder (Fig. 1). They combine an editable canvas, interactive visualizations, and the hand-crafting of weights.

The Kernel Tuner is a parametric tool for crafting a single layer of weights to extract basic features from a line drawing. Generally, the process of using it begins by making a drawing on the canvas to serve as the basis for evaluating sets of parameters. Next, the user adjusts the sliders within the interface to rapidly test different parameters. The canvas can also be rotated to see how tolerant the kernels are to variance. While these updates are made, the Kernel Tuner provides real time visualizations of how the network responds. Through this iterative process, the tool facilitates human-led jumps through the search space of possible kernels.

The Network Builder uses the kernels from the Kernel Tuner as its first layer and assists with the more ambitious goal of hand-crafting multiple layers of convolution and pooling. Since explanation becomes more difficult with multiple layers, the Network Builder also offers opportunities for understanding the network's function through observation. It achieves this by plugging in the network as a reward function for a generative line drawing system.

The line drawing system operates iteratively on the cur-



(a) (b)

Figure 1: The (a) **Kernel Tuner** and (b) **Network Builder** combine interactive visualizations, a canvas, and editable weights.

rent state of a canvas by drawing or erasing marks of a few pixels in length. The algorithm has two options: it can start a new line or continue a line it is already drawing. Either way, it generates batches of random segments, tests how each segment changes the activation score, and then chooses the highest score. Through this method, the algorithm greedily maximizes the activation of the network. The system terminates when it can no longer find segments that sufficiently improve the activation score (based on a tuned threshold). Additionally, it uses a line end detector to inject options that connect to existing line ends. This small modification makes the system much more likely to draw meaningful shapes.

Typically, the process of using the Network Builder starts with the manual entry of kernel weights (with the help of functions for rotation, reflection, and shifting). Then, the generative algorithm is run to produce several sample outputs to evaluate the success of the weights. If the samples do not match expectations, the generative algorithm is run again until it reproduces an aspect that is not intended (such as premature stopping). The user can pause the algorithm to draw and erase on the canvas while observing changes in network activations. This helps to explain why the final activation score is not responding as expected. Following inspection of the network activations, the steps are repeated. This crafting process involves both observation (of the generative system's response to a given canvas state) and explanation (through the tracing of network weights).

The first author used the Kernel Tuner and Network Builder to experiment with hand-crafting weights as a means of creating line drawings. In addition, the author explored various calibrations of the generative algorithm to produce artistic output. The following sections provide an overview of case studies for each tool and a short description of the artistic process utilizing the tools.

Findings & Discussion

Kernel Tuner

As a specific example, the Kernel Tuner was used to produce a set of weights for a line end detector for the drawing system (Fig. 2). After about an hour of experimentation, a set of eight kernels of five pixels across was chosen (Fig. 2a). Fewer and smaller kernels minimized the number of calculations performed during convolution and thus allowed the algorithm to operate more efficiently. Since the detector did not have to be perfect, these kernels were an attractive balance between speed and accuracy. It is important to note that the eight kernels produced were not a one-size fits all solution for detecting line ends in images. Instead, they were a solution for a particular creative project with a certain type of image.

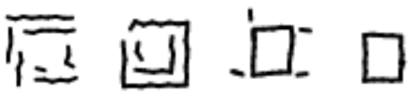
A distinct advantage of the Kernel Tuner was the visibility into the effects of different options. The explanatory visuals provided insights into the relative balance between kernels. Initially, our intuition was that these convolutional kernels would give each pixel a name such as "upwards facing line end" or "left facing right angle." Closely watching the activations and interacting with the system demonstrated how difficult it was to disentangle signals. As more types and rotations of kernels were added many kernels were activated at any given pixel. This suggested that the internal abstractions in a network are better thought of in terms of adjectives on continuous scales rather than one-hot vectors of labels matching nouns. A patch of pixels cannot simply be labeled as a "corner." It is "corner"-like, but also "vertical line"-like and "horizontal line"-like. The Kernel Tuner supported exploration, produced useful kernels, and yielded insights into the internal representations of neural networks.

(a) (b)

Figure 2: (a) Kernels for detecting line ends. (b) Ends detected on a drawing (highlighted in red).

Network Builder

The Network Builder was used to create flexible detectors of increasingly complex visual concepts including boxes (Fig. 3), houses, and bottles (Fig. 4). One of the first networks designed with the Network Builder was for robustly detecting different-sized boxes. To start, the network was designed using positive weights corresponding to vertical and horizontal lines. This attempt produced extra lines and lacked corners (Fig. 3a). Various patterns of construction were explored, such as overlap between parts, sizes of kernels, the depth of negative margins, and the relative magnitude of weights (Fig. 3b and c). After hours of experimentation, the proper combination of positive connections to lines and corners, as well as negative margins yielded flexibly shaped boxes with a single continuous line and no extra artifacts (Fig. 3d).



(a) (b) (c) (d)

Figure 3: A progression of boxes produced by the generative line drawing system as the Network Builder was used to produce a flexible box-detecting network.

Throughout these experiments, the Network Builder facilitated updating the weights and understanding their effects, but it fell short of providing clear intuitions as to the impact of a given change. The complexity of balancing weights within and between features made encoding anything beyond trivial concepts impractical. Even encoding a single line with robustness to different positions proved challenging as it was difficult to predict whether fixing one part would cause an issue somewhere else.

Artistic Output

Next, we explored the production of artistic outputs using these tools. The generative system (using the bottle-detecting network as a reward function) could be run on user

input as seen in Fig. 4 or on a blank canvas as seen in Fig. 5. Fig. 5 demonstrates a more complex workflow. Multiple versions of the generative system maximizing different parts of the bottle-detecting network (starting from the lower layers and moving up) were run. The artistic process involved writing small programs for running the generative algorithm as well as tweaking thresholds and parameters within the algorithm (such as the ratio of drawing to erasing or when to halt). Fig. 5 also demonstrates the further step of printing out and hand painting the result.

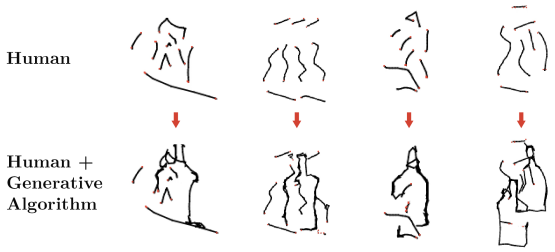
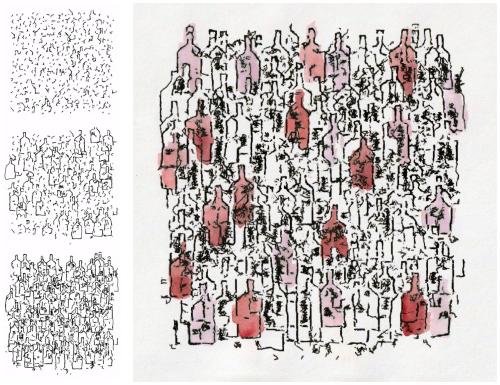


Figure 4: The four images on top are random human input. The generative algorithm (with bottle-detecting network as reward function) was run on the drawings until it halted, altering the drawings to look like bottles (shown at bottom).

Our goal with this conceptual work was to leverage the system's strength, which is its ability to robustly and dynamically respond to input, and to see if direct manipulation of network weights could be used to make intentional output. The findings demonstrate that the system is able to respond to human or random input and to transform it into rough, but recognizable, shapes.



(a) (b)

Figure 5: (a) A sequence where the generative algorithm progressively builds up a drawing from a blank canvas. (d) The result printed out and hand-painted. For more documentation visit: github.com/ulberge/interactive-network

Conclusion

This paper has proposed and evaluated two tools for handcrafting networks for art-making. The purpose of these tools was to facilitate manipulating network weights by providing information at the point of action. This research demonstrated the possibility of discovering insights and making intentional, granular decisions through direct engagement, yet the successes were relatively minor. The findings indicated that crafting more complex visual concepts within networks would likely require data-driven methods. To address this, future work could explore targeted training while maintaining the interactive visualizations inside the networks. One possibility would be to manipulate weights through synthetic datasets coupled with rich labeling at multiple levels of abstraction. Operating on the interior of neural networks presents challenges, but understanding the abstract encodings contained within them is an intriguing reward.

Acknowledgments

We are grateful for input from Golan Levin and members of the Computational Design Laboratory at CMU.

References

Antipov, G.; Berrani, S.-A.; Ruchaud, N.; and Dugelay, J.-L. 2015. Learned vs. Hand-Crafted Features for Pedestrian Gender Recognition. In *Proceedings of the 23rd ACM international conference on Multimedia*, MM '15, 1263–1266.

Klingemann, M. 2018. Neural Glitch. Retrieved from http://underdestruction.com/2018/10/28/neural-glitch/.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020a. An Overview of Early Vision in InceptionV1. *Distill*.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020b. Zoom In: An Introduction to Circuits. *Distill*.

Pye, D. 1968. *The nature and art of workmanship*. University Press Cambridge.

PyTorch. Finetuning Torchvision Models. Retrieved from https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html.

RunwayML. RunwayML | Machine learning for creators. Retrieved from https://runwayml.com/.

Sharif Razavian, A.; Azizpour, H.; Sullivan, J.; and Carlsson, S. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 806–813.

Smilkov, D.; Carter, S.; Sculley, D.; Viégas, F. B.; and Wattenberg, M. 2017. Direct-Manipulation Visualization of Deep Networks. *arXiv:1708.03788*.

Zhang, Y.; Li, W.; Zhang, L.; Ning, X.; Sun, L.; and Lu, Y. 2020. AGCNN: Adaptive Gabor Convolutional Neural Networks with Receptive Fields for Vein Biometric Recognition. *Concurrency and Computation: Practice and Experience*.

Zhu, J.; Liapis, A.; Risi, S.; Bidarra, R.; and Youngblood, G. M. 2018. Explainable ai for designers: A human-centered perspective on mixed-initiative co-creation. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8.