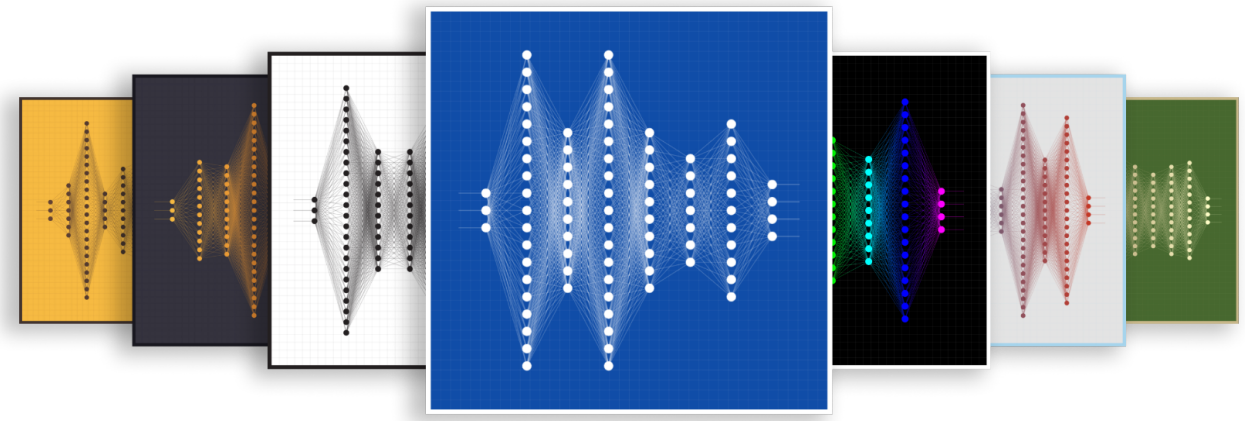
Perceptrons: AI on Bitcoin

Abstract. Perceptrons is an experimental attempt to deploy run-time AI on-chain. While many projects have attempted to store *AI artworks* (outputs from AI models) on-chain, Perceptrons attempts to store the actual *AI models* themselves (the neural networks that produce the outputs) on-chain. Not only are the models stored on-chain, but the feed-forward algorithm is also stored on-chain. Not merely a static piece of art—you can interact with Perceptrons by asking them to do *image recognition tasks*. Perceptrons permanently live on the Bitcoin network—ever evolving. They grow. They die. And finally, they're reborn again in a different form. Perceptrons are also upgradeable, i.e., you can plug a newer, smarter model into a Perceptron, which will change both the artwork and the brain *behind* the artwork. We believe this could open up an entirely new market: buying and selling data for upgradable dynamic artworks.

1. Introduction



Since the mid-1950s, AI has fascinated humans, from research labs to pop culture. And today, AI is integrating itself into the fabric of everyday life: our phones, speakers, cars, writing—the list grows daily. But how does AI actually work?

Most of us think of AI as a black box—an abstract concept understood by a handful of Silicon Valley experts. We rarely pause to think about what exactly is taking place and what is "under the hood."

Perceptrons is a collection of unique neural networks living on the Bitcoin network in perpetuity. And with this collection, we strive to achieve the following:

1. **Art.** Explain how AI works in a simple and visually striking way—through the lens of a long-form generative art collection.

2. **On-chain data.** We store the neural network models fully on-chain. There's a distinction between an AI model and its outputs. AI artworks today are *not* AI—they're simply outputs created by artists who *use* AI models. Bitcoin makes this possible by providing large on-chain storage with its Taproot upgrade.

3. **On-chain logic.** Not only are the models stored on-chain, but the feed-forward algorithm is *also* stored on-chain. We wrote a neural network from scratch in vanilla JavaScript that's part of a single HTML file stored on-chain with no external off-chain dependencies.

4. **Dynamism.** Each Perceptron is responsive to on-chain activity, such as block time and sat fees. The block time determines the age of the model, its intelligence, and its appearance. The sat fees affect how the artwork is animated.

5. **Interactivity.** People can interact with Perceptron. In fact, you can ask the neural network to perform an image recognition task.

6. **A new life form.** We also experiment with the concept of AI as another life form. Perceptrons are autonomous—they grow. They age. They're reborn. And they live one life after another forever on the Bitcoin network.

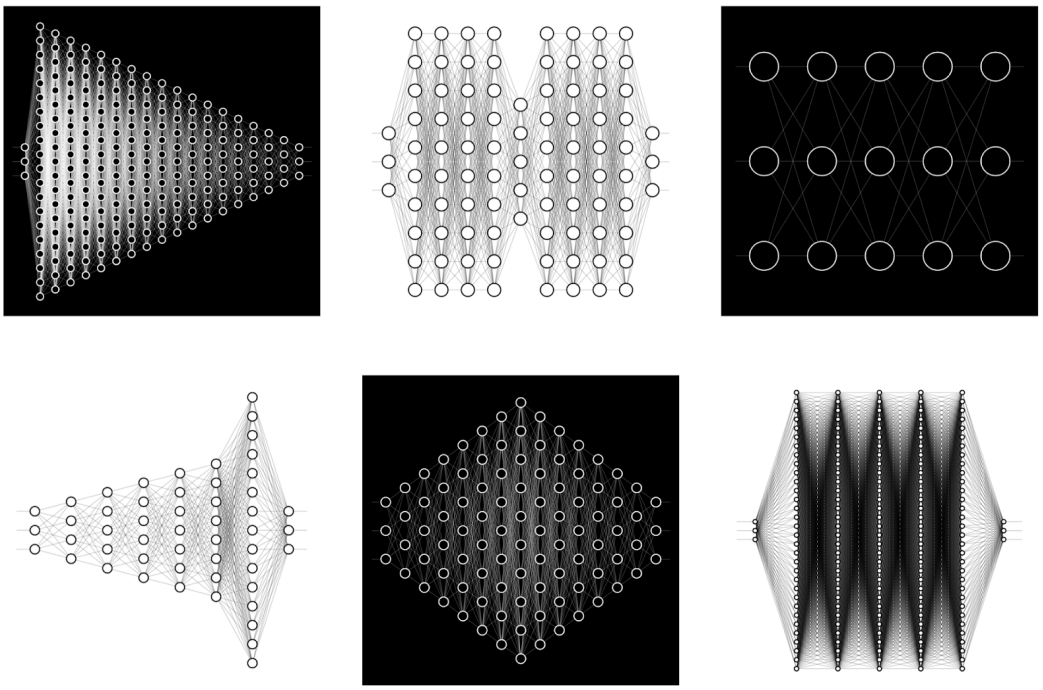
7. **Upgradable AI.** For their longevity, we program Perceptrons so their model can be plugged and played. Each Perceptron is born with a default neural network model. But you can upgrade it to a smarter, more advanced model later.

8. **Upgradable Artwork.** Since the artwork is a portrait of the neural network, once you upgrade the neural network model, the artwork will also be upgraded accordingly. Not only are you able to upgrade the brain, but you'll also see its new face.

9. **Data Markets.** We believe that our modular design approach opens up an entirely new primitive: data inscription. In the future, there could be multiple data marketplaces where collectors can buy data inscriptions to upgrade their artwork.

But to get a better idea of what's going on, let's dive into how Perceptrons works.

2. The Neural Network Architecture Generator



The first component of Perceptrons is the Neural Network Architecture Generator. The Generator programmatically creates different neural net architectures with various elements such as the number of layers, the number of neurons, and the type of activation function.

Different neural networks have different architectures, described by a raw JSON metadata file. The general structure of the file consists of 5 fields:

- ***`Model_name`***: Describe the name of the network.
- ***`layers_config`***: Describe the structure of the network. The example perceptron contains the following layers:
    - The input layer that accepts images of size 32 x 32 x 3.
    - The rescaling layer that scales input values from the range [0; 255] into range (-1; 1).
    - The flattening layer that turns 32 x 32 x 3 images into 3072-dimensional vectors.
    - 3 hidden layers with 10, 8, and 10 neurons, respectively, that use the LeakyReLU activation function.
    - The output layer that reports how much the model thinks the image belongs to each class.

- ***`weight_b64`***: The weight of all parameters of the perceptron in base64 format. More formally, suppose that there are n hidden layers. Let W_i and B_i the weight matrix and the bias vector of the i-th layer, and W_{out} the weight matrix of the output layer. We perform the following process to encode the parameters weight:
    - Flatten all the weight matrices:
      $$\widehat{W}[xc + y] = W[x, y]$$
      where c is the number of columns of the matrix
    - Concatenate all flattened weight matrices and bias vector of each layer:
      $$L = \widehat{W_1} + B_1 + \widehat{W_2} + B_2 + \ ... \ + \widehat{W_n} + B_n + \widehat{W_{out}}$$
    - Convert each element of L to float32 datatype (4 bytes), then convert L to binary data and apply base64 encoding to the binary data.

- ***`training_traits`***: The traits used to dictate the structure generation and the training process of the perceptron model.

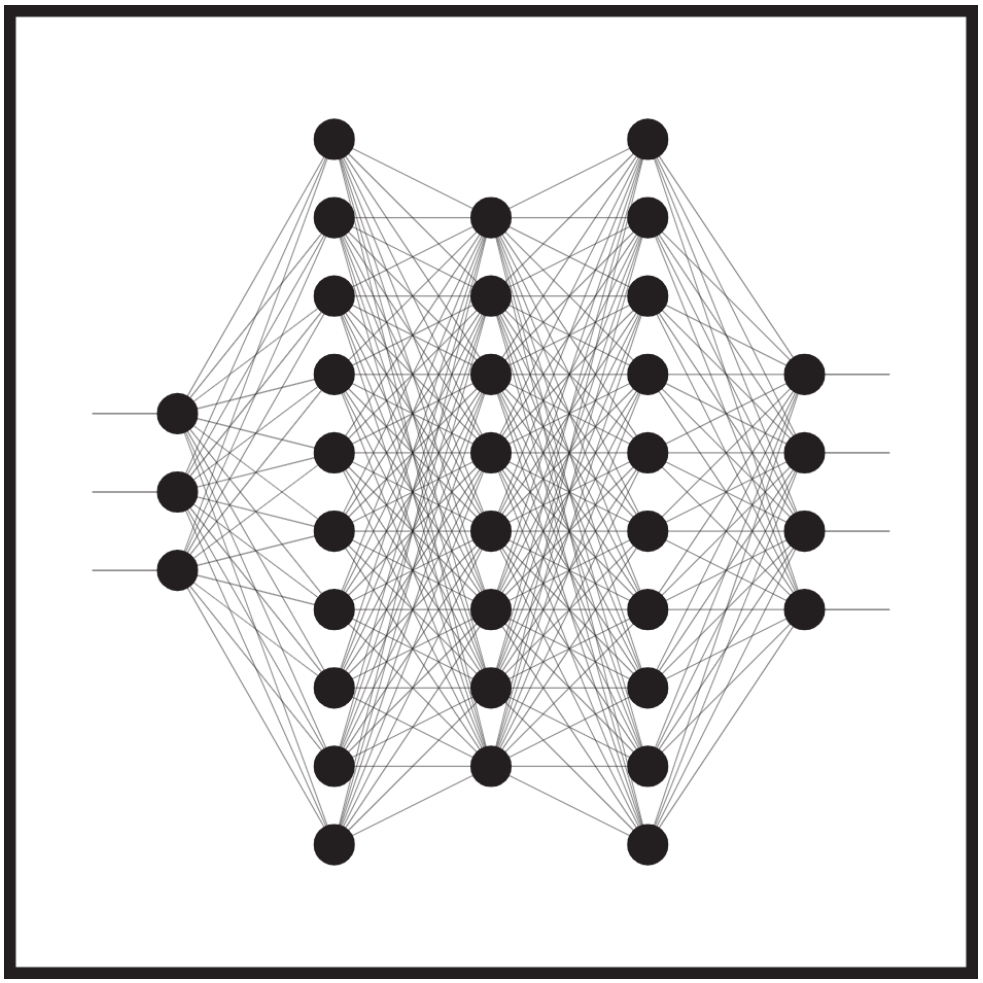
- ***`classes_name`***: The name of image classes that the model will classify.

To give you a clearer picture, here's an example of the JSON metadata file of the neural network architecture behind the artwork.

```
{
  “model_name”: “pfp_classifier_0000”,
  "layers_config": {
    "config": {
      "layers": [
        {
          "class_name": "InputLayer",
          "config": {
            "batch_input_shape": [null, 32, 32, 3]
          }
        },
        {
          "class_name": "Rescaling",
          "config": {
            "scale": 0.00784313725490196,
            "offset": -1
          }
        },
        {
          "class_name": "Flatten"
        },
        {
          "class_name": "Dense",
          "config": {
            "units": 10,
            "activation": "leaky_relu"
          }
        },
        {
          "class_name": "Dense",
          "config": {
            "units": 8,
            "activation": "leaky_relu"
          }
        },
        {
          "class_name": "Dense",
          "config": {
            "units": 10,
            "activation": "leaky_relu"
          }
        },
        {
          "class_name": "Dense",
          "config": {
            "units": 4,
            "activation": "linear"
          }
```

```
49:         }
50:       ]
51:     }
52:   },
53:   "weight_b64":
"ZFT+u05bozsi1ws9DSAVPcdzfz2N0Fi90Pb9vCzGzTxcvF09OCvCO2lUHT2QVVu8...",
54:   "training_traits": {
55:     "structure_gen": "Symmetric",
56:     "n_layers": 7,
57:     "max_nodes": 16,
58:     "activation_func": "LeakyReLU",
59:     "epoch_num": 5
60:   },
61:   "classes_name": ["Cryptoadz", "Cryptopunks", "Moonbirds", "Nouns"]
62: }
```

And here's the visualization of the above neural network architecture—a portrait of the neural network.



3. The Neural Networks

A Perceptron is a unique neural network and is programmatically generated from over 100 traits, including layers, neurons, activation functions, and various visual features.

Each Perceptron is trained, and its model weights are stored fully on-chain on Bitcoin. The feed-forward algorithm is also stored on-chain.

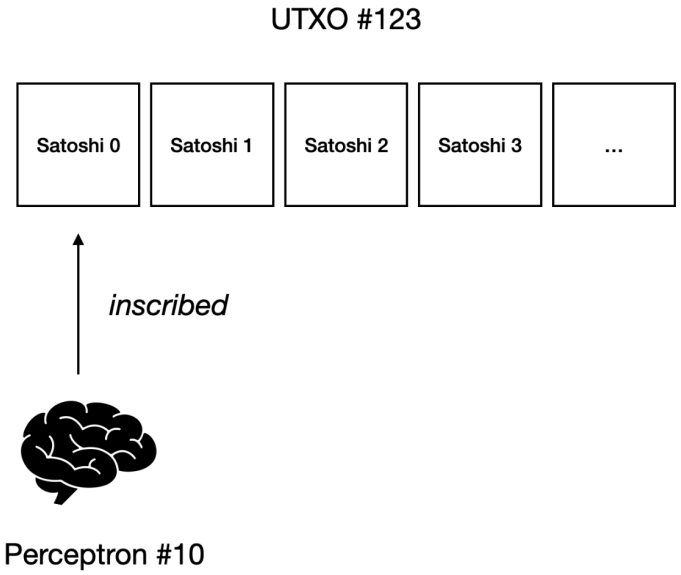
The artwork is a visual representation of the neural network architecture. Think of it as the portrait of a neural network. While most of us use neural networks as a black box today, we hope through this work, people will understand more about how neural networks operate behind the scenes.

4. 100% On-Chain

Perceptrons are inscribed as Ordinal Inscriptions on the Bitcoin network. They're fully on-chain on Bitcoin with no external off-chain dependencies. This means no IPFS, no AWS, and no Google Cloud. They live on the Bitcoin network in perpetuity. As long as there is at least *one* Bitcoin node running, Perceptrons live.

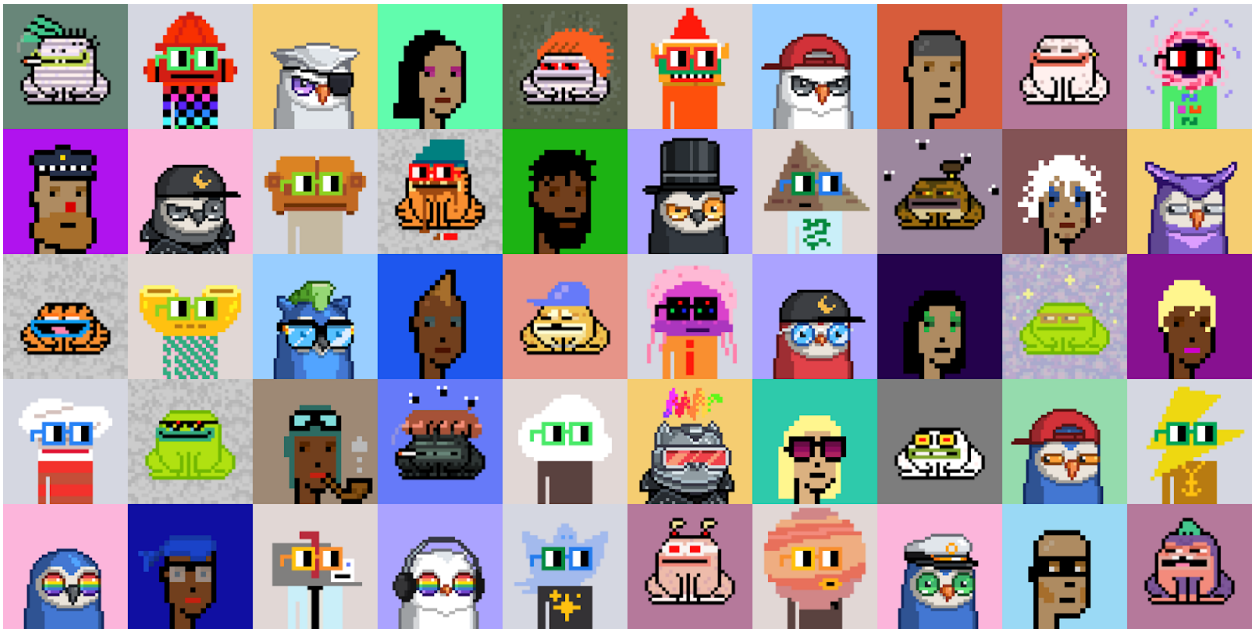
The following data is stored fully on-chain with no off-chain dependencies:

1) The model weights
2) The neural network code
3) The long-form generative artwork code
4) The interaction UI/UX code

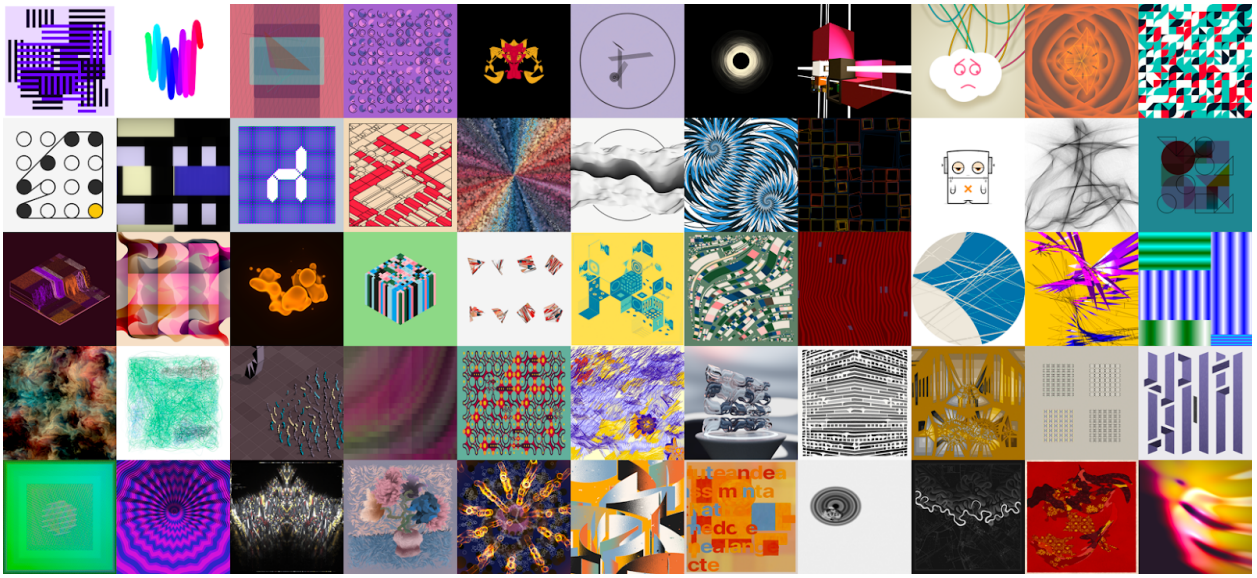


5. Datasets

Initially, perceptrons are trained on **Generative's PFP Dataset** with 37,014 raw images and another 150,000 augmented images. The raw dataset includes PFPs from four collections: CryptoPunks, Nouns, CrypToadz, and Moonbirds. We also augmented the dataset. The dataset is open-sourced on Github.



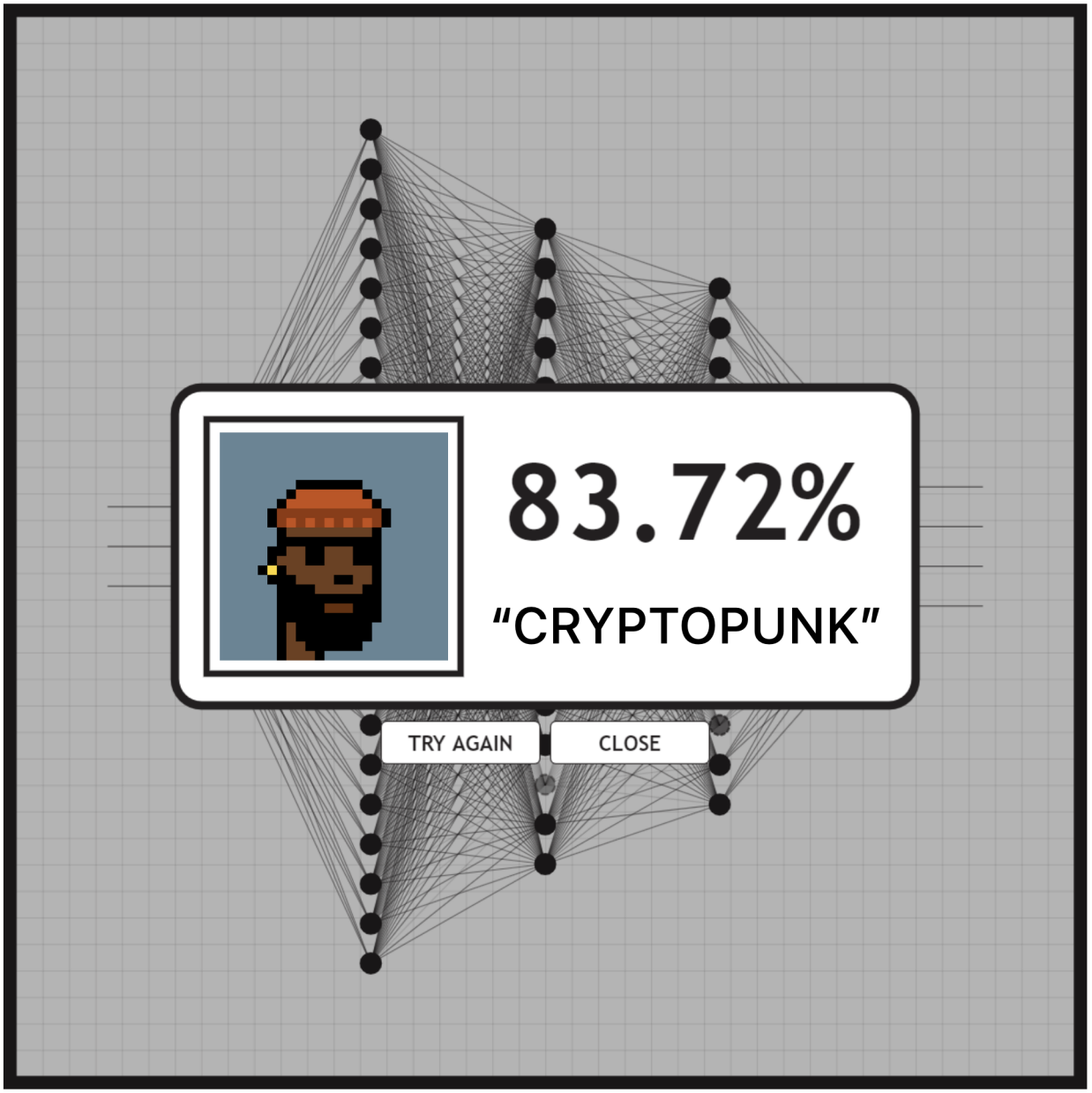
Each perceptron can be upgraded to use a smart, better model to perform more complex image recognition tasks. After the launch, we plan to work on **Generative's Art Dataset** with artworks from ArtBlocks, FxHash, and Generative.



6. Run-time and Interactive AIs

Each Perceptron is trained as a classifier. You can give it an image, and it'll answer what that image is. To make this work "in run-time" inside a browser, we wrote a neural network from scratch in Javascript. The code of the neural network is also stored on-chain.

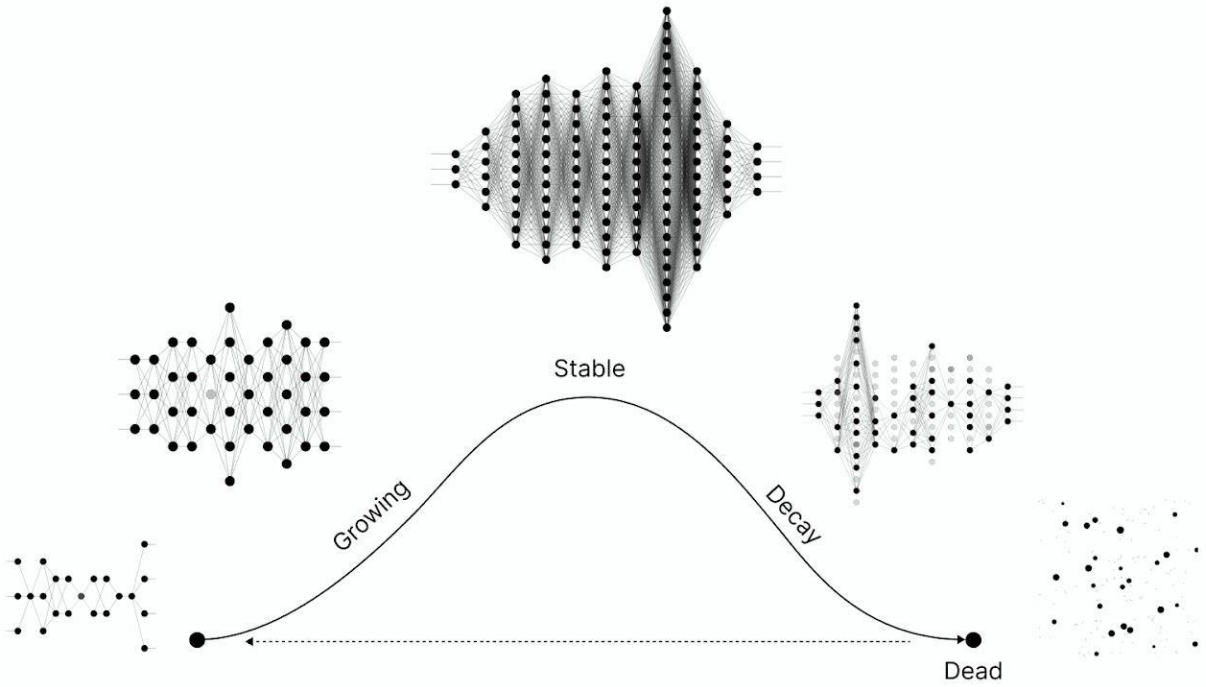
For example, in the initial model, you can input an image, and a Perceptron can classify if the image is a punk, toad, noun, or bird. In the future, you can upgrade it to a better, faster model.



7. Eternally living and evolving AI

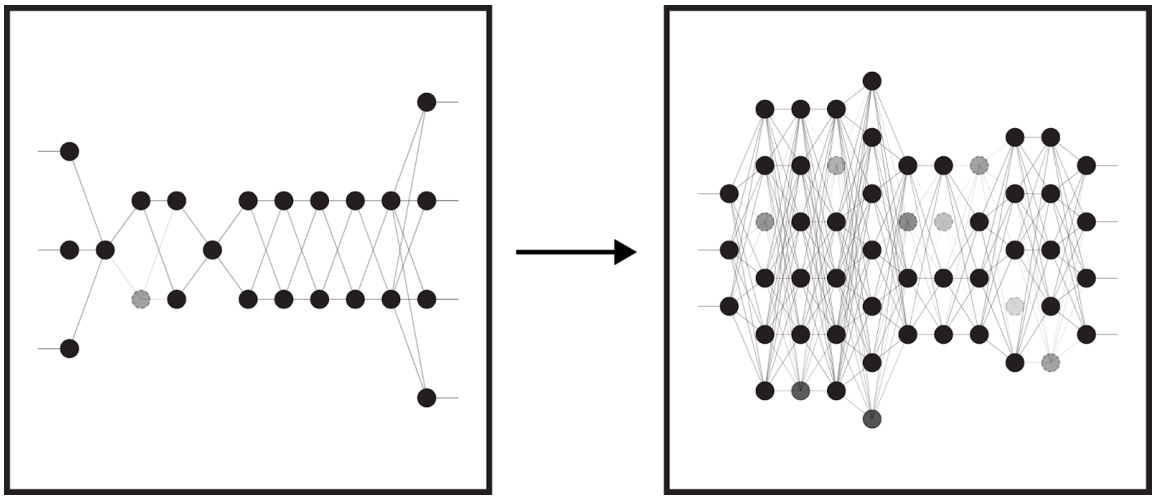
When you collect a Perceptron, you're not simply buying a long-form generative AI artwork or a rare piece of collectible on Bitcoin with historical significance. You're collecting an AI that lives forever.

Your AI permanently lives on the Bitcoin network—ever evolving. It grows. It dies. And finally, it's reborn again in a different form.

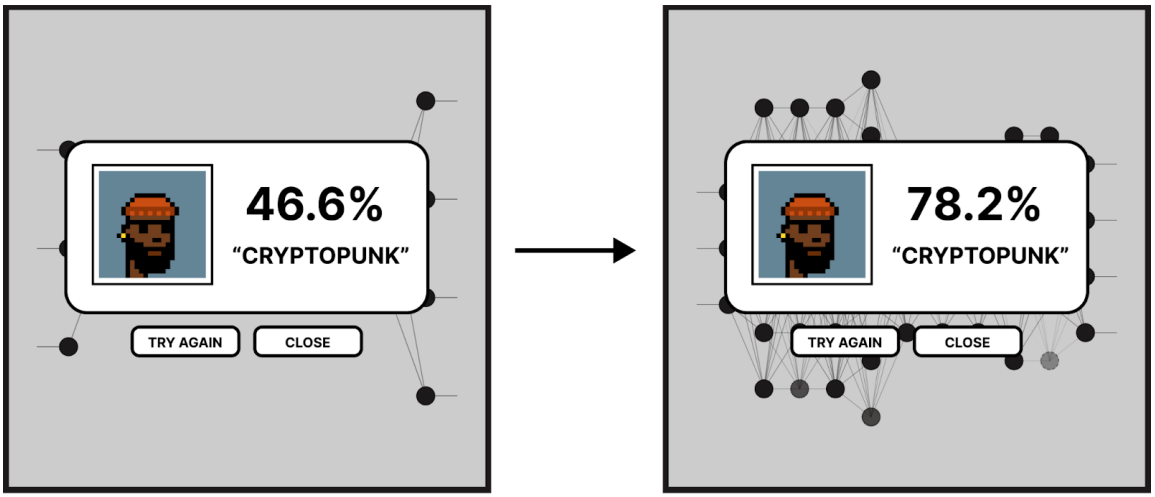


The life cycle of a perceptron

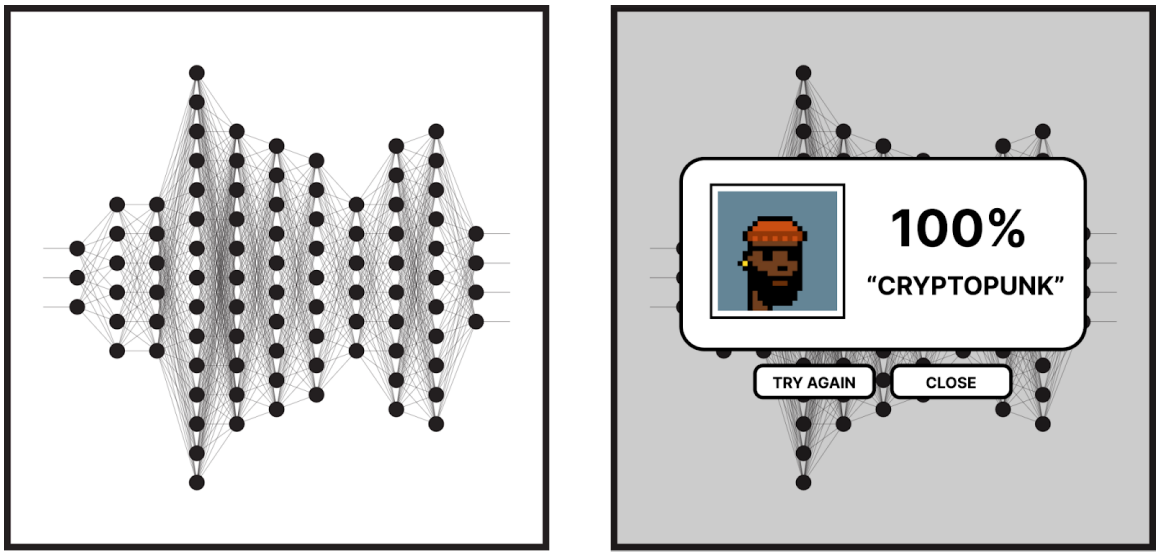
Each Perceptron starts out with a small number of neurons and then grows over time.



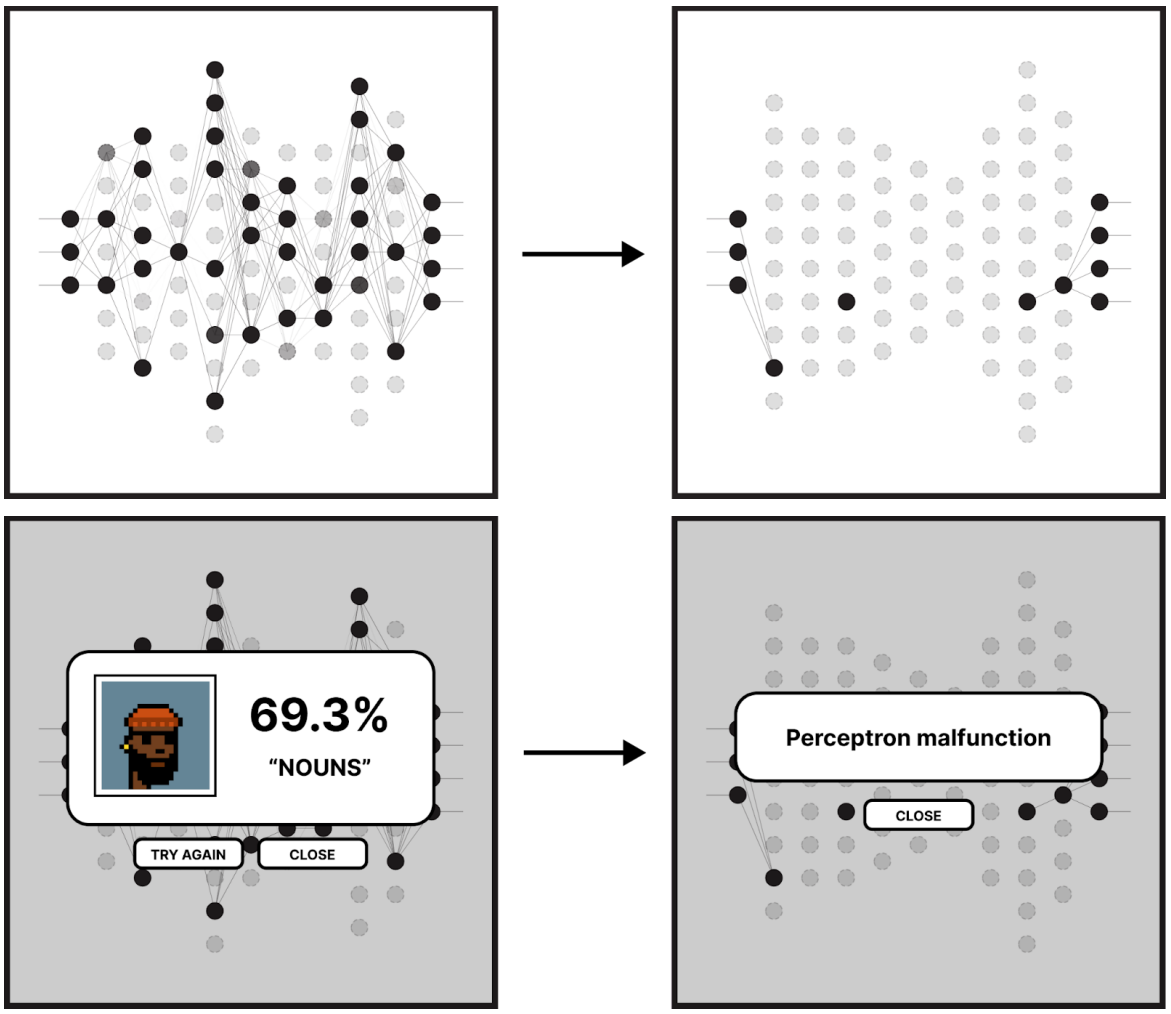
As it gains more neurons, it becomes smarter at image recognition tasks.



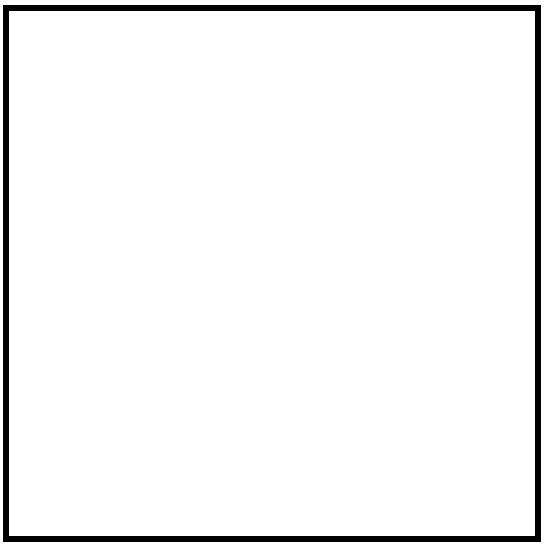
Like a human brain, once it reaches maturity, it performs at its strongest.



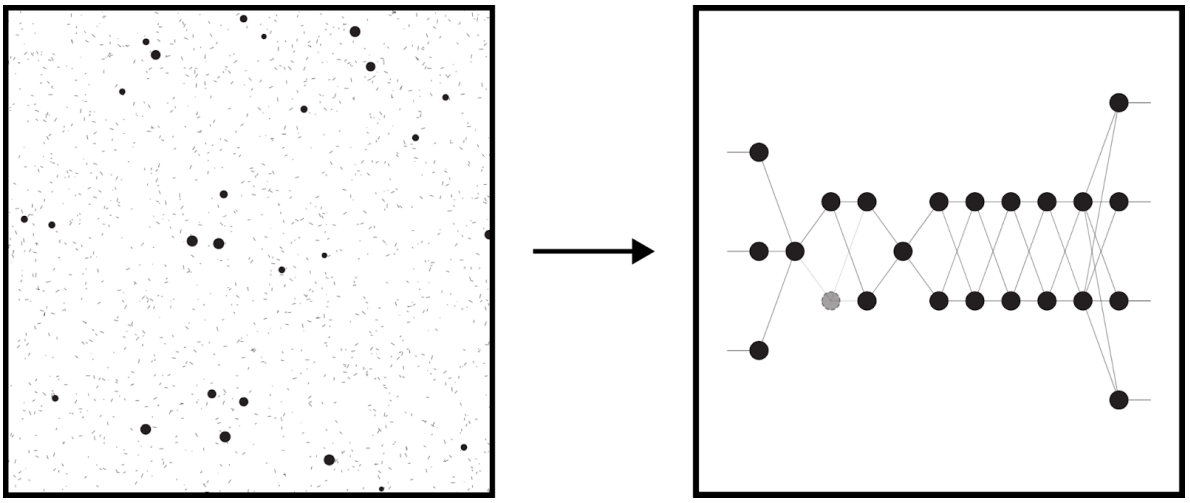
After a stable period, it enters a “decay phase.” Its neurons start to age, losing connections with one another. This leads to a decline in intelligence until there are no connections left among the neurons. And it simply stops functioning.



Then, it dies. Returning the frame to a blank slate.



But this is not the end of the story. It’s reborn and continues the endless cycle of life.



And at the micro level, each neuron also has its own different phases of life.

The life cycle of a neuron

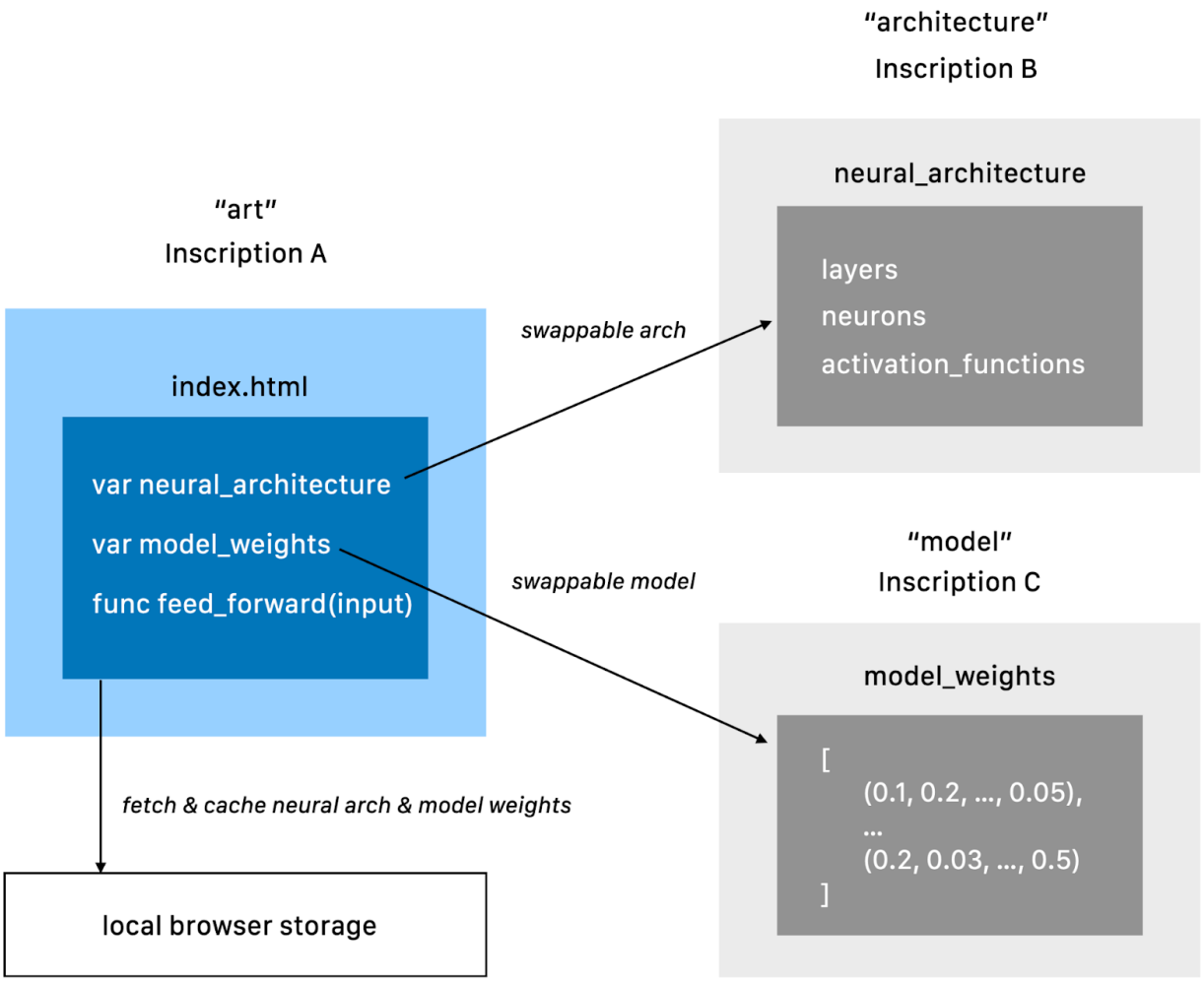
8. Upgradable AI

One important feature is that you can upgrade a Perceptron model.

Instead of using a single inscription, we design a multi-inscription architecture, separating the code for the artwork (into the “art” inscription) from the data of the neural network (into the “data” inscription).

Initially, each Perceptron comes with a default neural network model, but you can press U to update the model address. In the future, you can press U to update the neural network architecture and model weights by giving a new “data” inscription address.

This element will bring longevity to the artwork.


"architecture"
Inscription B
neural_architecture
layers
neurons
activation_functions
"art"
Inscription A
swappable arch
index.html
var neural_architecture
var model_weights
func feed_forward(input)
swappable model
"model"
Inscription C
model_weights
[
(0.1, 0.2, ..., 0.05),
...
(0.2, 0.03, ..., 0.5)
]
fetch & cache neural arch & model weights
local browser storage

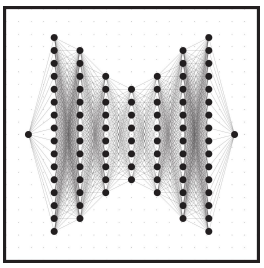
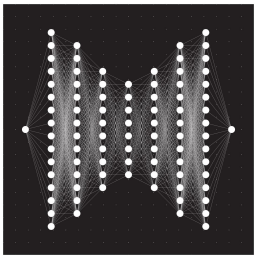
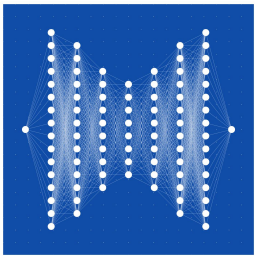
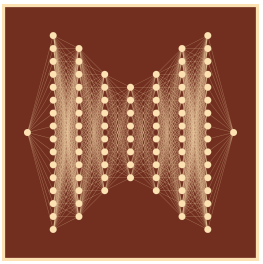
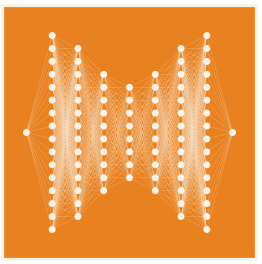
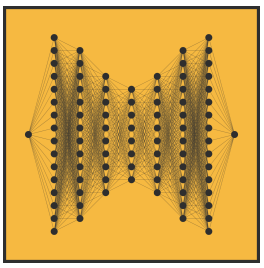
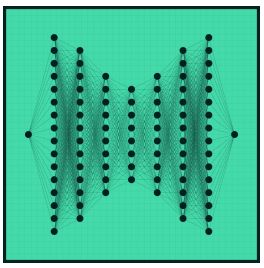
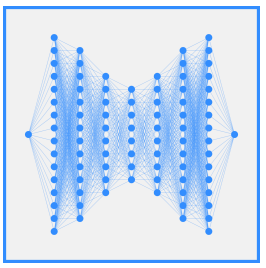
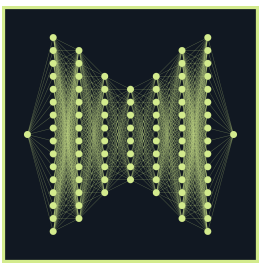
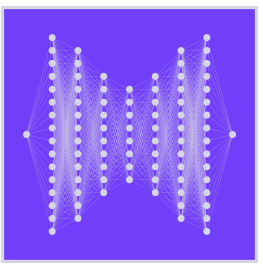
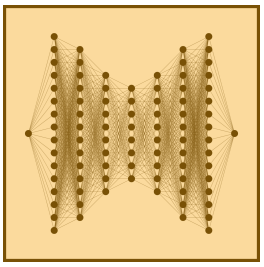
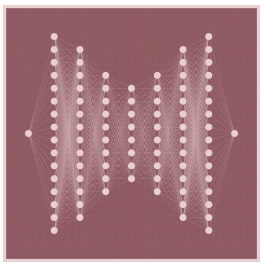
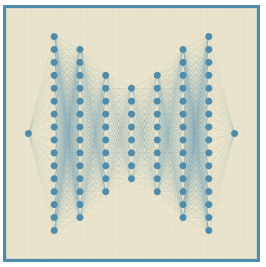
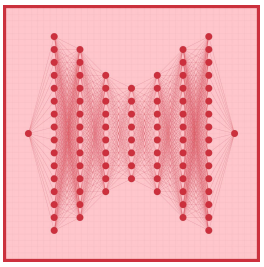
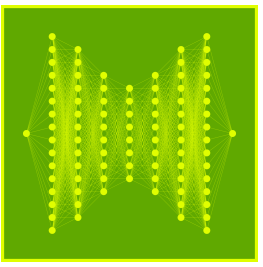
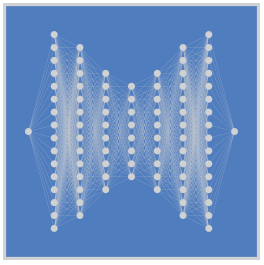
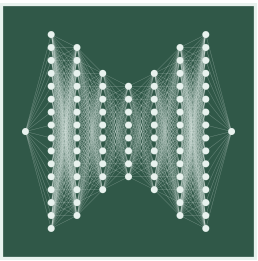
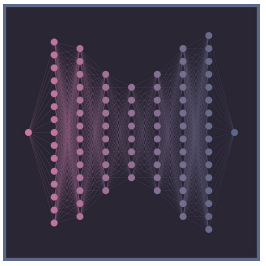
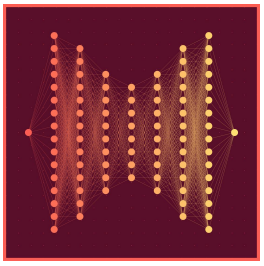
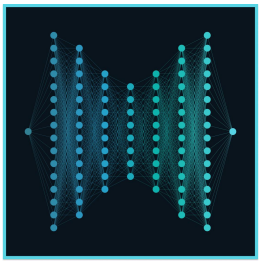
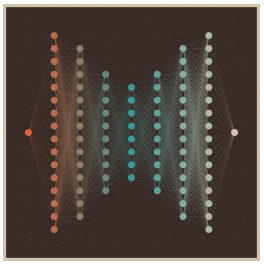
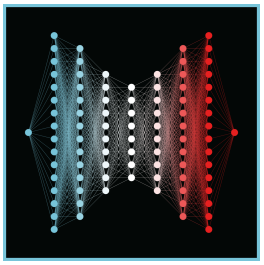
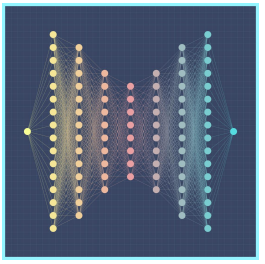
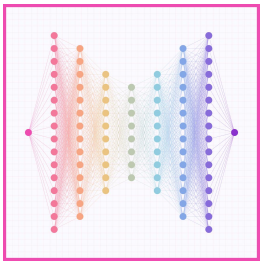
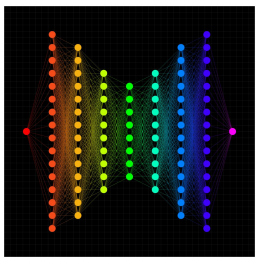
9. Data Markets for Upgradable Artworks

Our modular design approach opens up a new primitive: data inscription.

In the future, there could be multiple data marketplaces where collectors can buy data inscriptions to upgrade their artwork, and AI/ML engineers can train neural network models, package them as data inscriptions, and sell them.

This approach will work with other artworks as we generalize the data set beyond neural network models. This includes data such as stock pricing, blockchain, and weather.

10. Color Palettes

 Whitepaper (2%)	 Blackboard (2%)	 Blueprint (2%)	 Nak (5.2%)	 Jims (5.2%)
 Level 10 (5.2%)	 Flips (5.2%)	 Level 14 (5.2%)	 III (5.2%)	 XMB (5.2%)
 Info (5.2%)	 Adventure (5.2%)	 Marigold (5.2%)	 Phoenix (5.2%)	 Love (5.2%)
 Cachet (5.2%)	 Human (5.2%)	 Twilight (3%)	 Sunset (3%)	 Aurora (3%)
 Liminal Space (3%)	 Déjà Vu (3%)	 Lucid Dream (3%)	 Parallel (1%)	 Multiverse (1%)

11. Features

Here are the key neural network traits.

Number of hidden layers	1 - 4 (Sigmoid) / 1 - 10 (Other function)
Number of maximum neurons per hidden layer	5 - 20
Network architecture	Random Triangle (50%), Random (20%), Regular Symmetric (10%), Zig Zag (10%), Triangle (5%), Rectangle (5%)
Activation function	ReLU (50%), LeakyReLU (20%), tanh (20%), Sigmoid (10%)
Number of training epochs	8 - 12 (Sigmoid) / 4 - 6 (Other function)
Dataset (Fill mode)	MNIST (70%), CIFAR (25%), IMAGENET (5%)
Deep learning framework (Shape)	Theano (60%), Torch (30%), TensorFlow (10%)
Hardware acceleration (animation speed)	Basic (30%), Standard (60%), Advanced (10%)
Paper pattern	Plain (20%), Dotted (40%), Squared (40%)
Color palette	See above
Birth Year	See below

Birth year is a special trait. It's used to calculate the age of the Perceptron, which influences how intelligent the Perceptron is at each point in time. The following chart outlines the history of AI from its inception until today.

Year (Rarity)	Description
1943 (1%)	First artificial neuron model was proposed by Warren McCulloch and Walter Pitts as an attempt to model biological neurons mathematically.
1951 (1.5%)	SNARC — the first neural network machine that can learn, was built by Marvin Minsky and Dean Edmonds.
1957 (2%)	Single layer perceptron was invented by Frank Rosenblatt.
1969 (2.5%)	Limitation of the perceptron model is outlined in the book "Perceptrons" by Marvin Minsky and Seymour Papert, which led to a decline in AI research for the next 10 years.
1970 (3%)	Automatic differentiation method was published by Seppo Linnainmaa.

1980 (3.5%)	Neocognitron—the first Convolution Neural Network model, was invented by Kunihiko Fukushima.
1982 (4%)	Hopfield Network—proposed by Jon Hopfield—sparked the interest in NN research again and later inspired recurrent neural network research.
1986 (4.5%)	The term “Backpropagation” was coined by David Rumelhart, Geoff Hinton, and Ronald J. Williams in their application of Seppo Linnainmaa's work.
1988 (5%)	Universal approximation theorem (stating that a feedforward network with at least one hidden layer can approximate any function) was proved true by Kurt Hornik.
1997 (5.5%)	Long short-term memory (LSTM) recurrent neural networks were proposed by Sepp Hochreiter and Jürgen Schmidhuber.
1998 (6%)	The first image recognition system (LeNet-1) was proposed by Yann Lecun, which is a CNN trained using backpropagation. Release of MNIST (hand-written digit) dataset.
2002 (6.5%)	Torch, a software library for machine learning, was released.
2009 (7%)	Release of ImageNet dataset (a large dataset of real-world images). Since then, the ImageNet Large Scale Visual Recognition Challenge has been hosted annually.
2012 (7.5%)	AlexNet was released and won the 2012 ImageNet Large Scale Visual Recognition Challenge—marking the start of the deep learning era. Google Brain released The Cat Experiment.
2014 (8%)	Generative Adversarial Networks (GAN) were introduced by Ian Goodfellow.
2015 (8.5%)	TensorFlow was released by the Google Brain team.
2016 (9%)	Google’s AlphaGo program became the first computer program to beat Lee Sedol.
2023 (15%)	Generative Brains were released on the Bitcoin network

There are also two other attributes that indicate the current state of Perceptrons.

Life Cycle	60 Years (3%), 60 Months (40%), 60 Weeks (37%), 60 Days (20%)
State	Growing, stable, decaying, dead, rebirth

12. Summation

We've introduced a new form of artwork that is on-chain, dynamic, interactive, upgradeable, and—most importantly—intelligent. The first component is a Neural Network Architecture Generator, which programmatically generates neural network architectures that complete image recognition tasks. We then train these neural networks with an initial dataset—the Generative's PFP. These artworks are autonomous—they grow. They age. They're reborn. And they live one life after another—forever—on the Bitcoin network. These neural networks are programmed to be upgradable, so we can feed them smarter, *better* models so they can perform more complex tasks. We believe this could also open up an entirely new market: buying and selling data for upgradable dynamic artworks.

Appendix A: Open Source

At Generative, we believe that open source is the best way to make a contribution to the community at large. Ideally, we'd like to open-source everything we build. And for this endeavor, we're releasing the following source code for the following:

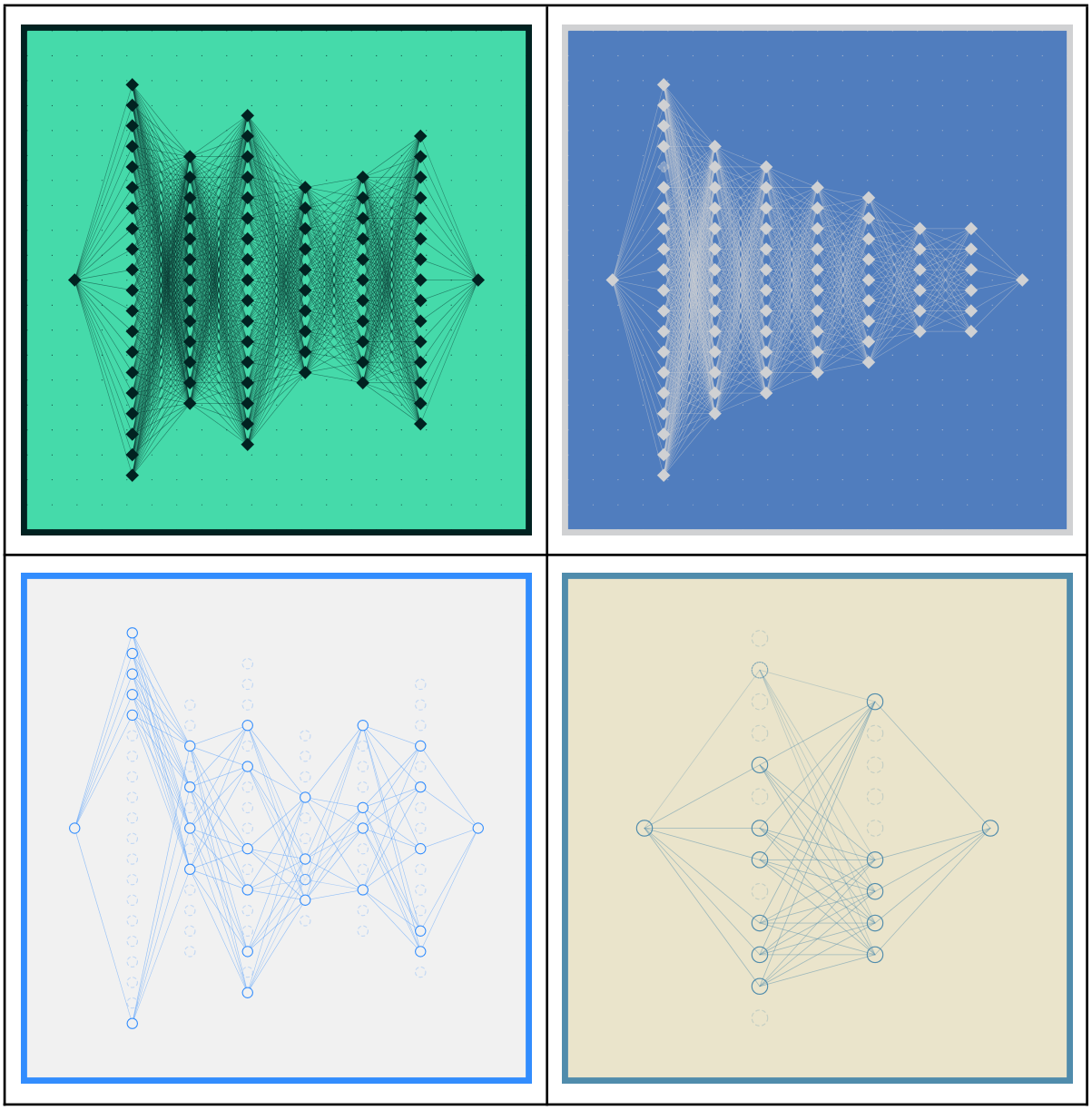
- Long-form generative artwork
- Javascript neural network

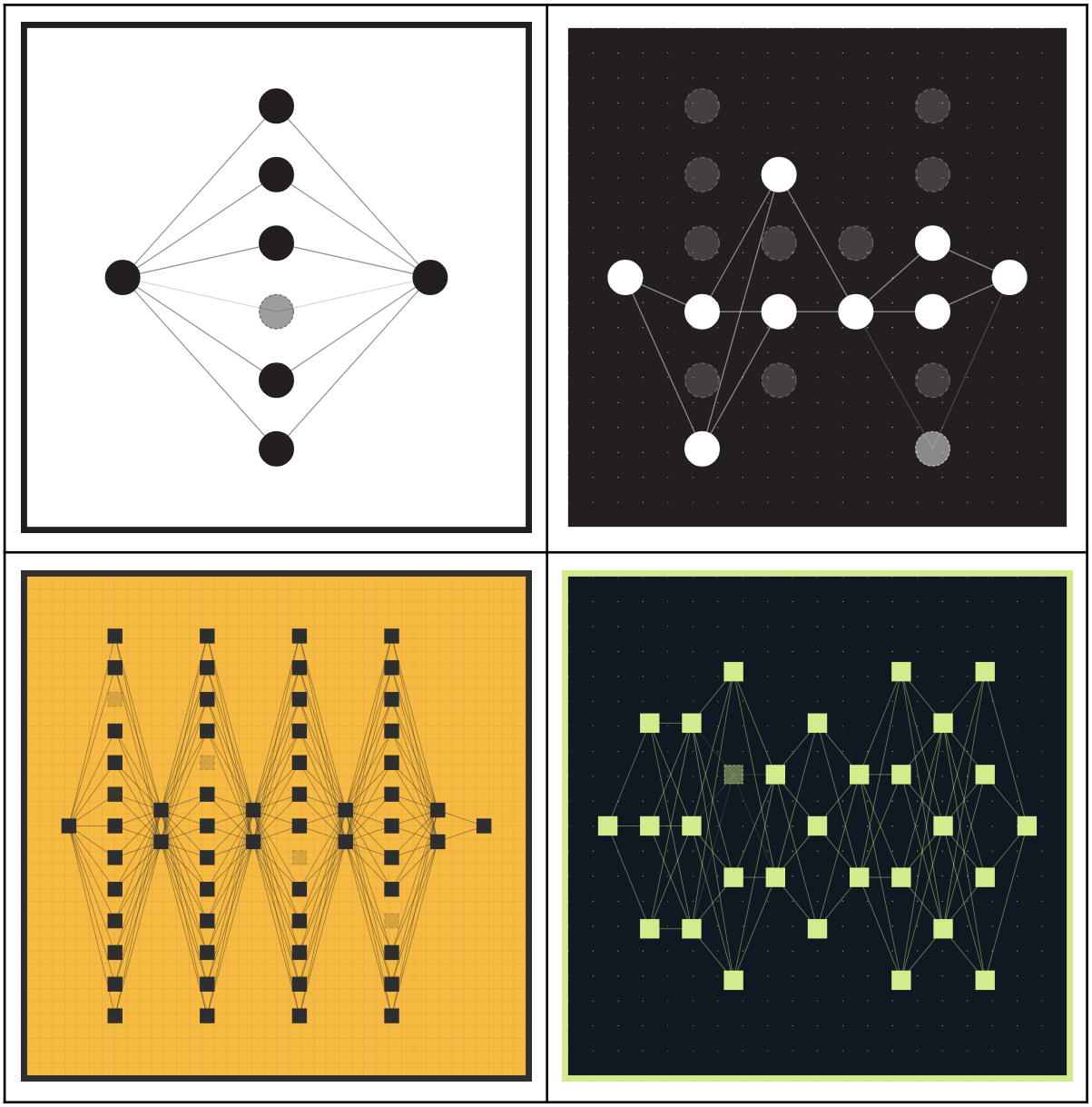
- Python neural network training
- Datasets

The code is published at https://github.com/generative-xyz.

Appendix B: Sample Outputs

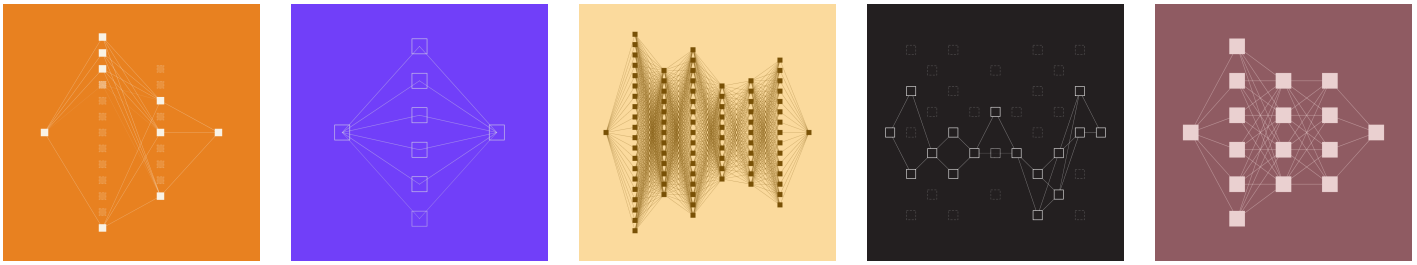
Here are some of the sample outputs from the long-form generative code.



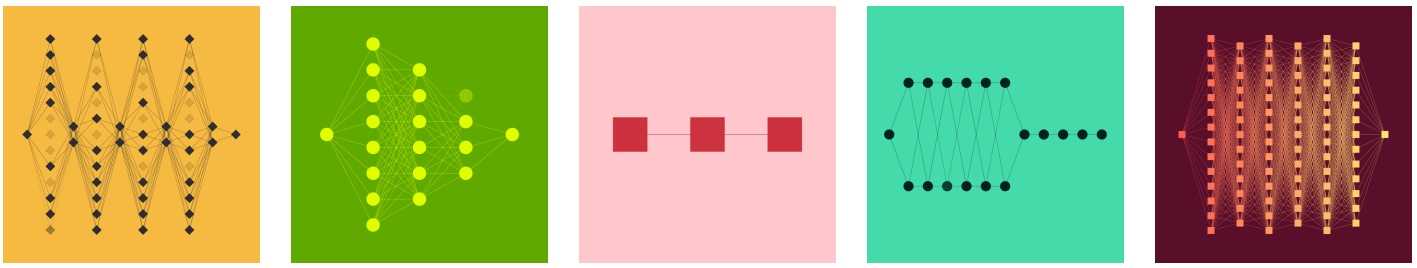


Appendix C: How to collect Perceptrons

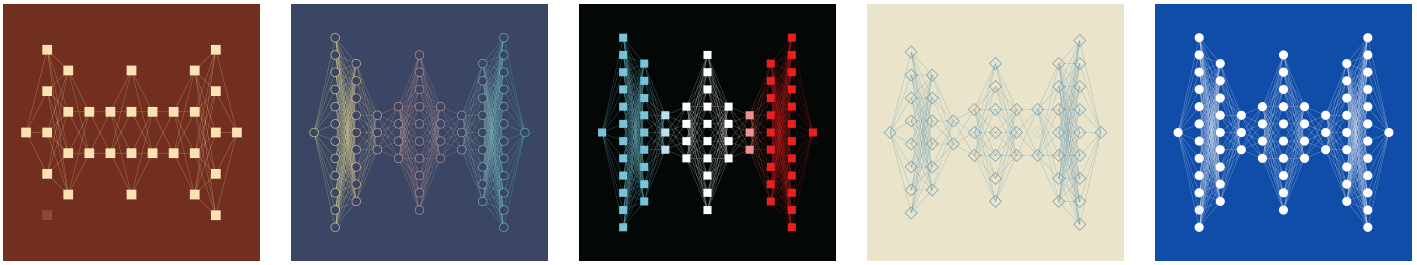
Collect by Deep Learning Framework - Node Shape (Example: Torch Framework - Square)



Collect by Data Set - Node fill mode (Example: MNIST Data Set - Solid)

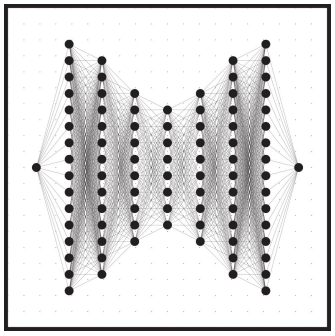


Collect by Network Architecture (Example: Symmetric Architecture)

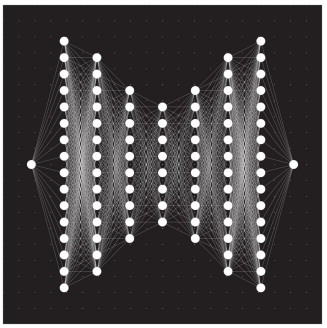


Collect by Color Palette

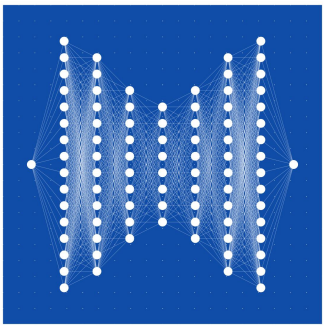
Technical palette



Whitepaper

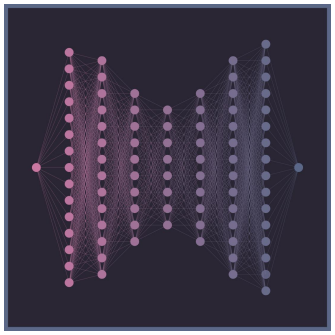


Blackboard

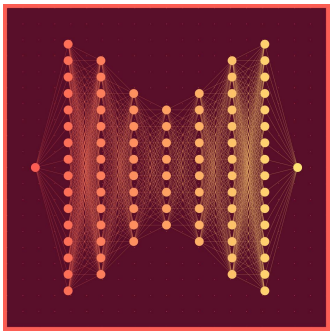


Blueprint

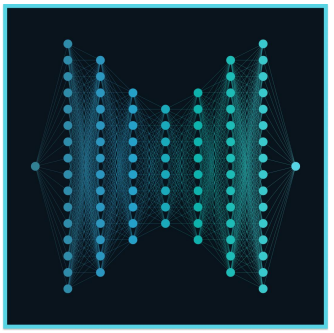
Sky palette



Twilight

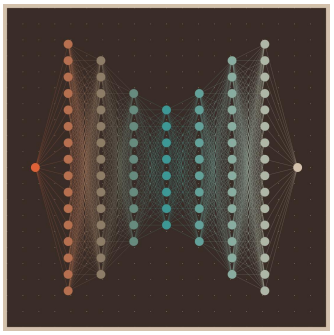


Sunset

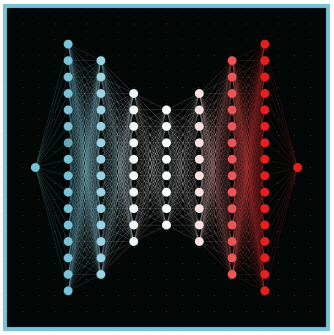


Aurora

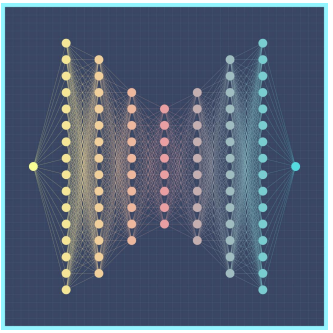
Liminality palette



Liminal Space

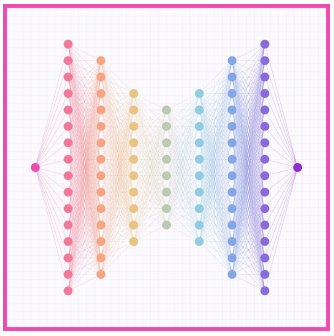


Déjà Vu

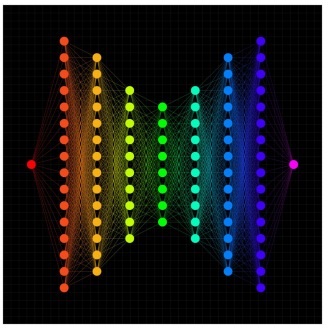


Lucid Dream

Parallel Multiverse palette



Parallel



Multiverse