



EXPLORING THE COLLABORATION BETWEEN
PROCEDURALISM & DEEP LEARNING

PRESENTATION BY ANASTASIA OPARA



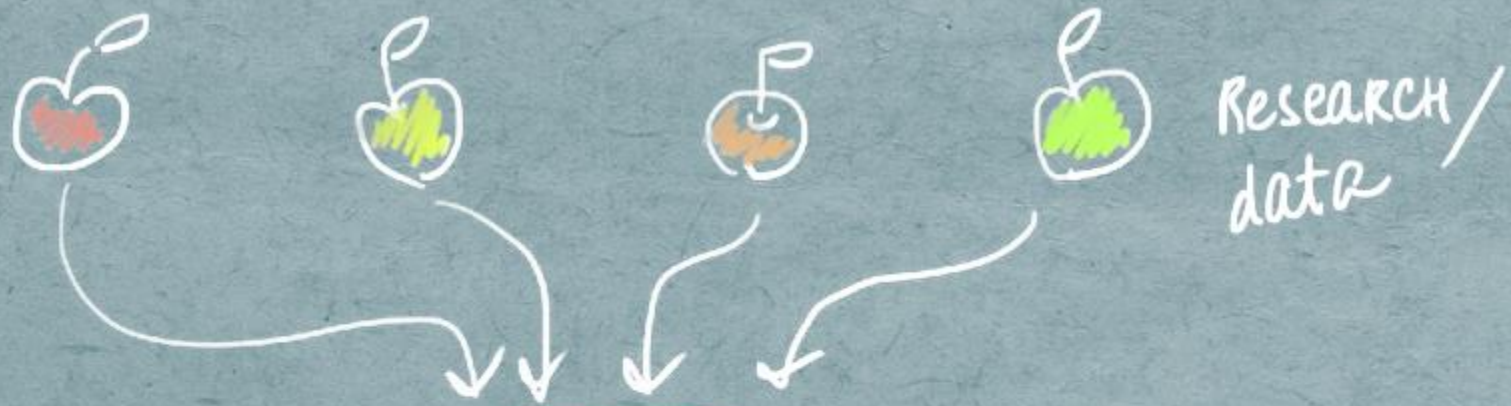
SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION

Massive Scale



Single Asset Scale





YOUR AMAZING
brain



"Apple juice"
of your data



algorithmic
Representation / model

creativity / craft ratio

creativity

craft



creativity / craft ratio

creativity

craft



creativity / craft ratio

creativity

craft



creativity / craft ratio

creativity

craft



minimize!



creativity / craft ratio

creativity

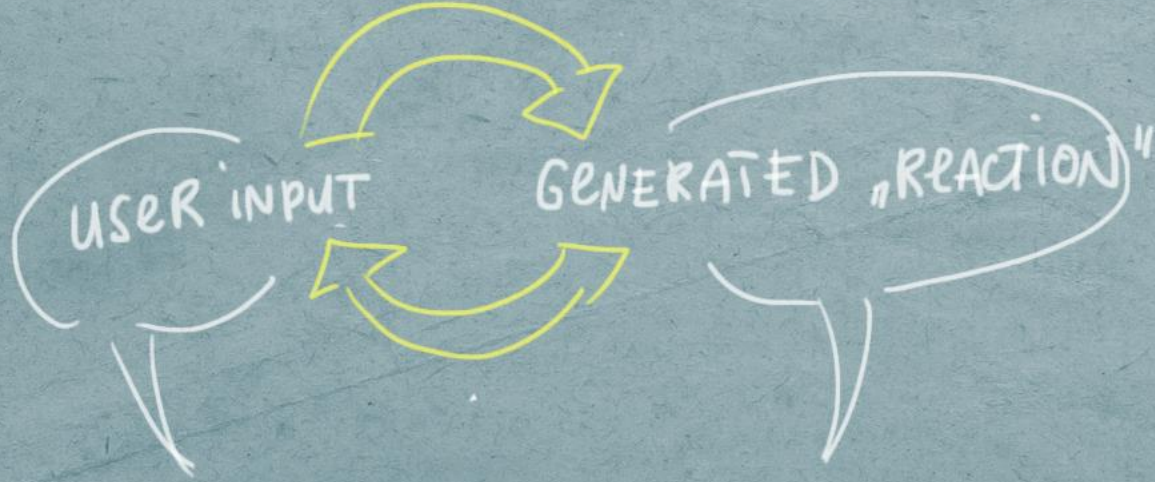
craft

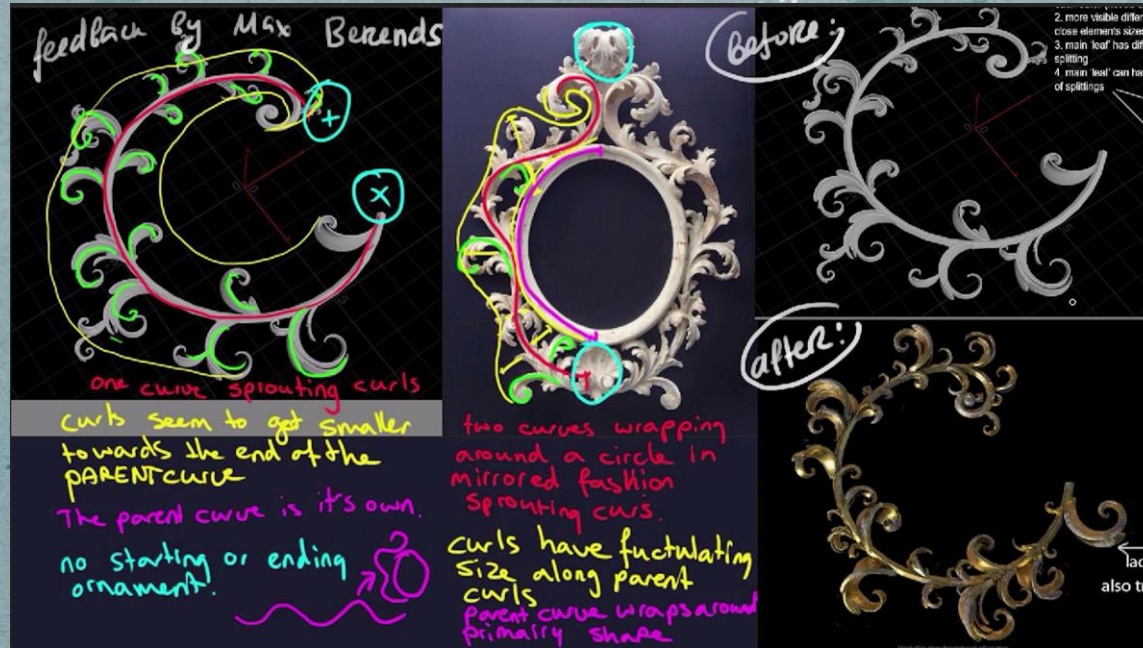


minimize!



"CONVERSATION"





*Quick!
To the hype-mobile!*



I love every
moment of pain
you bring me~

A



B



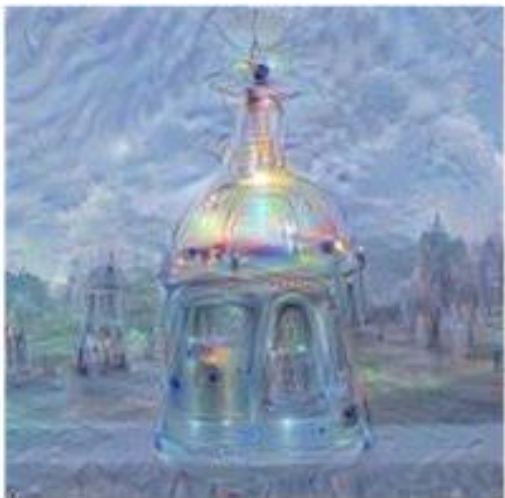
“Image Style Transfer Using Convolutional Neural Networks,” Gatys et al. 2016.

“Artistic style transfer for videos”,
Manuel Ruder, Alexey Dosovitskiy and Thomas Brox, 2016.

<https://www.youtube.com/watch?v=Khuj4ASldmU>



Horizon



Towers & Pagodas



Trees



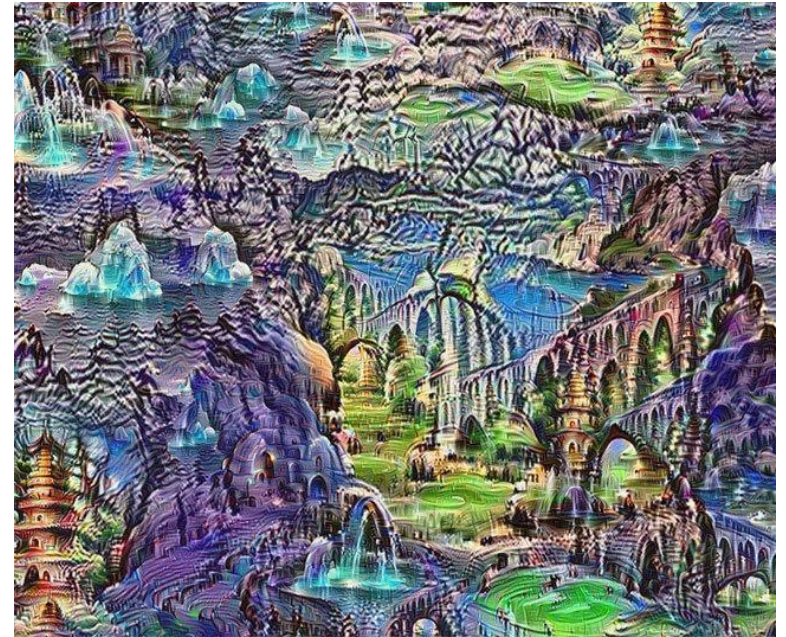
Buildings



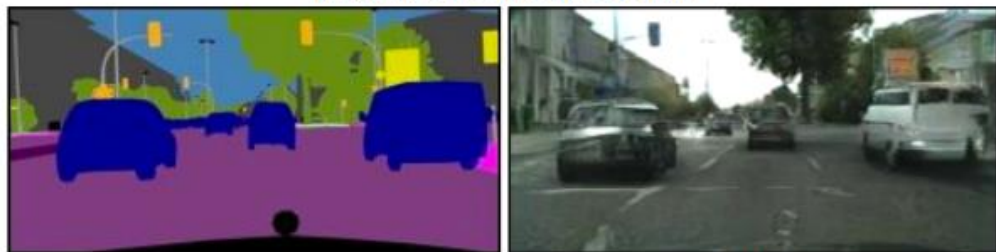
Leaves



Birds & Insects



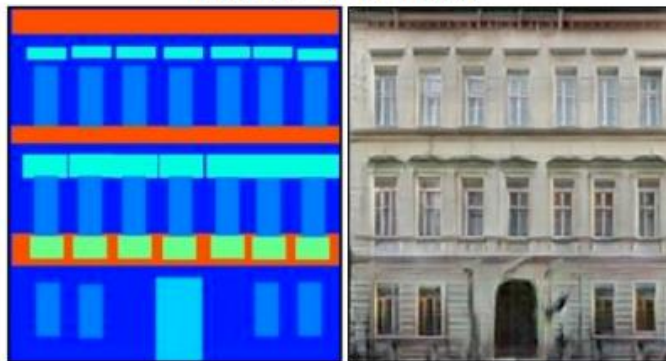
Labels to Street Scene



input

output

Labels to Facade



input

output

BW to Color



input

output

Aerial to Map



input

output

Day to Night



input

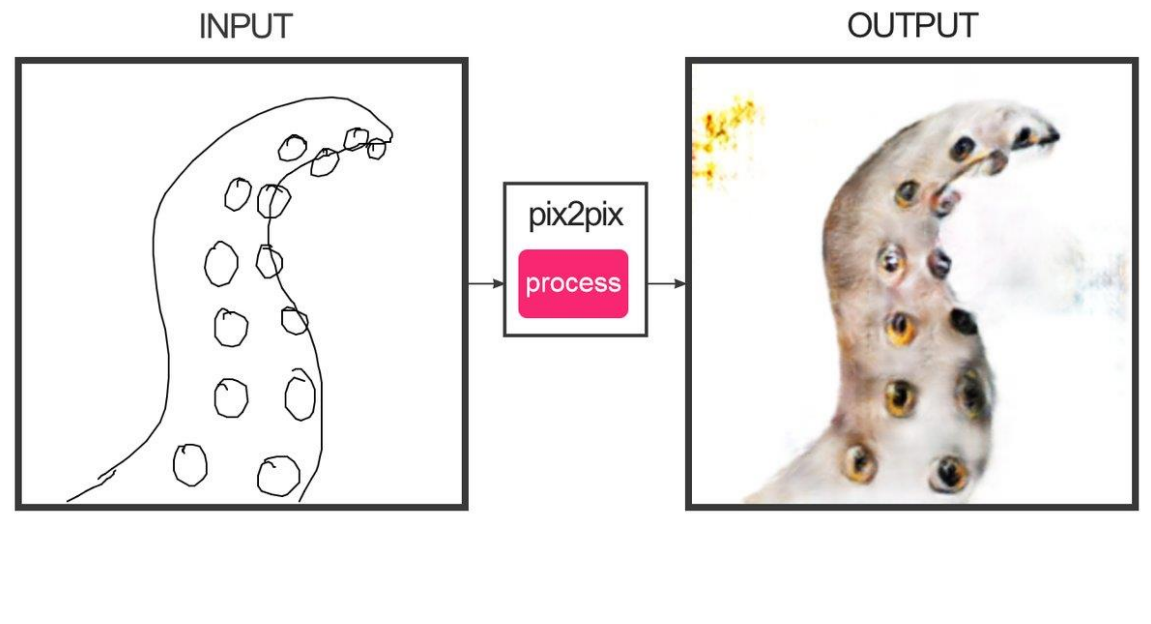
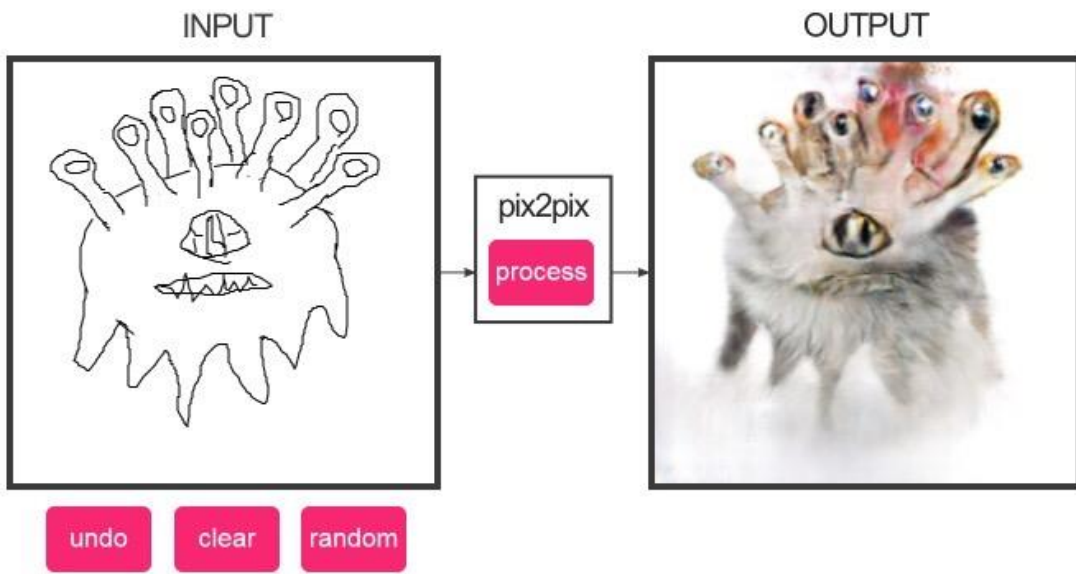
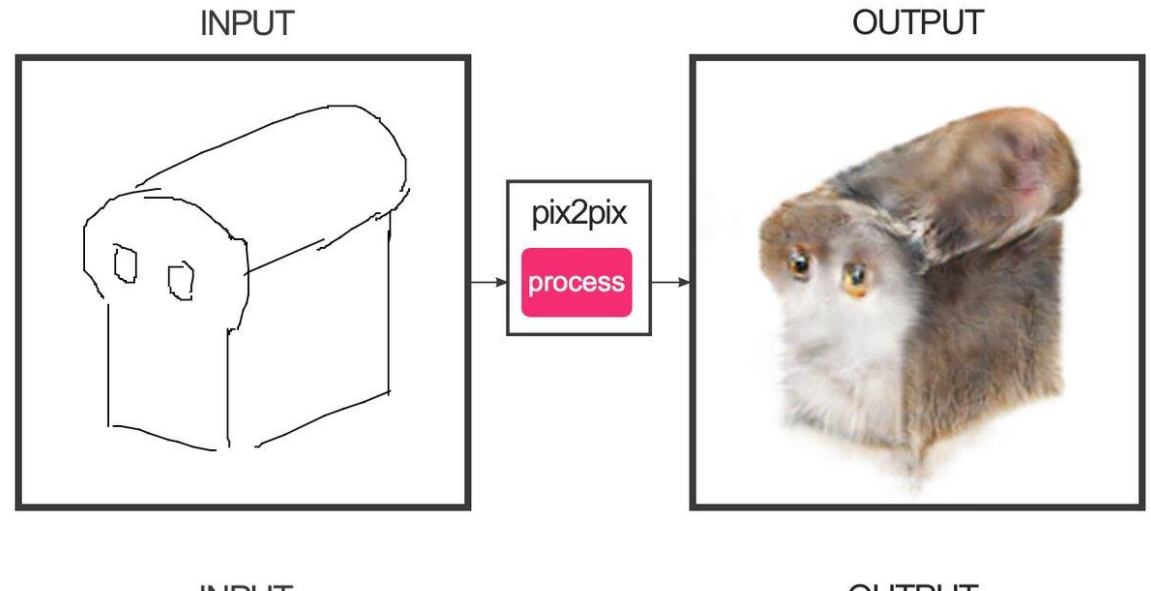
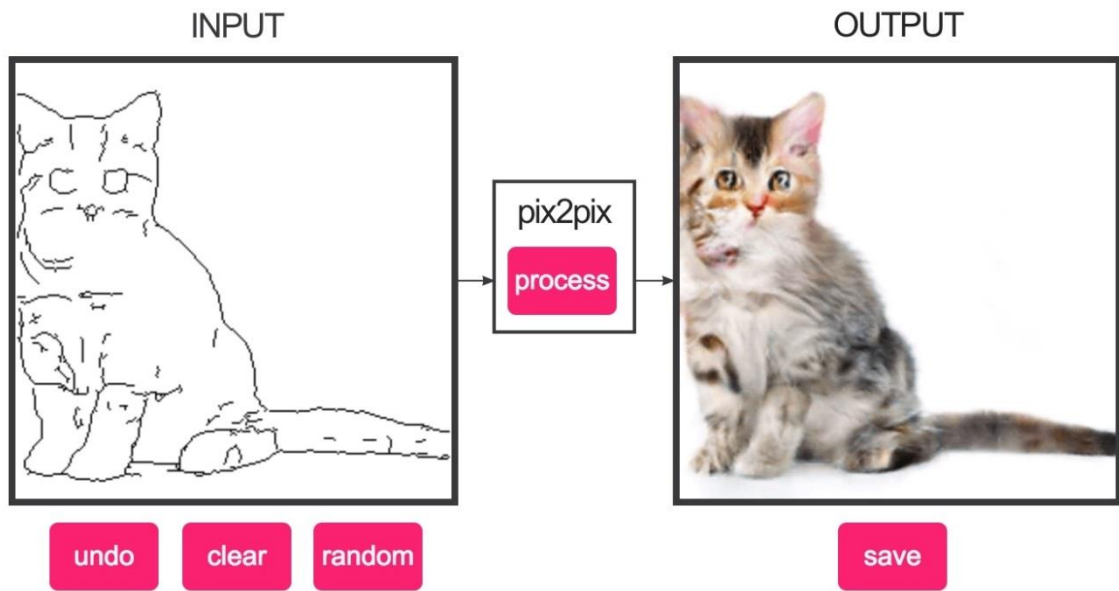
output

Edges to Photo



input

output



“Phase-Functioned Neural Networks for Character Control”,
Daniel Holden et al., 2017

<https://www.youtube.com/watch?v=UI0Gilv5wvY>

Experimental Self-Learning AI in Battlefield 1

<https://www.youtube.com/watch?v=ZZsSx6kAi6Y>

SEED – Project PICA PICA

<https://www.youtube.com/watch?v=LXo0WdIElJk>



$$Y = A(\underbrace{wx}_{\text{linear function}} + b)$$

activation



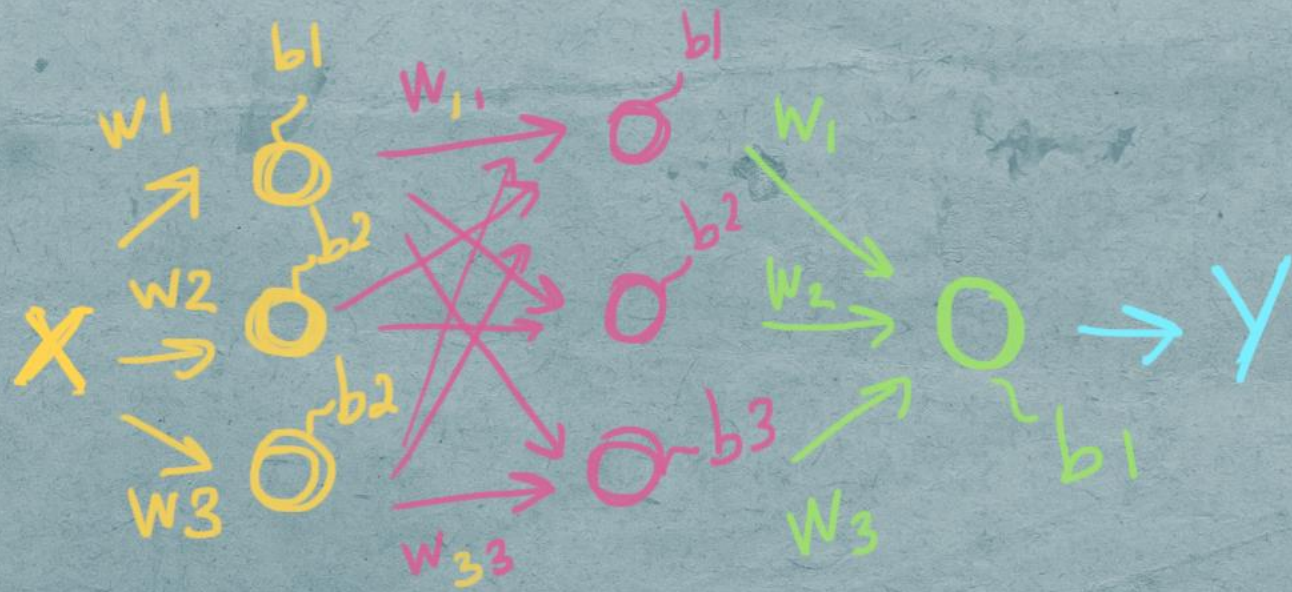
linear function!

$$f(x) = ax + b$$



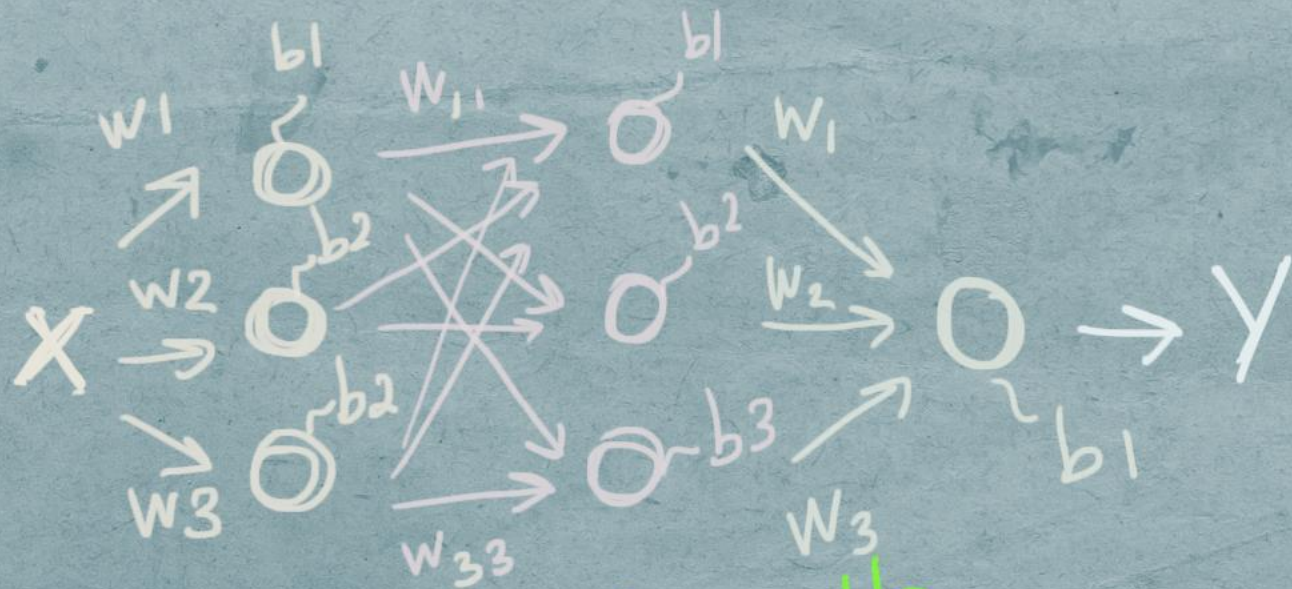


$$Y = A(w + A(w + A(wx + b) + b) + b)$$



$$Y = A(W + A(W + A(WX + b) + b) + b)$$

$[w_1, w_2, w_3]$
 $\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$
 $\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$
 $\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$
 $\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$
 $[b_1]$

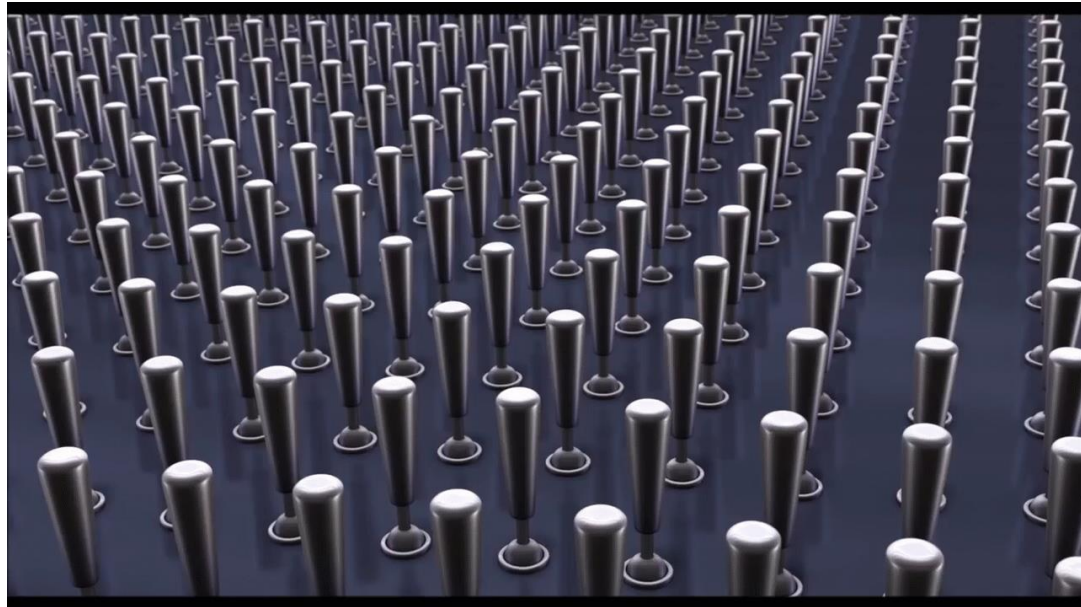
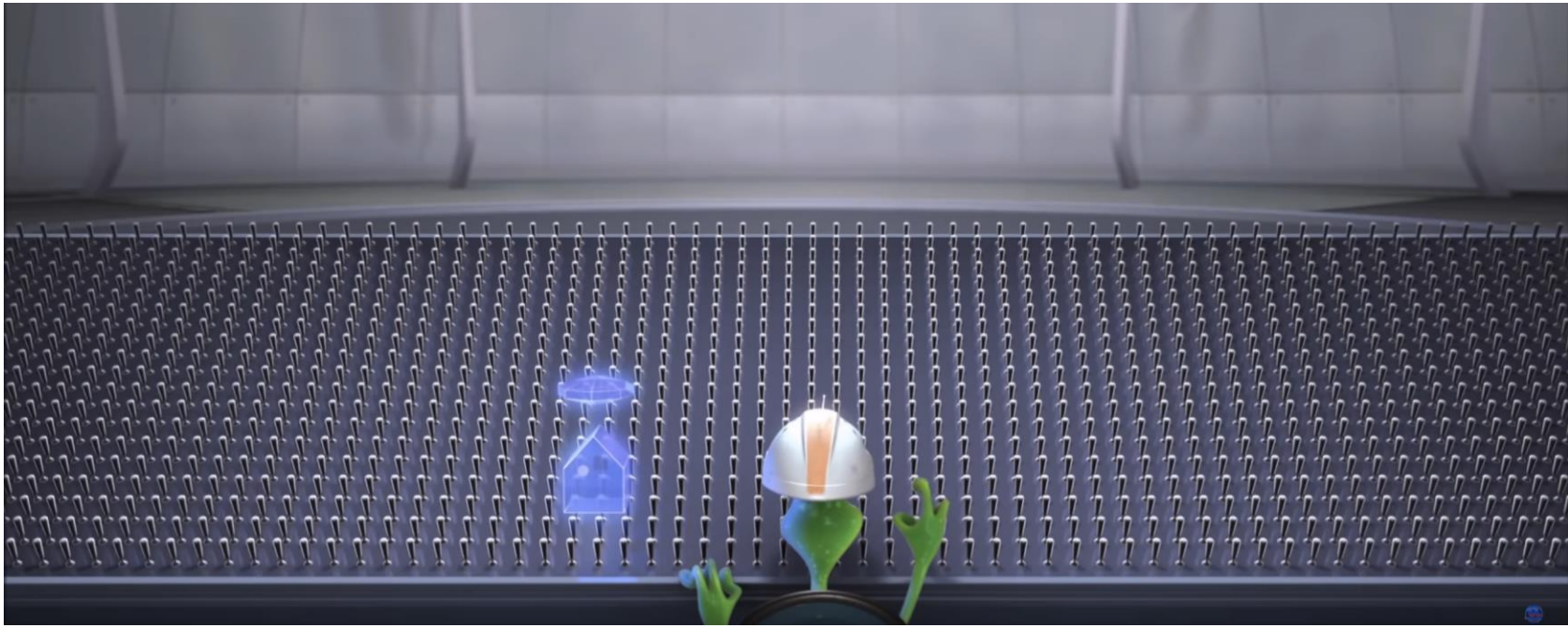


$$Y = A(W + A(W + A(WX + b) + b) + b)$$

learnable

$$\begin{matrix}
 [w_1, w_2, w_3] & \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} & \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} & \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} & \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} & [b_1]
 \end{matrix}$$

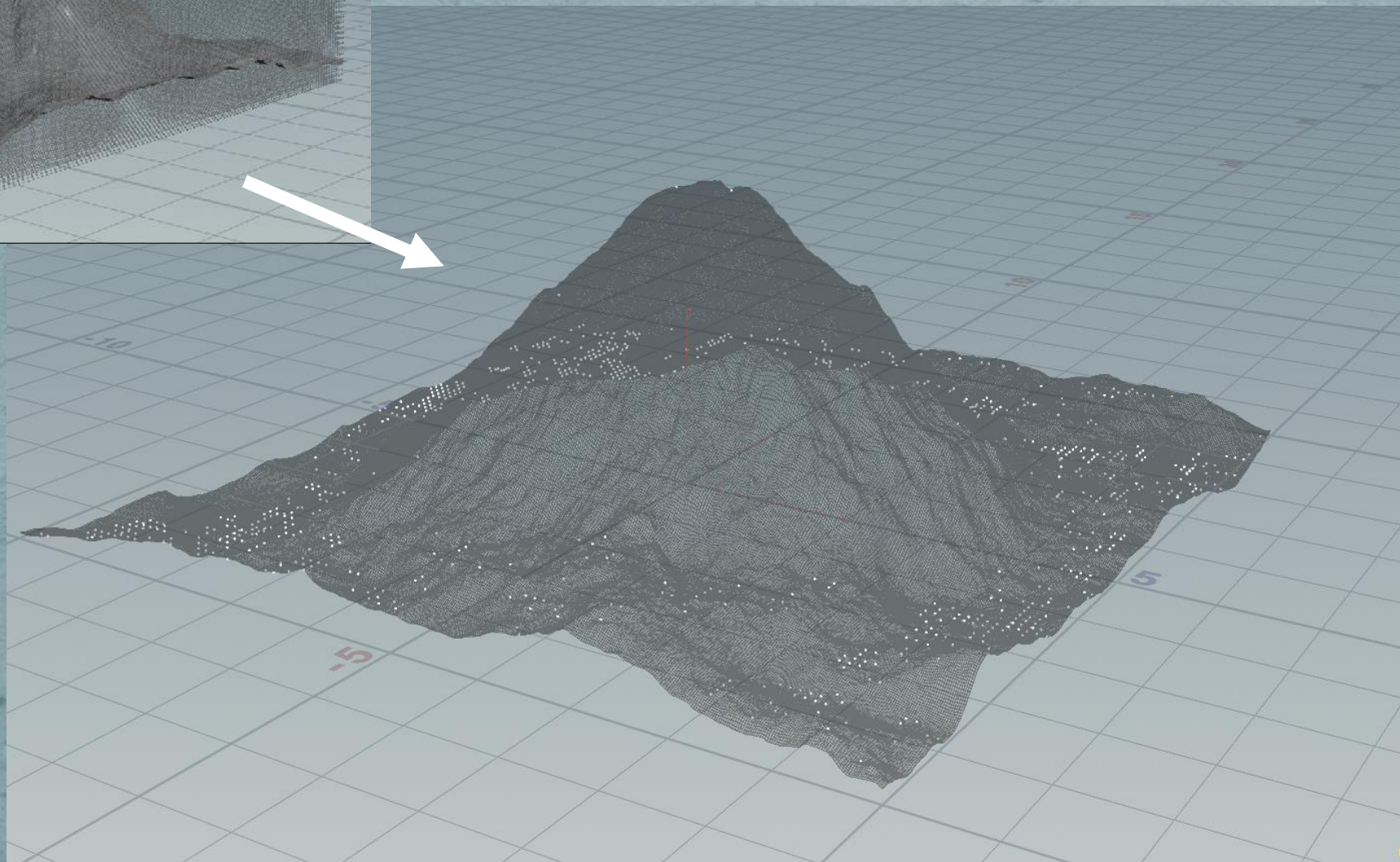
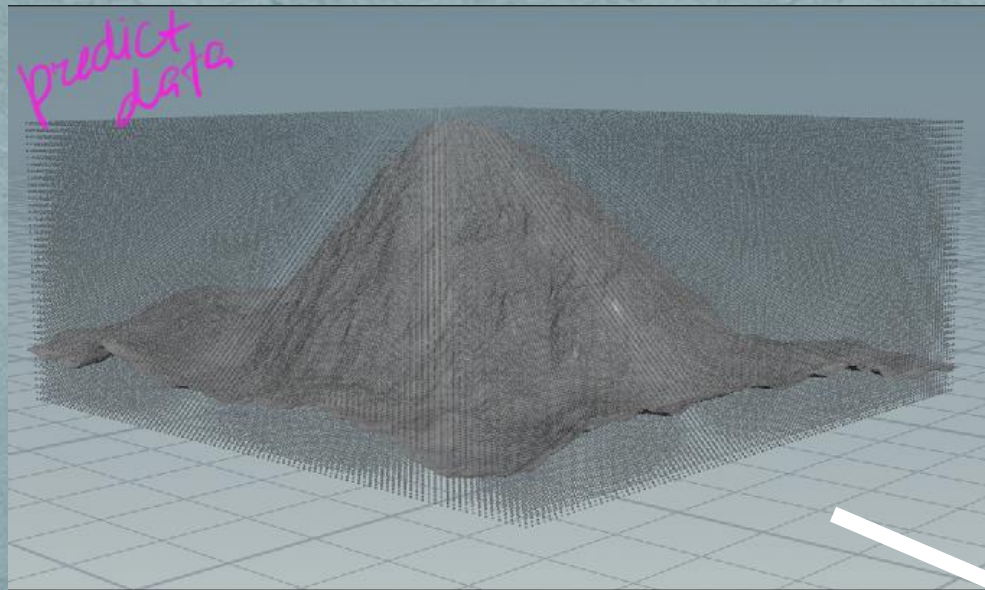
provided by user

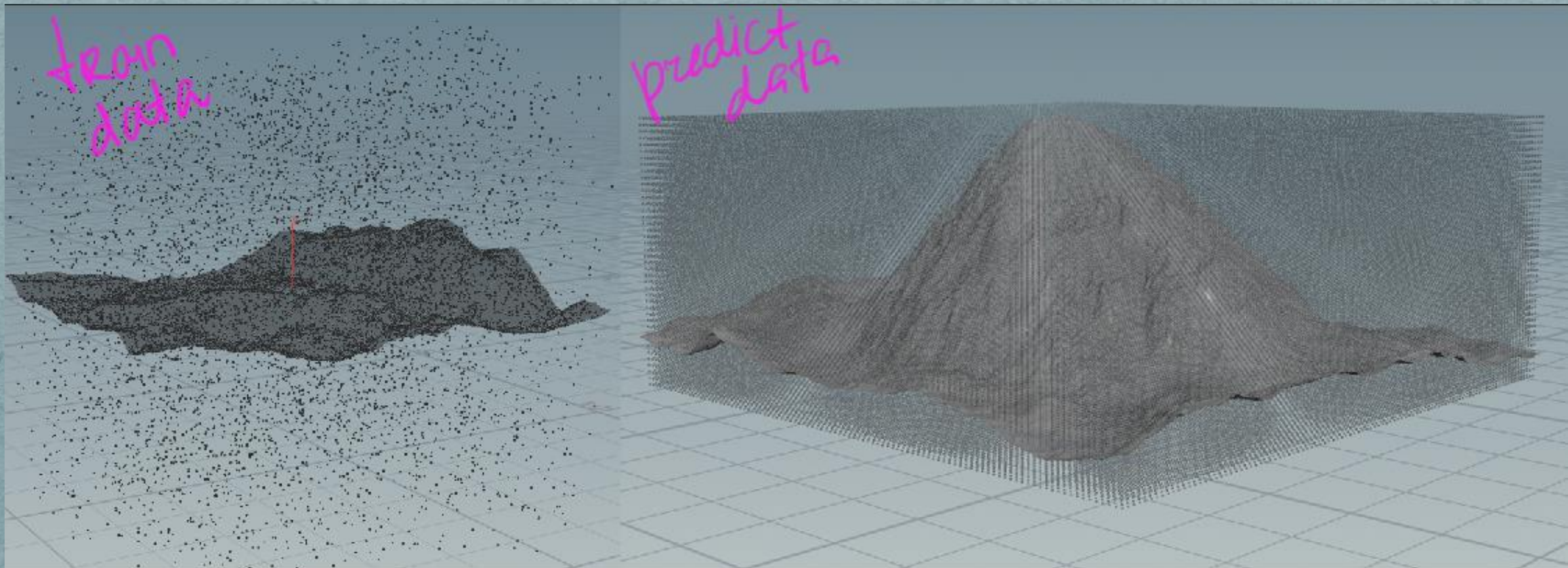


From Pixar's "Lifted"

my deep dive into the depths







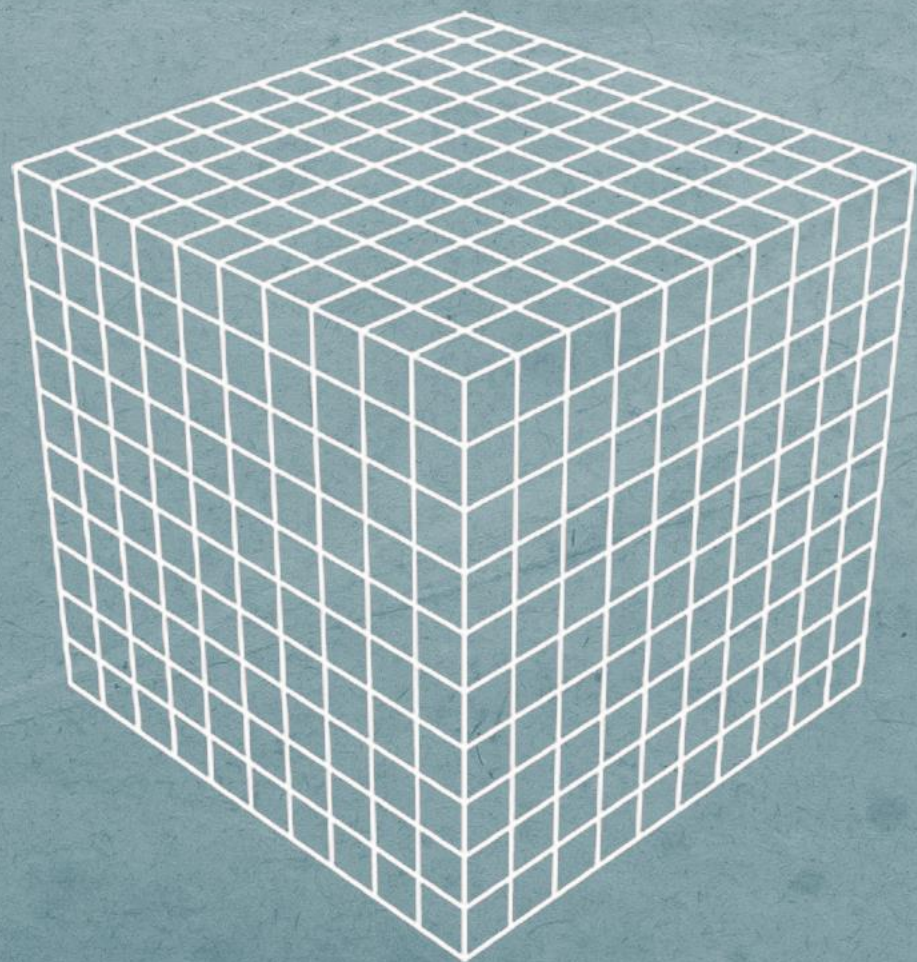
Procedural Lake Village
Anastasia Opara 2016

Example of the
Generated Content



www.anastasiaopara.com

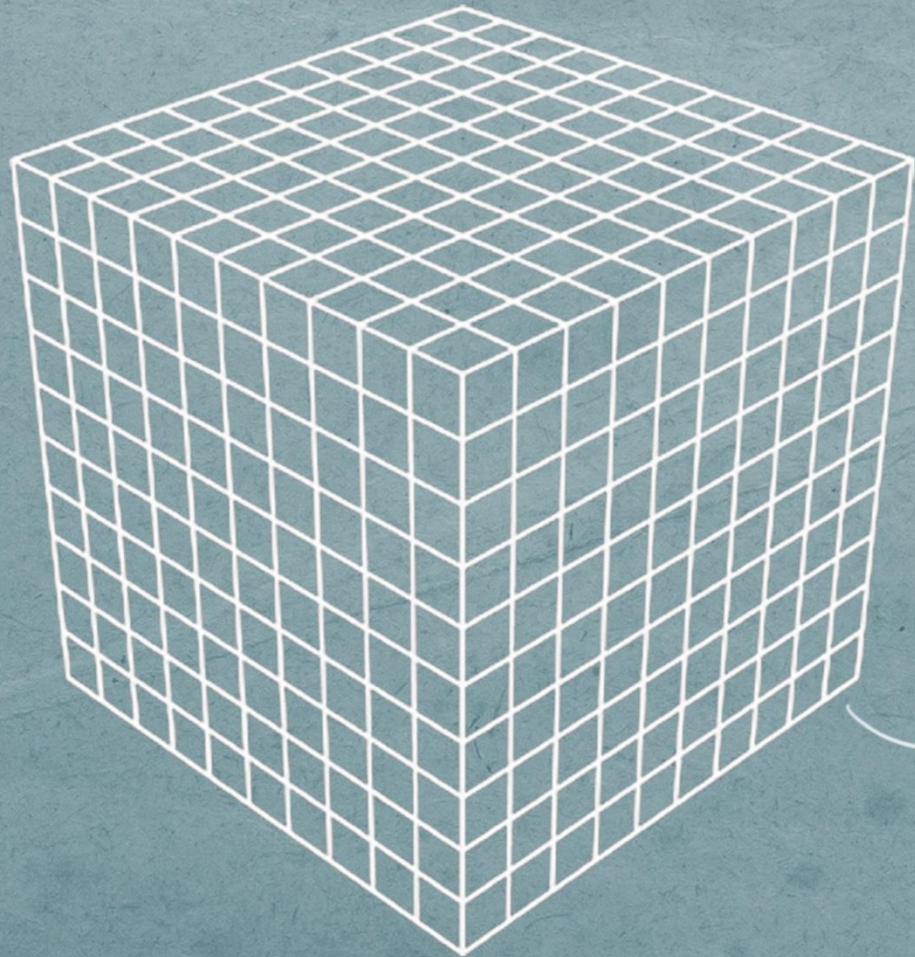
Houdini
3D ANIMATION TOOLS



← $[+x, -x, +y, -y, +z, -z]$

↓ for every

$[wall, door, window, \dots]$



$[+X, -X, +Y, -Y, +Z, -Z]$

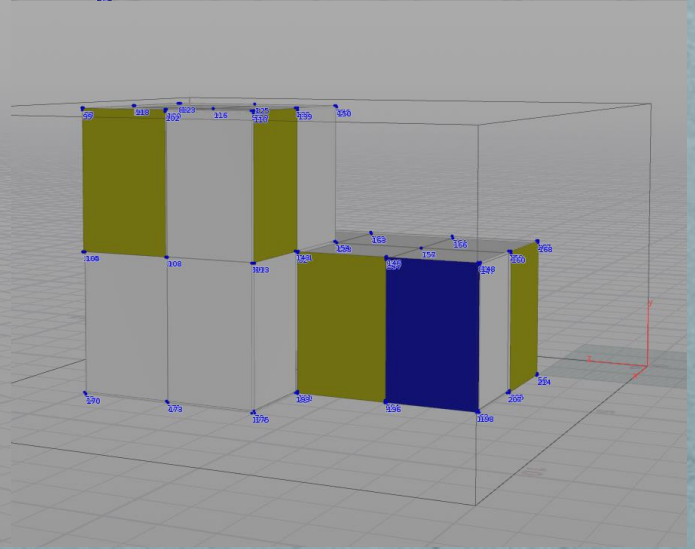
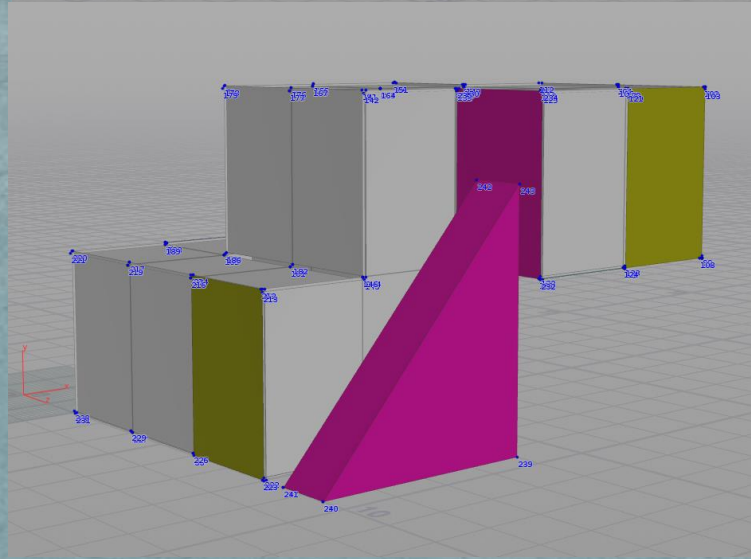
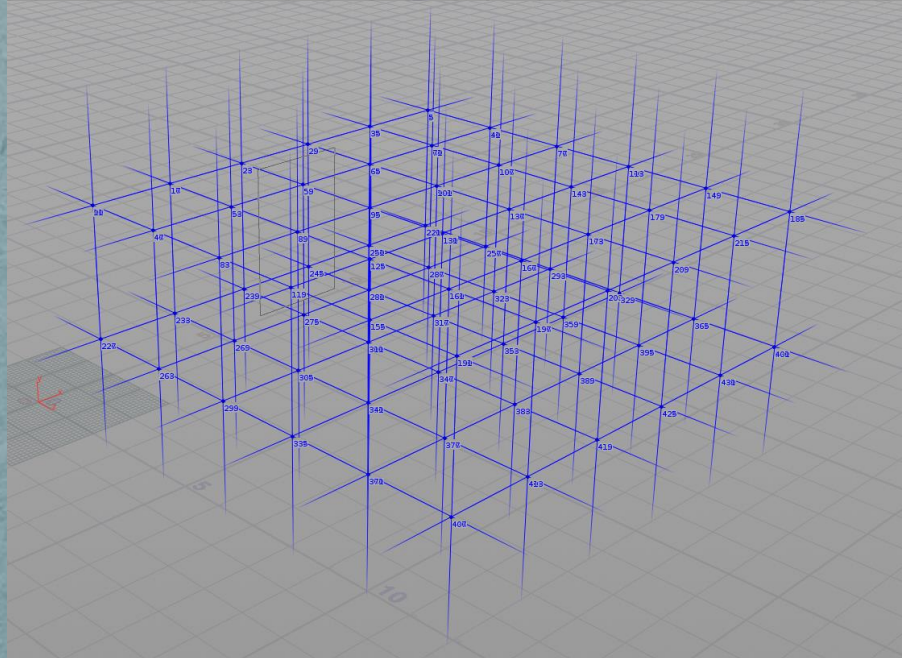
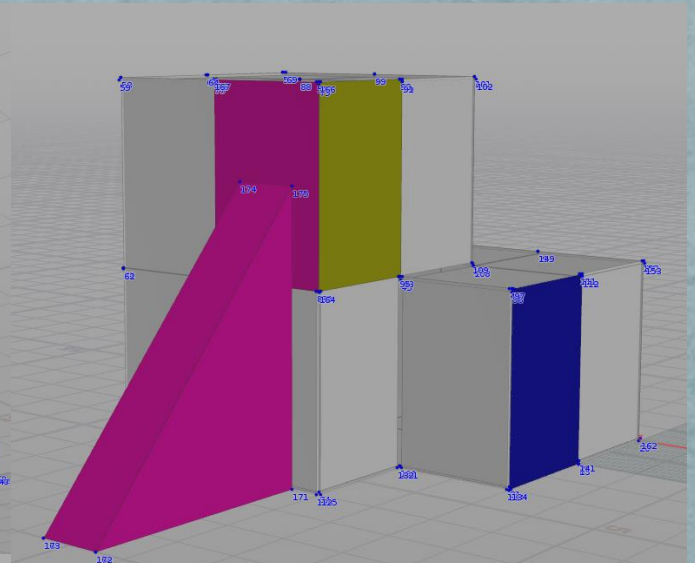
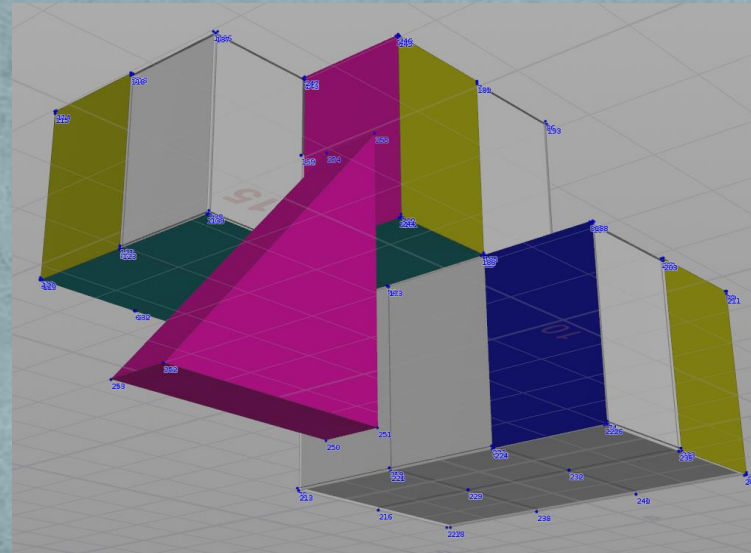
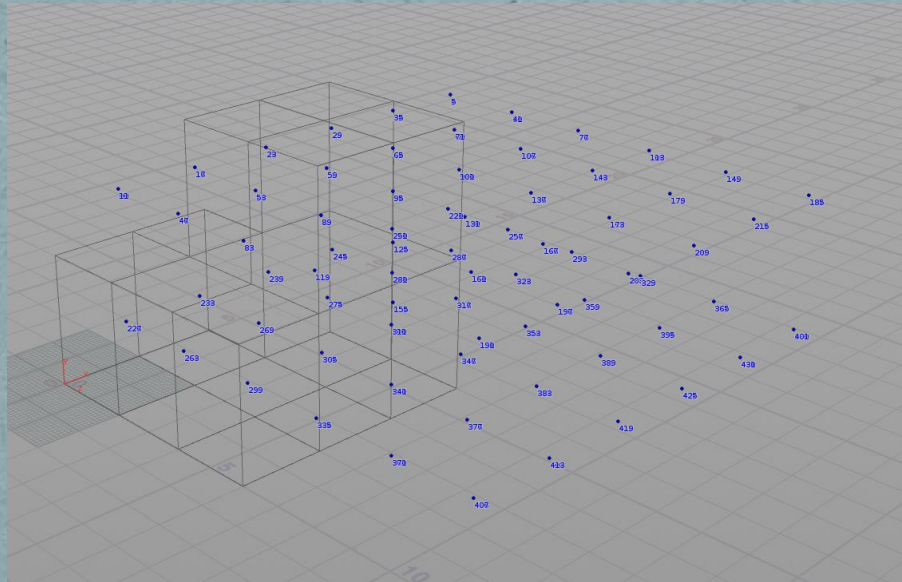
for every

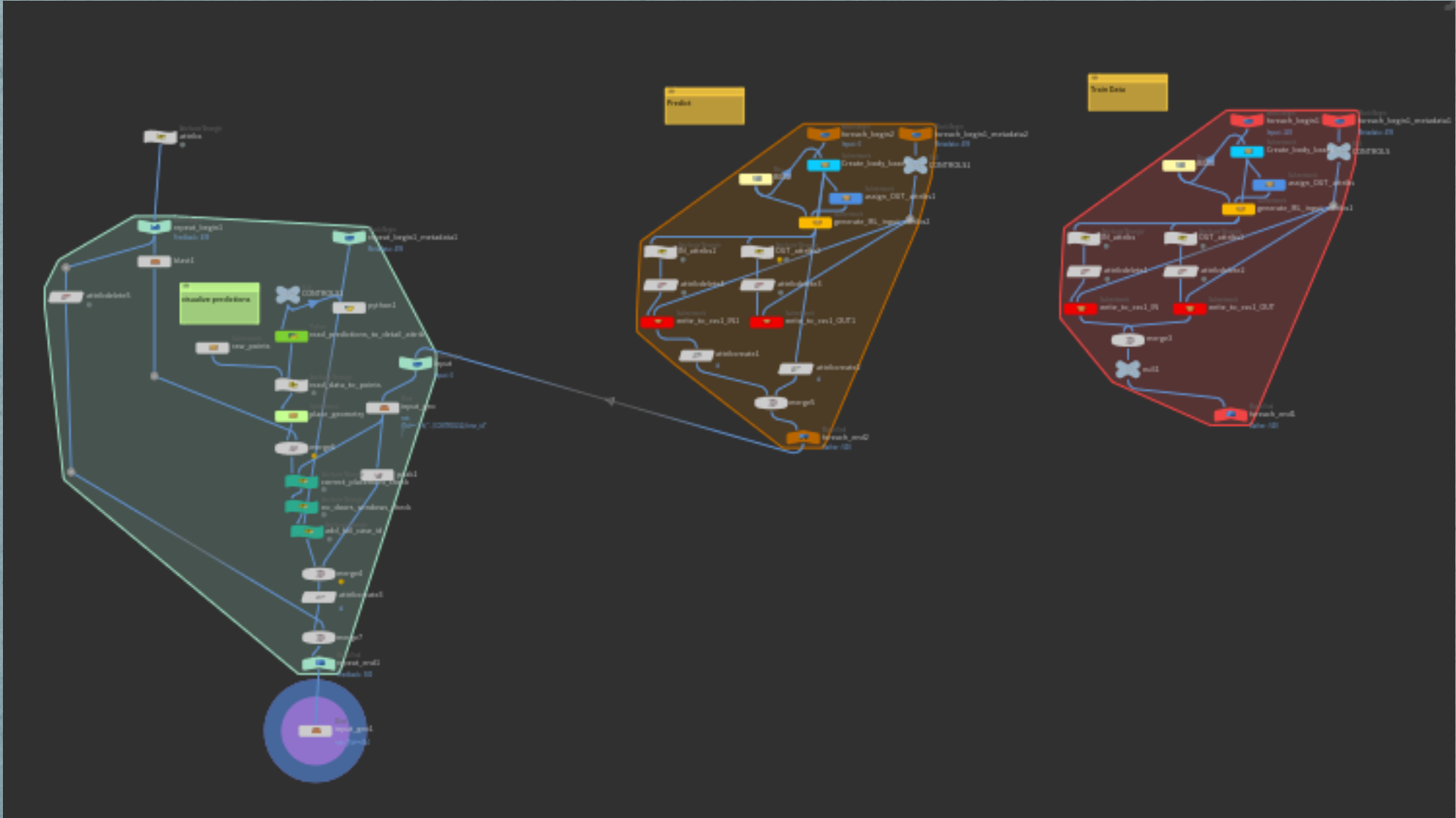
[wall, door, window, ...]

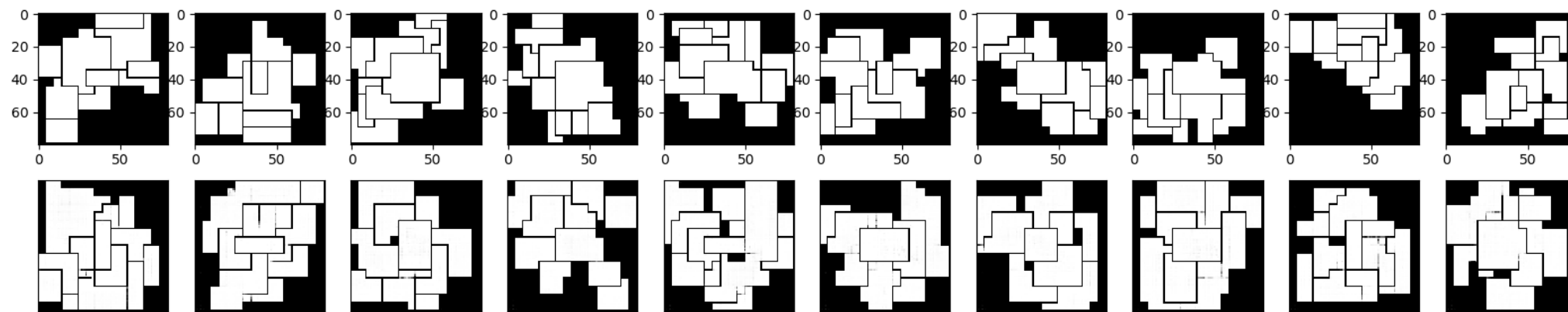
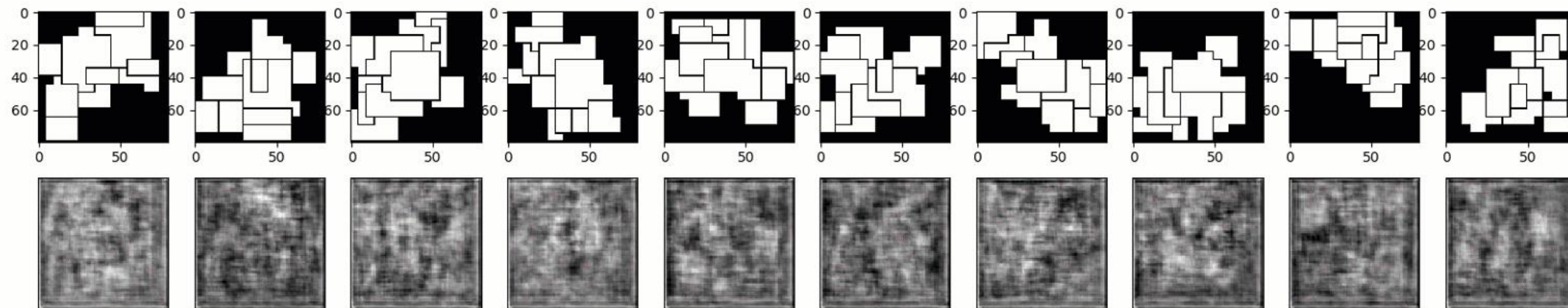


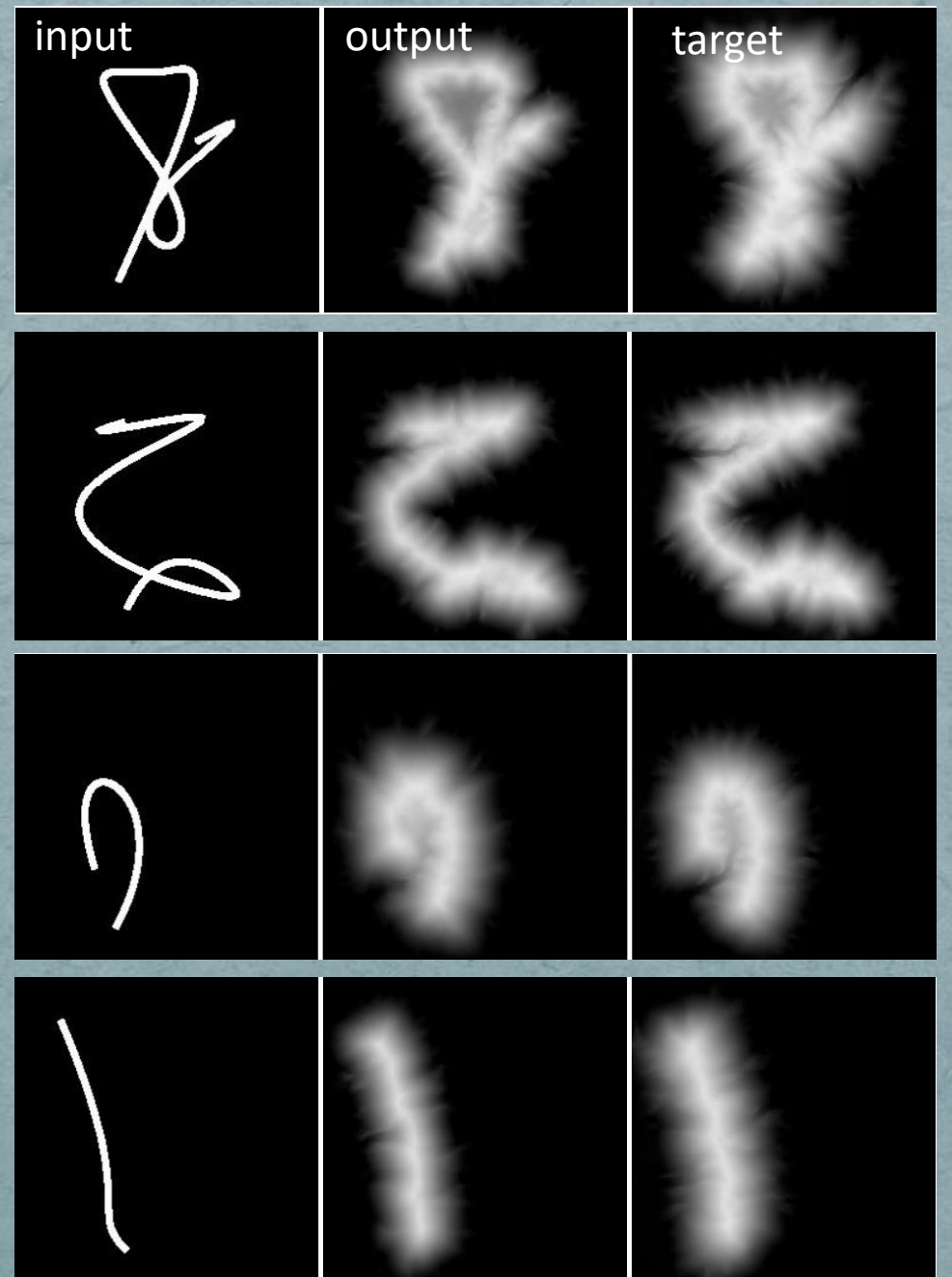
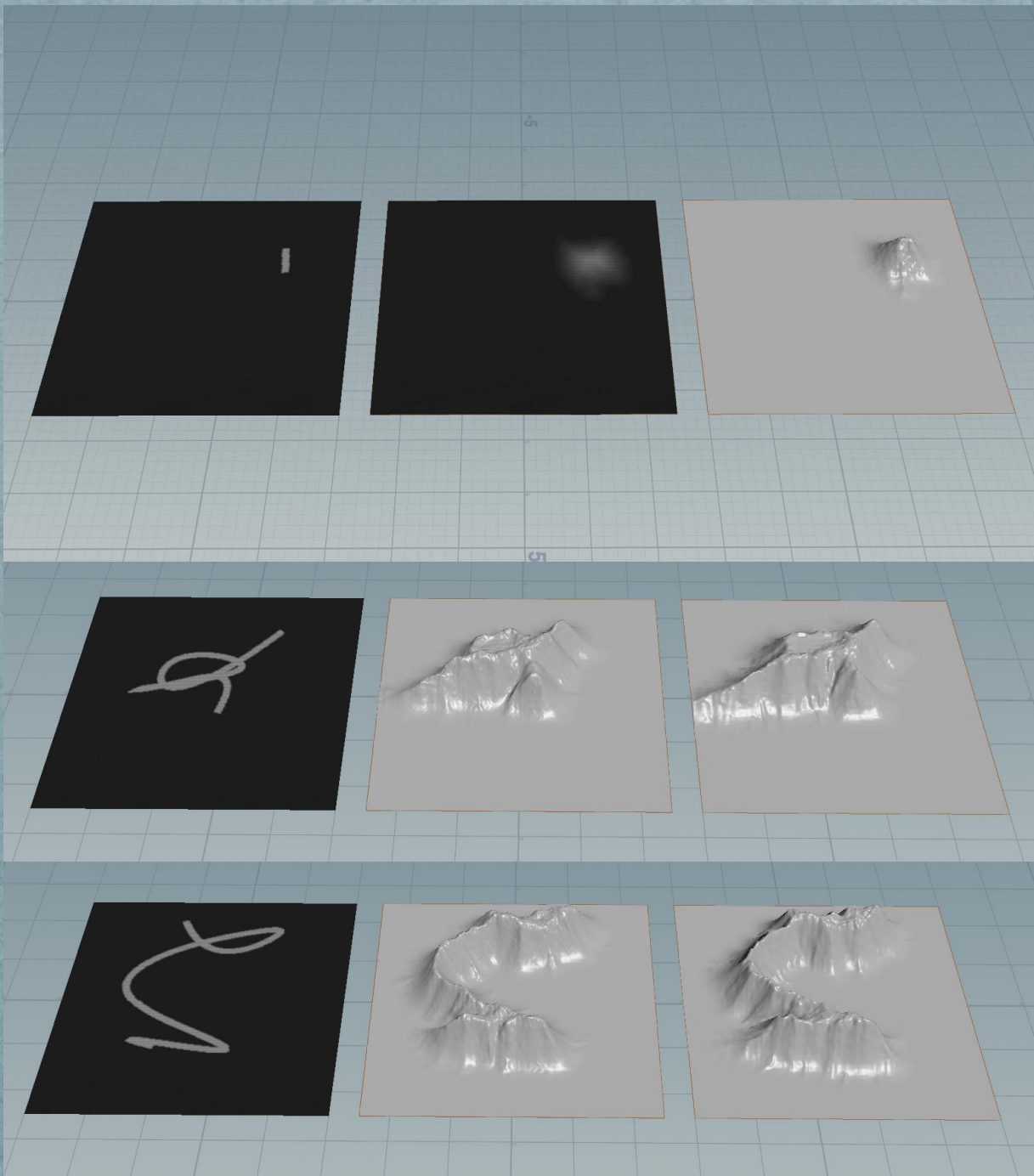
"X"
"window"

$[0, 0, 1]$, +X
 $[0, 0, 0]$, -X
 $[0, 0, 0]$, +Y
...
 $[0, 0, 0]$, -Z









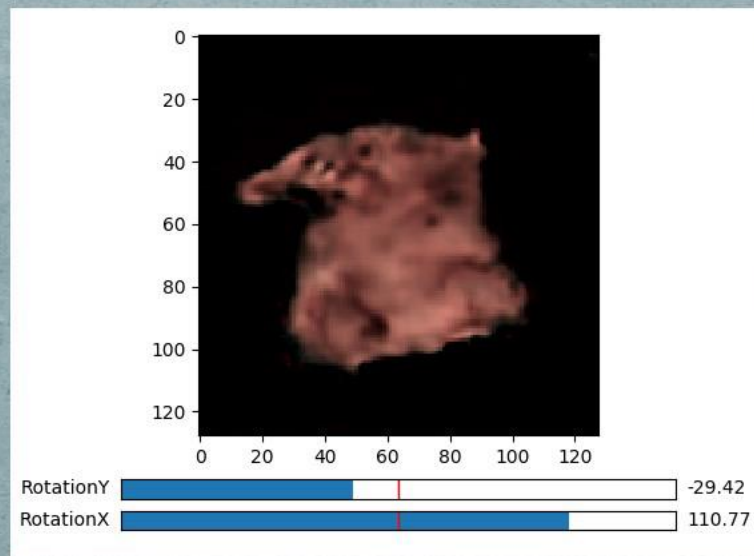
THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.





I had this very
disturbing dream,
doctor... I was
trying to imagine
a pig from an
angle I had never
seen before...



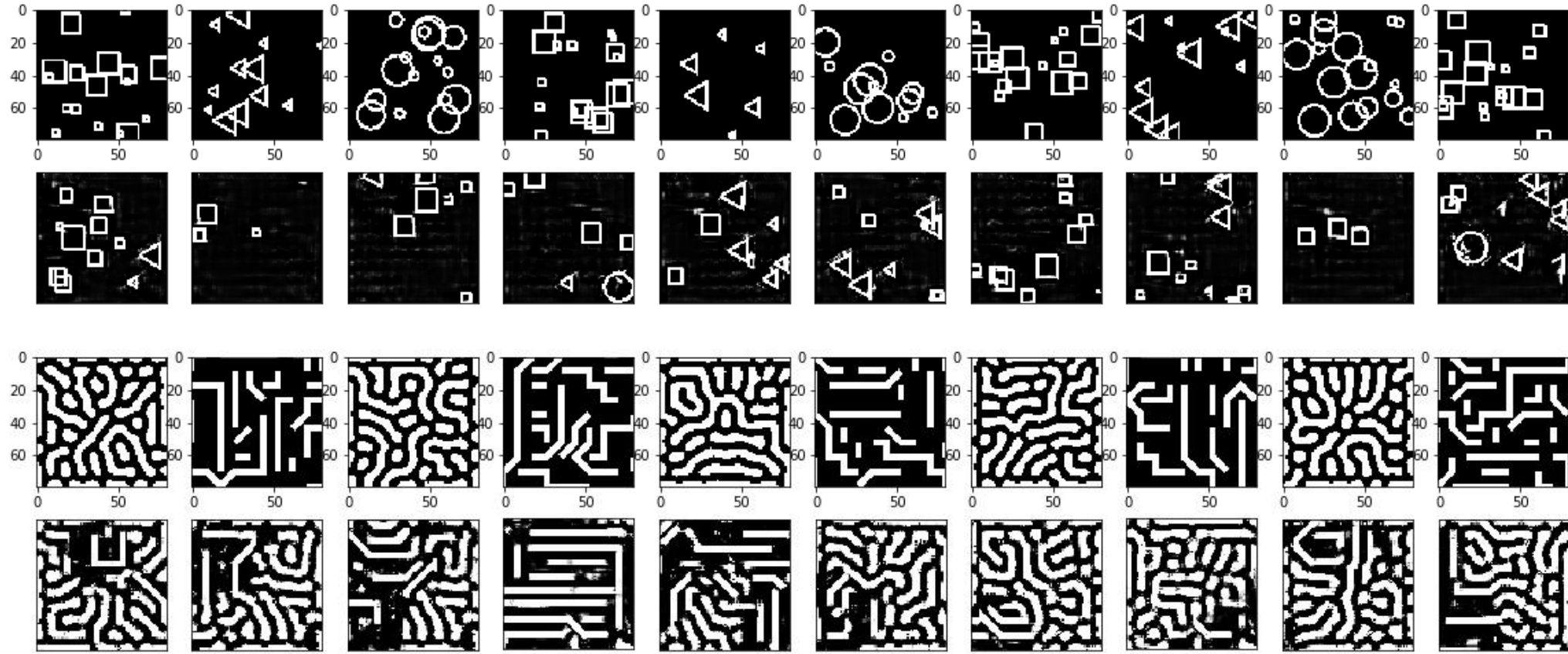
Proceduralism + DL today

Procedural
data

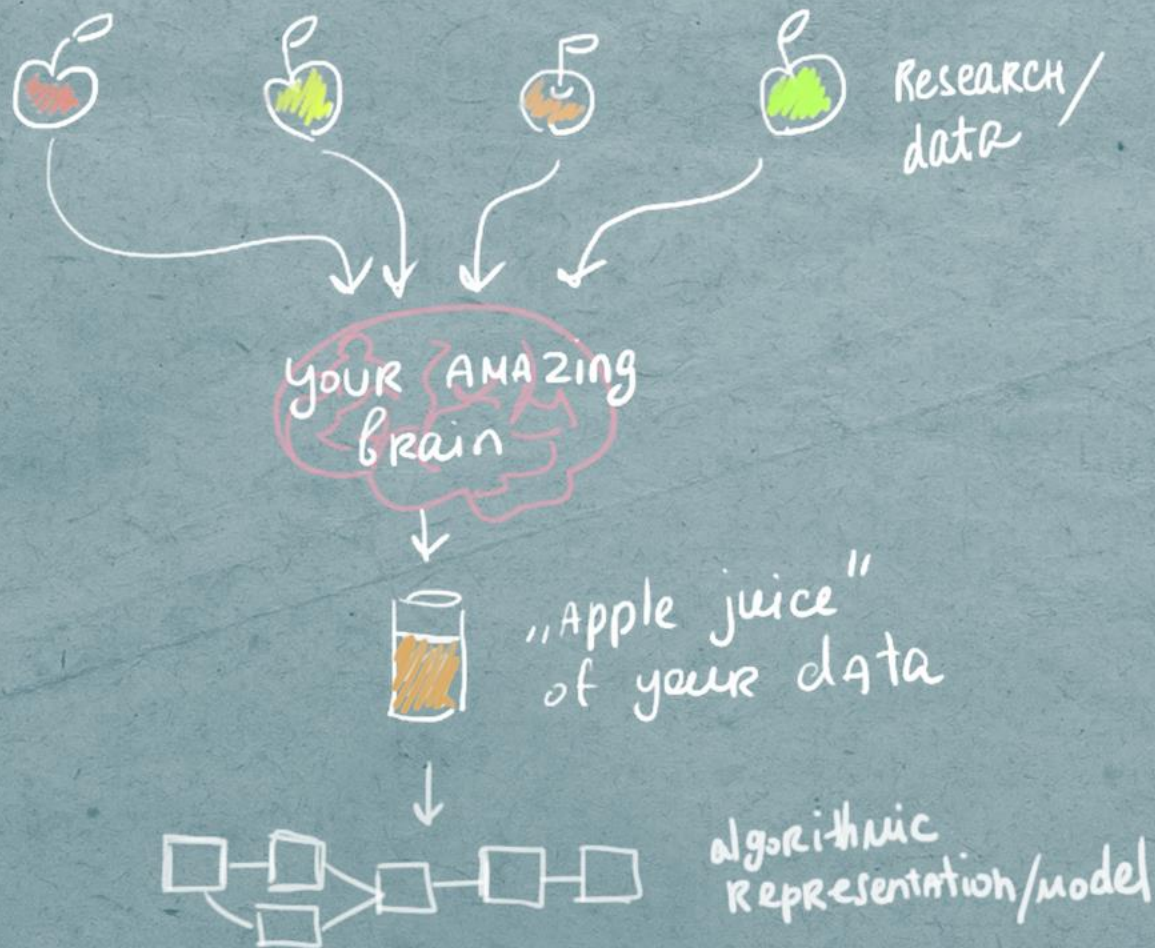
NNs as a
compression

combining multiple
hand-crafted networks

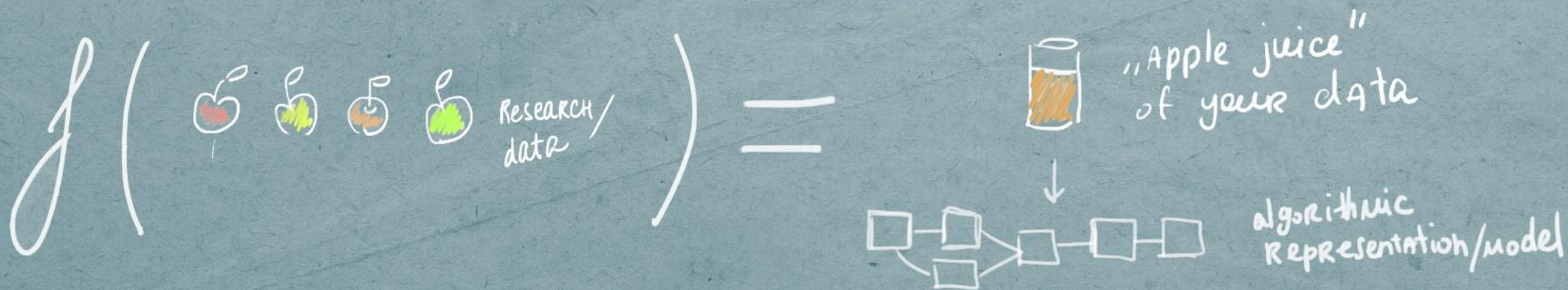
Proceduralism + DL today



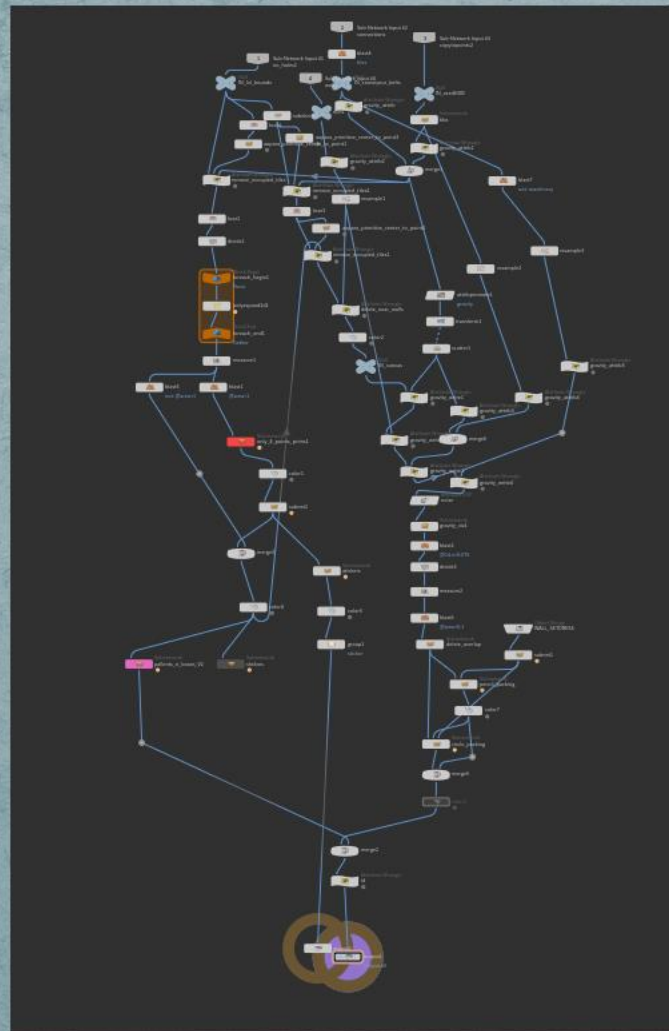
Proceduralism + DL tomorrow?



Proceduralism + DL tomorrow?

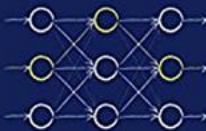


Proceduralism + DL tomorrow?



How does one learn this stuff?

MAKE YOUR OWN NEURAL NETWORK



A gentle journey through the mathematics of neural networks, and making your own using the Python computer language.

TARIQ RASHID

"Make your own neural network"

In Leibniz Notation:
If $y=f(x)$ and $u=g(x)$ then,
 $\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$, where $\frac{dy}{du}$, $\frac{du}{dx}$, and u activation function of the sum of out weights multiplied by output. We will get the following:

$$\frac{\partial E}{\partial w_{ja}} = -2(t_j - o_j) \cdot \frac{\partial \sigma}{\partial w_{ja}}$$

$$\frac{\partial E}{\partial w_{ja}} = -2(t_j - o_j) \cdot \sigma(1 - \sigma)(\sum w_{ja}x_j)$$

Some functions turn into horrible expressions once you try to differentiate them. The sigmoid has a nice and easy to use result. It's one of the reasons the sigmoid is popular for activation function for NNs.

$$\frac{\partial \text{sigmoid}(x)}{\partial x} = \text{sigmoid}(x) \cdot (1 - \text{sigmoid}(x))$$

$$\frac{\partial E}{\partial w_{ja}} = -2(t_j - o_j) \cdot \sigma \text{sigmoid}(\sum w_{ja}x_j) \cdot (1 - \sigma \text{sigmoid}(\sum w_{ja}x_j)) \cdot \frac{\partial (\sum w_{ja}x_j)}{\partial w_{ja}}$$

$$\frac{\partial E}{\partial w_{ja}} = -2(t_j - o_j) \cdot \sigma \text{sigmoid}(\sum w_{ja}x_j) \cdot (1 - \sigma \text{sigmoid}(\sum w_{ja}x_j)) \cdot x_j$$

That is the magic expression we've been working for. We omit x_j because it doesn't interest us. We are only interested in the slope.

Now, how do we find a function for the weights between the input and the hidden layer? We can simply rebuild this expression! We can use the back-propagated error, let's call it δ_j . The sigmoid function stays the same, but the sum expression inside σ refer to the preceding layers, so the sum of over all the inputs moderated by the weights into a hidden node.

$$\frac{\partial E}{\partial w_{ij}} = -(\delta_j) \cdot \sigma \text{sigmoid}(\sum w_{ij}n_i) \cdot (1 - \sigma \text{sigmoid}(\sum w_{ij}n_i)) \cdot n_i$$

Remember that the weights are changed in a direction opposite to the gradient. We also moderate the change by using a learning rate, which we note here as α .

$$\text{new } w_{ja} = \text{old } w_{ja} - \alpha \cdot \frac{\partial E}{\partial w_{ja}}$$

keep a learning journal!

Image-to-Image Translation with Conditional Adversarial Networks

Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory, UC Berkeley
{isola, junyanz, tinghui, efros}@eecs.berkeley.edu

04v2 [cs.CV] 22 Nov 2017

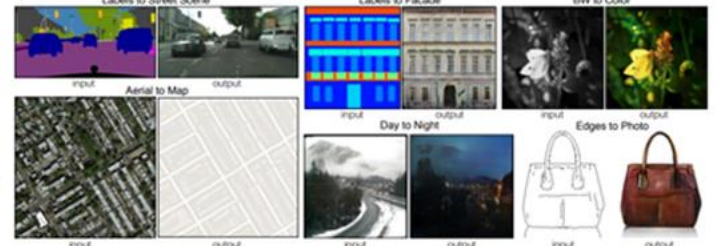


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

read & implement papers & explore other people's implementations!

MIT Massachusetts Institute of Technology


6.034, Fall 2010

Artificial Intelligence
Patrick H. Winston

Lecture 1: Introduction and Scope

MIT AI course

course from deeplearning.ai on COURSERA



deeplearning.ai | COURSERA

Connectionism is a set of approaches in the fields of artificial intelligence, cognitive science and philosophy of mind, that attempts to represent mental or behavioral phenomena as emergent processes of *interconnected networks of simple units*. (Wikipedia)

Anastasia Opara (@anastasiaopara)



SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION

STOCKHOLM - LOS ANGELES - MONTRÉAL - REMOTE

[EA.COM/SEED](https://ea.com/seed)

WE'RE HIRING!

↖ Breda!
(contact:
Jasper Bekkers)