Visualizing Neural Nets

By Daniel Fishbein and Jivitesh Debata

ARITHMOPHOBIA WARNING

The rest of this presentation may not be appropriate for some audiences.

Viewer discretion is advised.

Summary:

1. Our story starts with Caution
2. Can we predict when Neural networks will fail
3. Brute Force Approach
4. Viewing the minima
5. Project Goal
6. Previous Work
7. First task
8. Our Second task
9. Our Experiment
10. Our Third Task
11. This is where we are now
12. Future work
13. Current Related Work
14. What we learned through this process?
15. Sources
16. QA

Our story starts with “Caution”.

Criticisms of “Caution”.

- “Cool idea but ‘Caution’ is not clearly defined.”
- “What are you guys trying to do?”
- **Could you be less vague on what you want to do??**

What we decided to do:

- Focused on when AI messes up.
- Can we predict when NN’s will fail?

Can we predict when NN's will fail?

- Explainable AI - "**Explainable AI** (**XAI**), also known as **Interpretable AI**, or **Explainable Machine Learning** (**XML**),[1] is artificial intelligence (AI) in which humans can understand the reasoning behind decisions or predictions made by the AI." - Wikipedia
  - Prediction accuracy
  - Traceability
  - Decision understanding

  -IBM

The Brute-Force Approach

The input space of a Neural Net is :

$$x^n$$

n = number of input parameters

x = number of different values n can take

The Brute-Force Approach

A Neural Net image classifier that takes 1280 X 720 colored images has an input space of:

$$(256 * 3)^{1280*720} \approx 10^{2659148}$$

Note: There are $\approx 10^{82}$ atoms in the observable universe

Paper published in 2021

Learning in High Dimension Always Amounts to Extrapolation

Randall Balestriero[1], Jérôme Pesenti[1], and Yann LeCun[1,2]

[1]Facebook AI Research, [2]NYU

{rbalestriero,pesenti,yann}@fb.com

Abstract

The notion of interpolation and extrapolation is fundamental in various fields from deep learning to function approximation. Interpolation occurs for a sample $\boldsymbol{x}$ whenever this sample falls inside or on the boundary of the given dataset's convex hull. Extrapolation occurs when $\boldsymbol{x}$ falls outside of that convex hull. One fundamental (mis)conception is that state-of-the-art algorithms work so well because of their ability to correctly interpolate training data. A second (mis)conception is that interpolation happens throughout tasks and datasets, in fact, many intuitions and theories rely on that assumption. We empirically and theoretically argue against those two points and demonstrate that on any high-dimensional (>100) dataset, interpolation almost surely never happens. Those results challenge the validity of our current interpolation/extrapolation definition as an indicator of generalization performances.

The moment of clarity

Machine learning, specifically Neural Nets, is taught as: “The weights are randomly initialized and optimized during the training to minimize a loss function.” - [1]. “The Network is trying to find the ‘local minima’ where it gets the best results” - Every Introduction to AI ever. ~CSCI 635

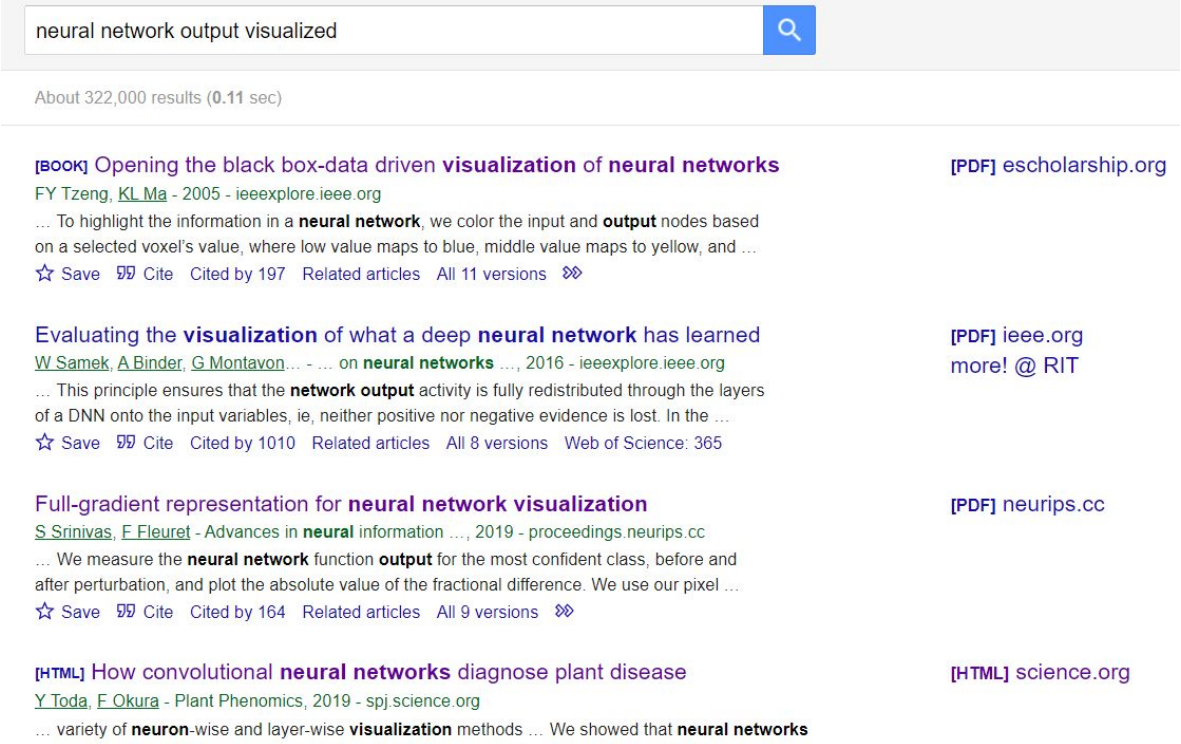
What if we didn’t care about finding a better minima and wanted to view the minima we are at?

View the minima

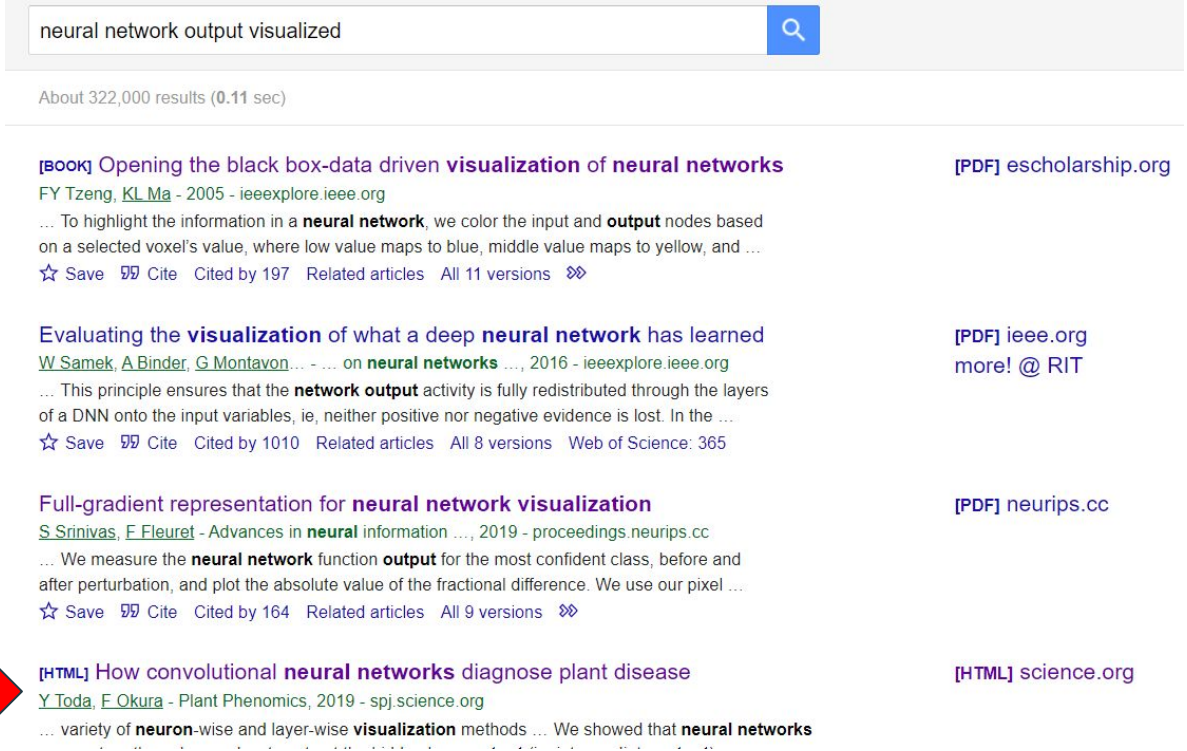
Project Goal

1.) Visualize what this landscape could look like.
2.) Visualise this landscape in an actual NN

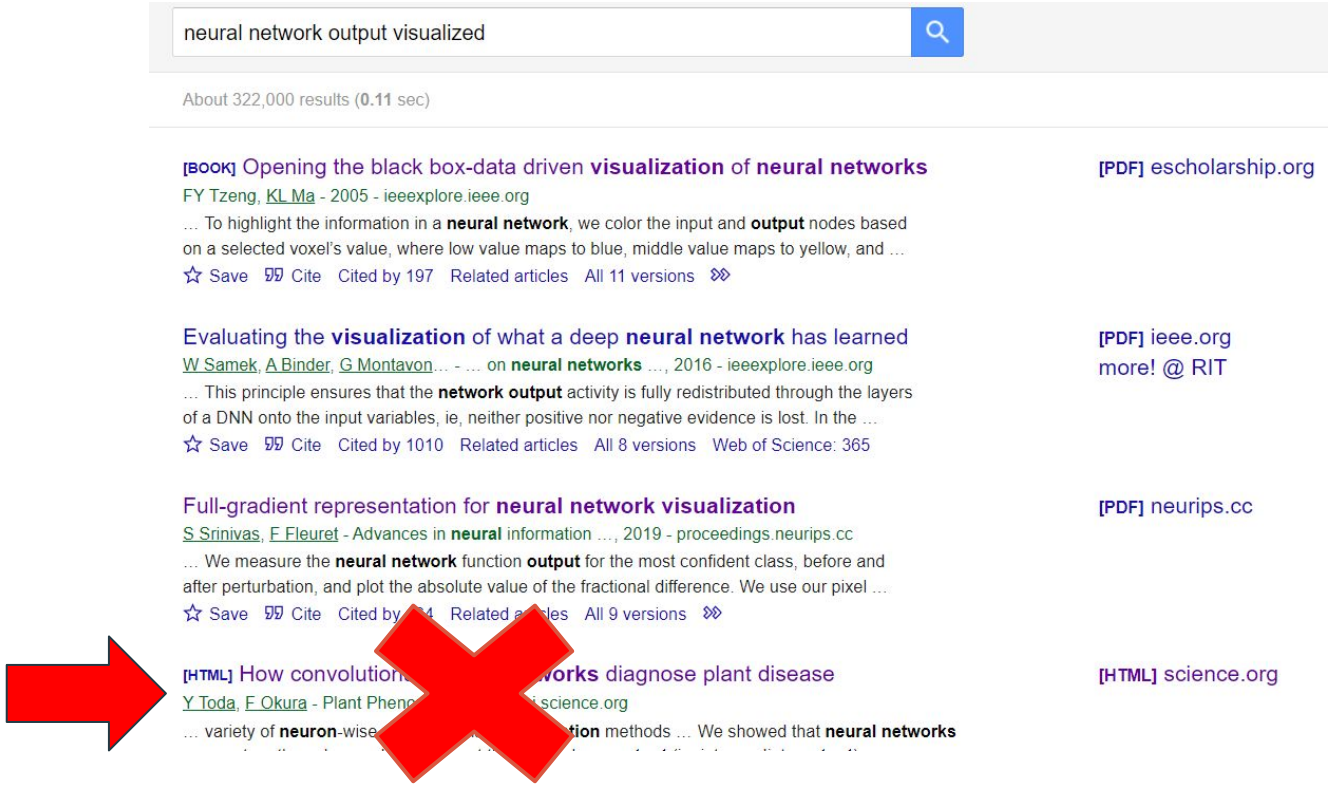
Previous work:



Previous work:



Previous work:



Previous work:

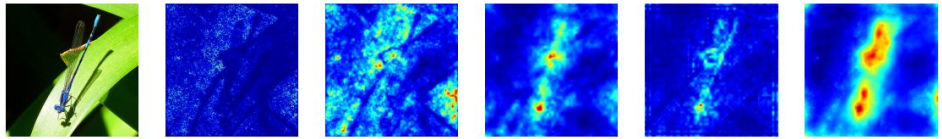


Figure 1: Visualization of bias-gradients at different layers of a VGG-16 pre-trained neural network. While none of the intermediate layer bias-gradients themselves demarcate the object satisfactorily, the full-gradient map achieves this by aggregating information from the input-gradient and all intermediate bias-gradients. (see Equation 2).

Full-Gradient Representation for Neural Network Visualization

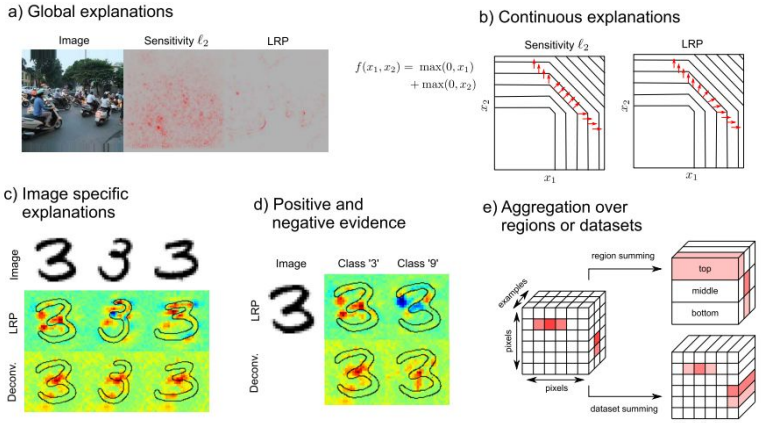
Suraj Srinivas
Idiap Research Institute & EPFL
suraj.srinivas@idiap.ch

François Fleuret
Idiap Research Institute & EPFL
francois.fleuret@idiap.ch

Abstract

We introduce a new tool for interpreting neural net responses, namely full-gradients, which decomposes the neural net response into input sensitivity and per-neuron sensitivity components. This is the first proposed representation which satisfies two key properties: *completeness* and *weak dependence*, which provably cannot be satisfied by any saliency map-based interpretability method. For convolutional nets, we also propose an approximate saliency map representation, called *FullGrad*, obtained by aggregating the full-gradient components.

Previous work:



Evaluating the Visualization of What a Deep Neural Network Has Learned

Wojciech Samek, *Member, IEEE*, Alexander Binder, *Member, IEEE*, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller, *Member, IEEE*

***Abstract*— Deep neural networks (DNNs) have demonstrated impressive performance in complex machine learning tasks such as image classification or speech recognition. However, due to their multilayer nonlinear structure, they are not transparent, i.e., it is hard to grasp *what* makes them arrive at a particular classification or recognition decision, given a new unseen data sample. Recently, several approaches have been proposed enabling one to understand and interpret the reasoning embodied in a DNN for a single test image. These methods quantify the "importance" of individual pixels with respect to the classification decision and allow a visualization in terms of a heatmap in pixel/input space. While the usefulness of heatmaps can be judged subjectively by a human, an objective quality measure is missing. In this paper, we present a general methodology based on region perturbation for evaluating ordered collections of pixels such as heatmaps. We compare heatmaps computed by three different methods on the SUN397, ILSVRC2012, and MIT Places data sets.**

mated image classification [1]–[4], natural language processing [5], [6], human action recognition [7], [8], or physics [9] (see also [10]). Since DNN training methodologies (unsupervised pretraining, dropout, parallelization, GPUs, etc.) have been improved [11], DNNs are recently able to harvest extremely large amounts of training data and can thus achieve record performances in many research fields. At the same time, DNNs are generally conceived as black box methods, and users might consider this lack of transparency a drawback in practice. Namely, it is difficult to intuitively and quantitatively understand the result of DNN inference, i.e., for an *individual* novel input data point, *what* made the trained DNN model arrive at a particular response. Note that this aspect differs from feature selection [12], where the question is: which fea-

Previous work:

Opening the Black Box — Data Driven Visualization of Neural Networks

Fan-Yin Tzeng* Kwan-Liu Ma*

Department of Computer Science
University of California at Davis

ABSTRACT

Arti cial neural networks are computer software or hardware models inspired by the structure and behavior of neurons in the human nervous system. As a powerful learning tool, increasingly neural networks have been adopted by many large-scale information processing applications but there is no a set of well de ned criteria for choosing a neural network. The user mostly treats a neural network as a black box and cannot explain how learning from input data was done nor how performance can be consistently ensured. We have experimented with several information visualization designs aiming to open the black box to possibly uncover underlying dependencies between the input data and the output data of a neural network. In this paper, we present our designs and show that the visualizations not only help us design more ef cient neural networks, but also assist us in the process of using neural networks for problem solving such as performing a classi cation task.

error bound, learning rate, training algorithm, hidden layer size, and the data vector used, are often chosen in a trial-and-error process.

We believe visualization, which proves to help illustrate and understand the behaviors of complex systems, can also help us understand ANNs and design better ANNs. Previous attempts in using visualization to gain understanding into ANNs, as discussed in Section 3, mainly studied the weights and connections of a neural network and analyzed neural networks in isolation; the data used by the neural network were mostly not looked at.

We therefore take a data-driven approach to the problem of visualizing ANN since gaining insights into a neural network requires the study of not only the network but also how it responds to the input data that it was designed to process. The methods we present enable the interactive exploration of both the input data and the neural network so as to gain more complete picture of how the neural network performs its task. The visualizations can also assist in the selection of network structure and other parameters for an assigned

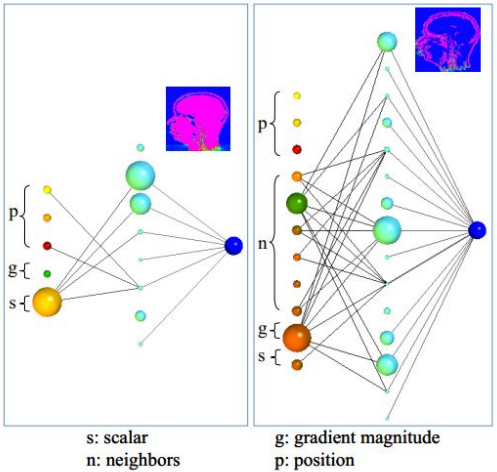
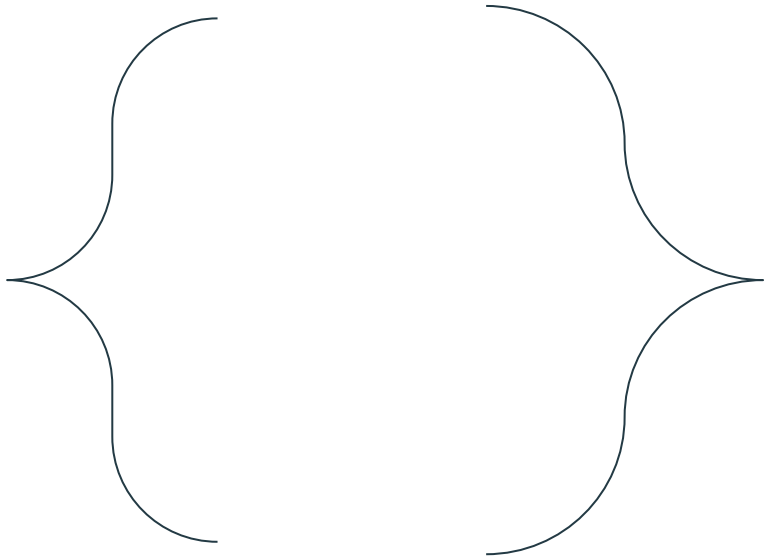


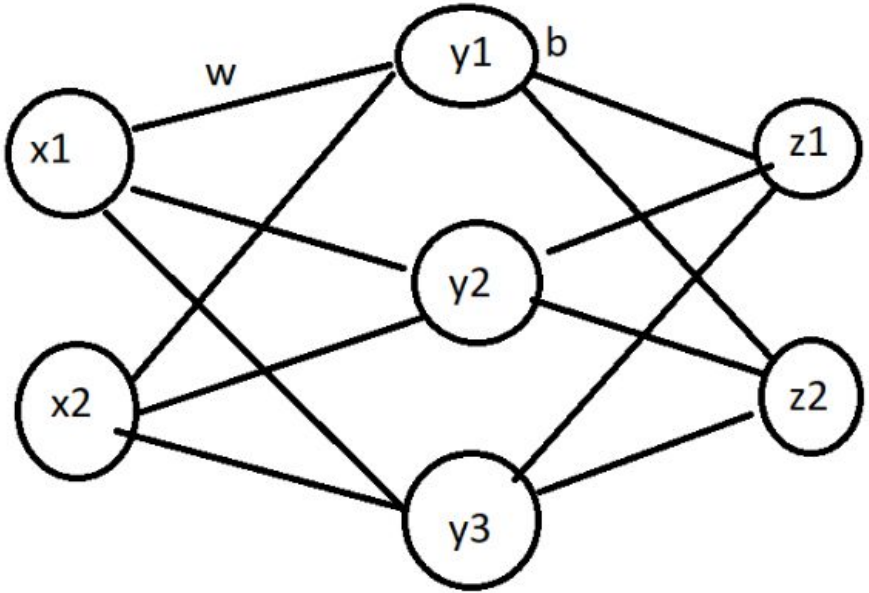
Figure 4: The left image shows a neural network which is trained for classifying the entire head from the data set. The scalar value is the main criterion considered in this classification. The right image is the result of classifying the boundaries. In this case, neighbors and gradient magnitude are shown to be more important. The classification result is shown at the upper right of each network.

Previous work: NONE (:



Our First Task:

Visualize what this landscape could look like.



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$y_i = \sigma(w_{x_1->y_i} * x_1 + w_{x_2->y_i} * x_2 + b_{y_i})$$

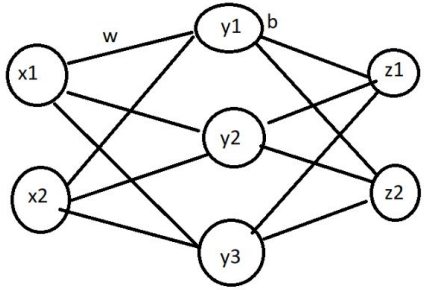
$$z_i = \sigma(w_{y_1->z_i} * y_1 + w_{y_2->z_i} * y_2 + w_{y_3->z_i} * y_3 + b_{z_i})$$

ARITHMOPHOBIA WARNING

The rest of the this presentation may not be appropriate for some audiences.

Viewer discretion is advised.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

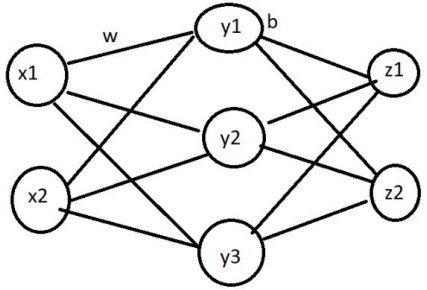


Our First Task:

Visualize what this landscape could look like.

$$(1 + \exp[-(w_{y_1->z_1} * \frac{1}{1 + \exp[w_{x_1->y_1} * x_1 + w_{x_2->y_1} * x_2 + b_{y_1}]}$$
$$+ w_{y_2->z_1} * \frac{1}{1 + \exp[w_{x_1->y_2} * x_1 + w_{x_2->y_2} * x_2 + b_{y_2}]}$$
$$+ w_{y_3->z_1} * \frac{1}{1 + \exp[w_{x_1->y_3} * x_1 + w_{x_2->y_3} * x_2 + b_{y_3}]}$$
$$+ b_{z_1})])^{-1}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

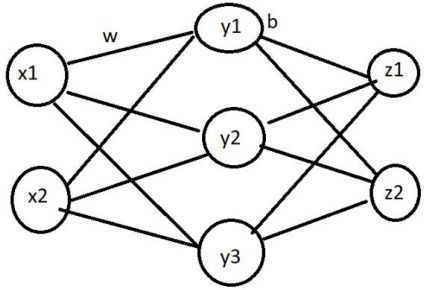


Our First Task:

Visualize what this landscape could look like.

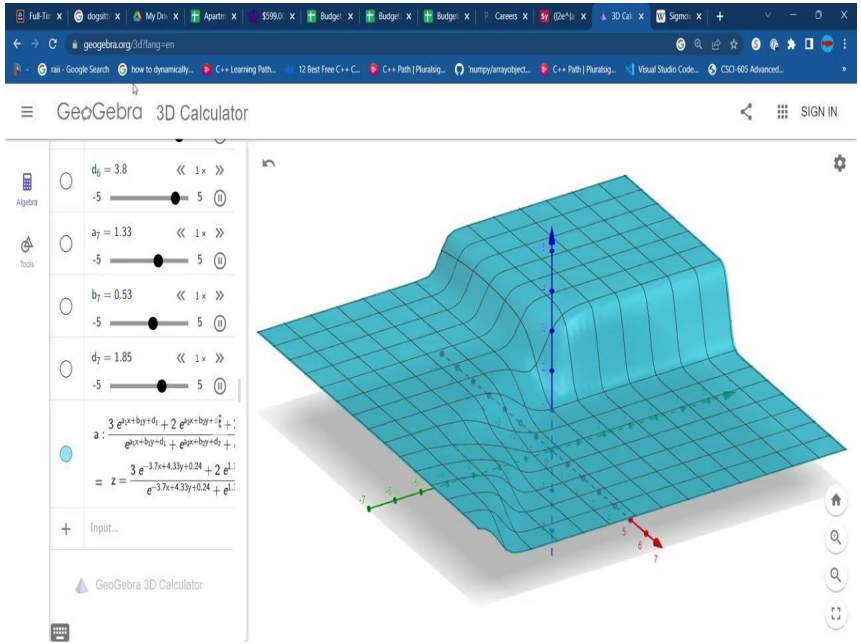
$$\left(\frac{3e^{a_1x+b_1y+d_1} + 2e^{a_2x+b_2y+d_2} + 2e^{a_3x+b_3y+d_3} + 2e^{a_4x+b_4y+d_4} + e^{a_5x+b_5y+d_5} + e^{a_6x+b_6y+d_6} + e^{a_7x+b_7y+d_7}}{e^{a_1x+b_1y+d_1} + e^{a_2x+b_2y+d_2} + e^{a_3x+b_3y+d_3} + e^{a_4x+b_4y+d_4} + e^{a_5x+b_5y+d_5} + e^{a_6x+b_6y+d_6} + e^{a_7x+b_7y+d_7}}\right)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

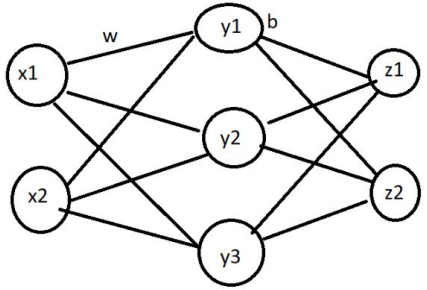


Our First Results:

Visualize what this landscape could look like.



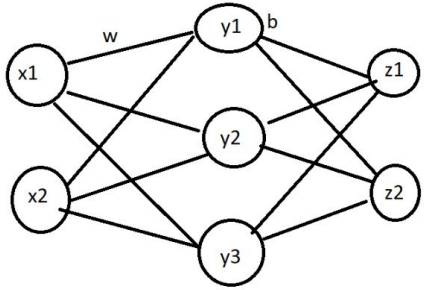
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Project Goal

1.) ~~Visualize what this landscape could look like.~~
2.) Visualise this landscape in an actual NN

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

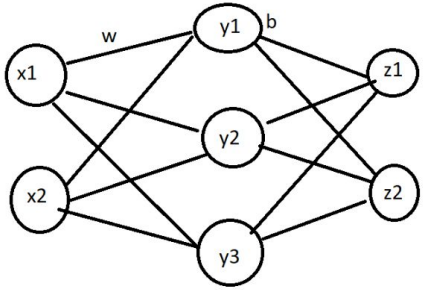


Our Second Task:

Visualise this landscape in an actual NN

1.) Make this network in Tensorflow or Pytorch
2.) Give it a task to optimize
3.) Visualize the output

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

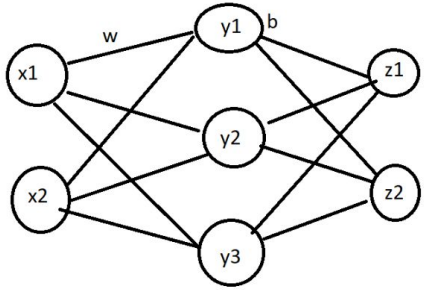


Our Second Task:

Visualise this landscape in an actual NN

1.) ~~Make this network in Tensorflow or Pytorch~~
2.) Give it a task to optimize
3.) Visualize the output

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



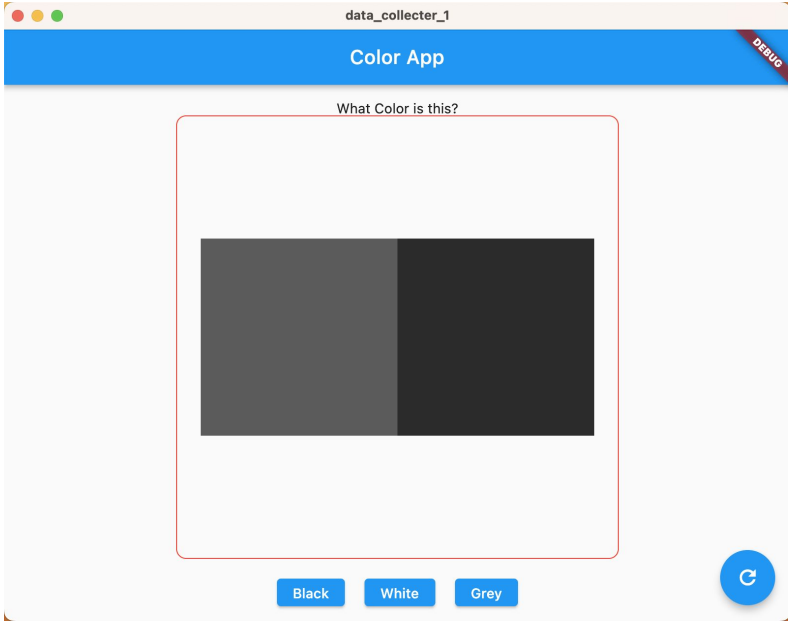
Our Experiment:

Black, White, Grey Experiment

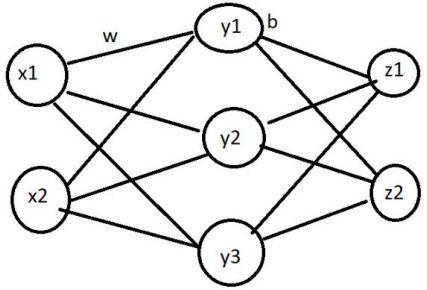
Task: Given 2 pixels, are they Black, White, or Grey?

a.) Mathematically defined dataset
b.) Human labeled dataset

What color is the “image” inside the red box?



$$\sigma(x) = \frac{1}{1+e^{-x}}$$



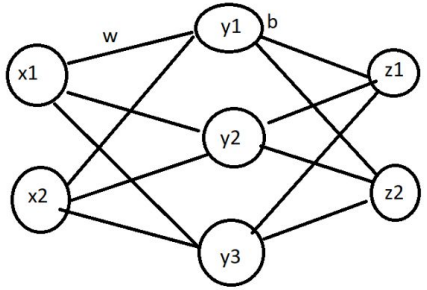
Our Experiment:

Black, White, Grey Experiment

- Mathematically defined dataset
  - S1 DEF
    - P1 >P2 | P1<128
    - P1 <P2 | P2>128
    - P1 + P2 =128 | otherwise
  - S2 DEF :
    - P1 + P2 < 255 | P1>P2
    - P1 + P2 <255 | P2<P1
    - P1 + P2 =128 | otherwise
  - S3 DEF
    - P1 + P2 < 128 | P1<128 P2<128
    - P1 + P2 >128 | P1<128 P2<128
    - Every 3rd image
- Human labeled dataset

Black	Grey	White
0	128	255

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



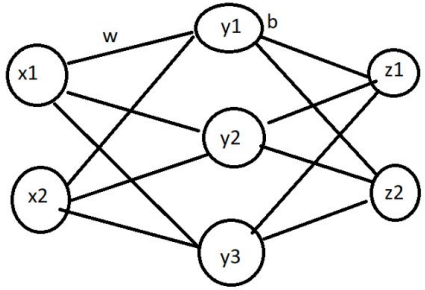
Our Experiment:

Black, White, Grey Experiment

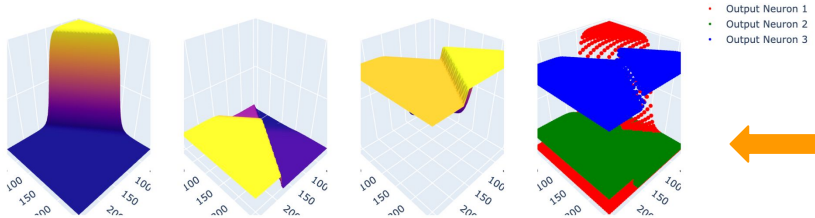
- Mathematically defined dataset
  - S1 DEF
    - P1 >P2 | P1<128
    - P1 <P2 | P2>128
    - P1 + P2 =128 | otherwise
  - S2 DEF :
    - P1 + P2 < 255 | P1>P2
    - P1 + P2 <255 | P2<P1
    - P1 + P2 =128 | otherwise
  - S3 DEF
    - P1 + P2 < 128 | P1<128 P2<128
    - P1 + P2 >128 | P1<128 P2<128
    - Every 3rd image
- Human labeled dataset
  - A human was given 2 squares and asked if the image is black white or grey.
  - The dataset is called “human.zip” and is available at: https://drive.google.com/file/d/1E5riOt3Dg_wGxwqwxixbAvKhB1-GSg4o/view?usp=share_link

Black	Grey	White
0	128	255

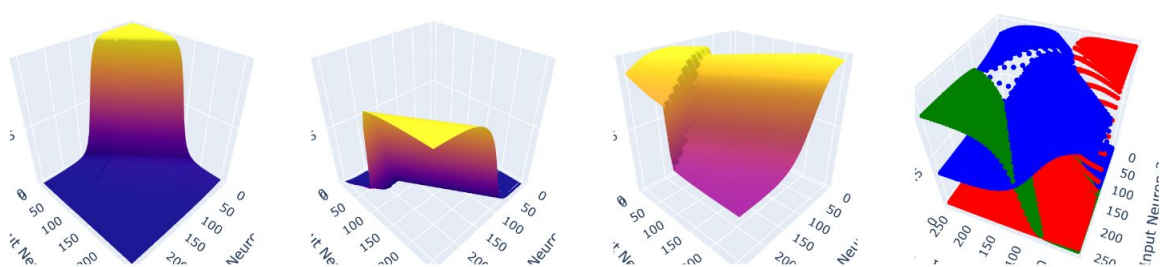
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Our results to Black, White, Grey:

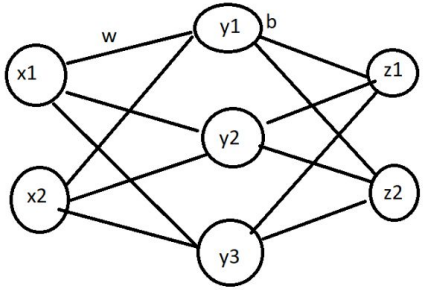


Human Data



Math Data

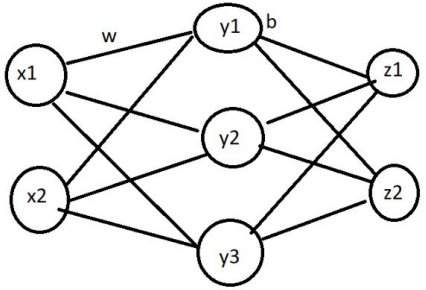
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Project Goal

1.) ~~Visualize what this landscape could look like.~~
2.) ~~Visualise this landscape in an actual NN~~
3.) Find limits/boundaries between these plains
    a.) Highlight their intersections

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



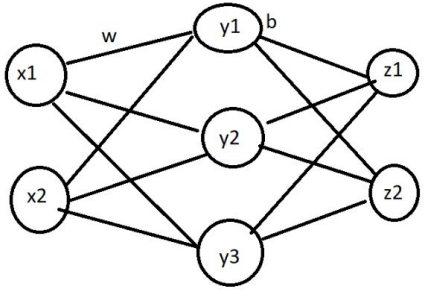
Our Third Task:

Find limits/boundaries between these plains and highlight their intersections

$$(1 + \exp[-(w_{y_1->z_1} * \frac{1}{1 + \exp[w_{x_1->y_1} * x_1 + w_{x_2->y_1} * x_2 + b_{y_1}]}$$
$$+ w_{y_2->z_1} * \frac{1}{1 + \exp[w_{x_1->y_2} * x_1 + w_{x_2->y_2} * x_2 + b_{y_2}]}$$
$$+ w_{y_3->z_1} * \frac{1}{1 + \exp[w_{x_1->y_3} * x_1 + w_{x_2->y_3} * x_2 + b_{y_3}]}$$
$$+ b_{z_1})])^{-1}$$

(Z_1 = Z_2), (Z_1 = Z_3), (Z_2 = Z_3)

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Our Third Task:

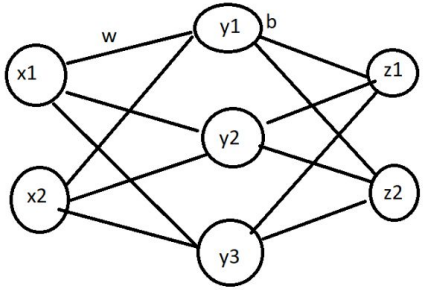
Find limits/boundaries between these plains and highlight their intersections

$$(1+\exp[-(w_{y_1->z_1} * \frac{1}{1+\exp[w_{x_1->y_1} * x_1 + w_{x_2->y_1} * x_2 + b_{y_1}]} + w_{y_2->z_1} * \frac{1}{1+\exp[w_{x_1->y_2} * x_1 + w_{x_2->y_2} * x_2 + b_{y_2}]} + w_{y_3->z_1} * \frac{1}{1+\exp[w_{x_1->y_3} * x_1 + w_{x_2->y_3} * x_2 + b_{y_3}]} + b_{z_1})])^{-1} =$$

$$(1+\exp[-(w_{y_1->z_2} * \frac{1}{1+\exp[w_{x_1->y_1} * x_1 + w_{x_2->y_1} * x_2 + b_{y_1}]} + w_{y_2->z_2} * \frac{1}{1+\exp[w_{x_1->y_2} * x_1 + w_{x_2->y_2} * x_2 + b_{y_2}]} + w_{y_3->z_2} * \frac{1}{1+\exp[w_{x_1->y_3} * x_1 + w_{x_2->y_3} * x_2 + b_{y_3}]} + b_{z_2})])^{-1}$$

(Z_1 = Z_2), (Z_1 = Z_3), (Z_2 = Z_3)

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



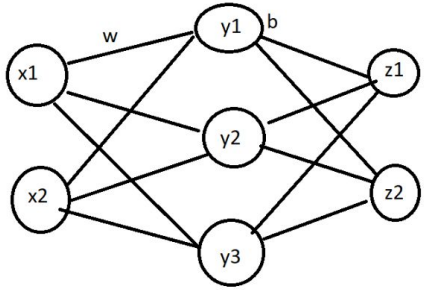
Our Third Task:

Find limits/boundaries between these plains and highlight their intersections

$$\frac{}{\_1\_2*y+b\_3}\Bigg)+b\_1\_1\Bigg) = \left(\frac{1}{1+e^{w\_2\_1*\left(\frac{1}{1+e^{w\_1\_1*x+w\_1\_2*y+b\_1}}\right)+w\_2\_2\left(\frac{1}{1+e^{w\_1\_1*x+w\_1\_2*y+b\_2}}\right)+w\_2\_3\left(\frac{1}{1+e^{w\_1\_1*x+w\_1\_2*y+b\_1}}\right)+b\_3\_2}}\right)$$

NATURAL LANGUAGE | MATH INPUT

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Our Third Task:

Find limits/boundaries between these plains and highlight their intersections

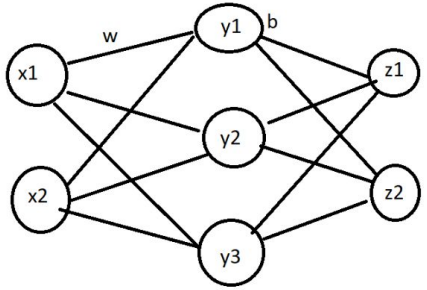
$$\frac{}{\_1\_2*y+b\_3}\Bigg)+b\_1\_1\Bigg) = \left(\frac{1}{1+e^{\,w\_2\_1*\left(\frac{1}{1+e^{\,w\_1\_1*x+w\_1\_2*y+b\_1}}\right)+w\_2\_2\left(\frac{1}{1+e^{\,w\_1\_1*x+w\_1\_2*y+b\_2}}\right)+w\_2\_3\left(\frac{1}{1+e^{\,w\_1\_1*x+w\_1\_2*y+b\_1}}\right)+b\_3\_2}}\right)$$

NATURAL LANGUAGE | MATH INPUT

POPULAR

Wolfram|Alpha doesn't understand your query

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Our Third Task:

Find limits/boundaries between these plains and highlight their intersections

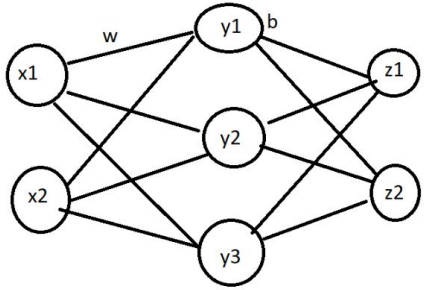
$$C_0 + C_1 e^{B_1} + C_2 e^{B_2} + C_3 e^{B_3} + C_4 e^{B_2+B_3} + C_5 e^{B_1+B_3} + C_6 e^{B_1+B_2} + C_7 e^{B_1+B_2+B_3} = 0$$

16

$$C_0 = w_{y_1->z_1} - w_{y_1->z_2} + w_{y_2->z_1} - w_{y_2->z_2} + w_{y_3->z_1} - w_{y_3->z_2} - b_{z_2} + b_{z_1}$$
$$C_3 = (w_{y_1->z_1} - w_{y_1->z_2} + w_{y_2->z_1} - w_{y_2->z_2} - b_{z_2} + b_{z_1})$$
$$C_2 = (w_{y_1->z_1} - w_{y_1->z_2} + w_{y_3->z_1} - w_{y_3->z_2} - b_{z_2} + b_{z_1})$$
$$C_1 = (w_{y_2->z_1} - w_{y_2->z_2} + w_{y_3->z_1} - w_{y_3->z_2} - b_{z_2} + b_{z_1})$$
$$C_4 = (w_{y_1->z_1} - w_{y_1->z_2} - b_{z_2} + b_{z_1})$$
$$C_5 = (w_{y_2->z_1} - w_{y_2->z_2} - b_{z_2} + b_{z_1})$$
$$C_6 = (w_{y_3->z_1} - w_{y_3->z_2} - b_{z_2} + b_{z_1})$$
$$C_7 = (-b_{z_2} + b_{z_1})$$

$$B_1 = -w_{x_1->y_1} * x_1 + w_{x_2->y_1} * x_2 + b_{y_1}$$
$$B_2 = -w_{x_1->y_2} * x_1 + w_{x_2->y_2} * x_2 + b_{y_2}$$
$$B_3 = -w_{x_1->y_3} * x_1 + w_{x_2->y_3} * x_2 + b_{y_3}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



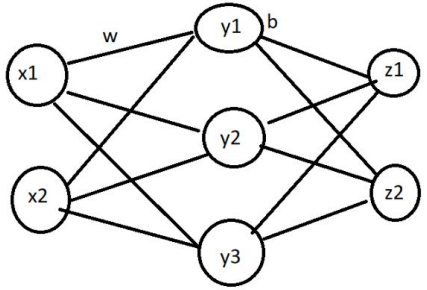
Our Third Task:

Find limits/boundaries between these plains and highlight their intersections

Given: $$\frac{dy}{dx} = ky$$

The General solution is: $$y = Ce^{kx}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Our Third Task:

Find limits/boundaries between these plains and highlight their intersections

Given: $$\frac{dy}{dx} = ky$$

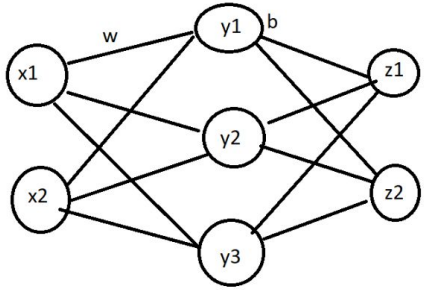
The General solution is: $$y = Ce^{kx}$$

Our equation:

$$C_0 + C_1 e^{B_1} + C_2 e^{B_2} + C_3 e^{B_3} + C_4 e^{B_2+B_3} + C_5 e^{B_1+B_3} + C_6 e^{B_1+B_2} + C_7 e^{B_1+B_2+B_3} = 0$$

And this is where we are now:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Our equation(s)

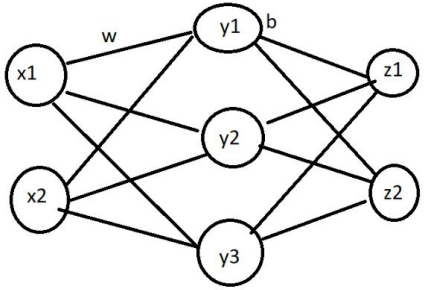
$$C_0 + C_1 e^{B_1} + C_2 e^{B_2} + C_3 e^{B_3} + C_4 e^{B_2+B_3} + C_5 e^{B_1+B_3} + C_6 e^{B_1+B_2} + C_7 e^{B_1+B_2+B_3} = 0$$

$$[I_{1-2}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\sigma(\sigma([x][w_1] + [b_1])[w_2] + [b_2])[I_{1-2}] = 0$$

And this is where we are now:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

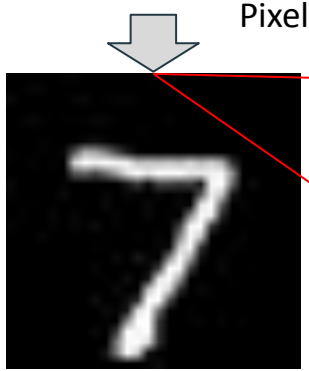


Model: GaussianNB
Accuracy: 52%

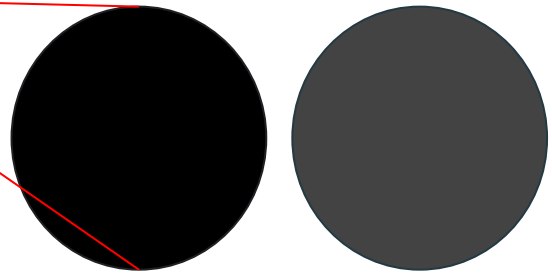
```
Modifying pixel:
Row:  0
Column: 15
Pixel_val: +1
prediction from:  9 to:  [2]
```

Original image:

Modified image:



Pixel that changed(by value : 1)



Future work:

- Work back into Diff. Eq. and use numerical methods to approximate x1 and x2… Xn
  - This will be like nested integral that we do not want to actually solve but rather approximate
  - This work moves into the Boundary Value problem which has extensive research but needs more time to understand.
- Understand Variational Spline theory
- Make an algorithm that given a feedforward NN architecture can output the points that are boundaries for the NN classifications.

Current Related Work:

A Spline Theory of Deep Networks

Randall Balestriero [1] **Richard G. Baraniuk** [1]

Abstract

We build a rigorous bridge between deep networks (DNs) and approximation theory via spline functions and operators. Our key result is that a large class of DNs can be written as a composition of *max-affine spline operators* (MASOs), which provide a powerful portal through which to view and analyze their inner workings. For instance, conditioned on the input signal, the output of a MASO DN can be written as a simple affine transformation of the input. This implies that a DN constructs a set of signal-dependent, class-specific

and then significantly improving performance over classical approaches.

Despite this empirical progress, the precise mechanisms by which deep learning works so well remain relatively poorly understood, adding an air of mystery to the entire field. Ongoing attempts to build a rigorous mathematical framework fall roughly into five camps: (i) probing and measuring DNs to visualize their inner workings (Zeiler & Fergus, 2014); (ii) analyzing their properties such as expressive power (Cohen et al., 2016), loss surface geometry (Lu & Kawaguchi, 2017; Soudry & Hoffer, 2017), nuisance management (Soatto & Chiuso, 2016), sparsification (Papyan et al., 2017), and gen-

Current Related Work:

What Kinds of Functions do Deep Neural Networks Learn? Insights from Variational Spline Theory*

Rahul Parhi† and Robert D. Nowak†

Abstract. We develop a variational framework to understand the properties of functions learned by fitting deep neural networks with rectified linear unit activations to data. We propose a new function space, which is reminiscent of classical bounded variation-type spaces, that captures the compositional structure associated with deep neural networks. We derive a representer theorem showing that deep ReLU networks are solutions to regularized data fitting problems over functions from this space. The function space consists of compositions of functions from the Banach spaces of second-order bounded variation in the Radon domain. These are Banach spaces with sparsity-promoting norms, giving insight into the role of sparsity in deep neural networks. The neural network solutions have skip connections and rank bounded weight matrices, providing new theoretical support for these common architectural choices. The variational problem we study can be recast as a finite-dimensional neural network training problem with regularization schemes related to the notions of weight decay and path-norm regularization. Finally, our analysis builds on techniques from variational spline theory, providing new connections between deep neural networks and splines.

What we learned through this process:

- Sometimes even obvious seaming questions have not yet been asked.
- It can be very hard to explain a new idea to someone without visual props.
- Sometimes it is easy to get bogged down in the work and loose light of the goal.
- Different people will have different strengths and weaknesses and those differences should be leveraged rather than dismissed.

Sources

- https://ai.plainenglish.io/neural-networks-for-beginners-b54e8e118c7
- https://www.ibm.com/watson/explainable-ai
- https://ieeexplore.ieee.org/abstract/document/1532820
- https://ieeexplore.ieee.org/abstract/document/7552539
- https://proceedings.neurips.cc/paper/2019/hash/80537a945c7aaa788ccfcdf1b99b5d8f-Abstract.html
- https://en.wikipedia.org/wiki/Explainable_artificial_intelligence
- https://www.wolframalpha.com/
- https://drive.google.com/file/d/1E5riOt3Dg_wGxwqwxixbAvKhB1-GSg4o/view?usp=share_link

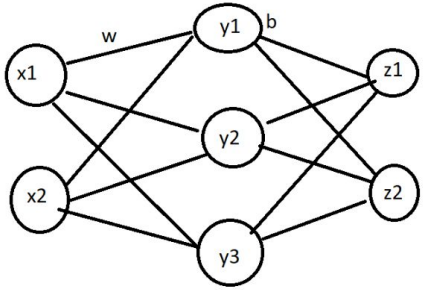
Q&A

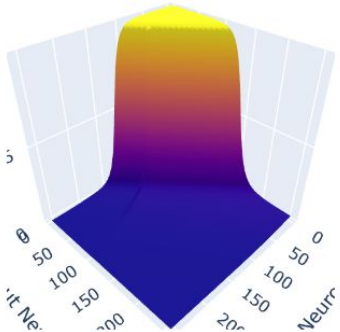
Model: GaussianNB
Accuracy: 0.52

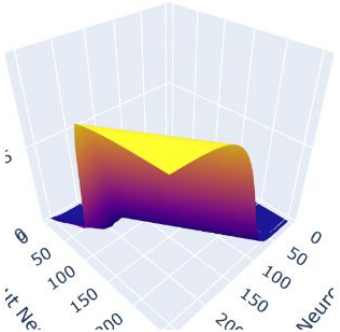
```
Modifying pixel:
Row:  0
Column: 15
Pixel_val: +1
prediction from:  9 to:  [2]
```

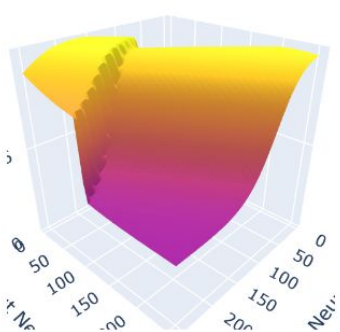
Original image:

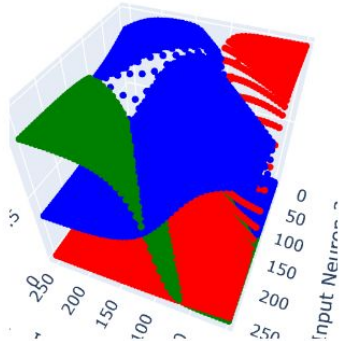
Modified image:





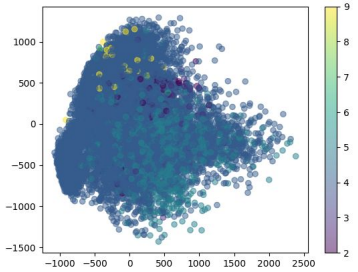


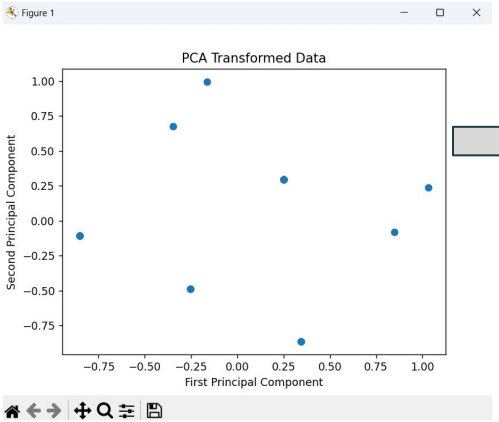




In case of dimensionality reduction question:

The neural network visualized (as feasibly as possible using dimensionality reduction)





After x.shuffle() for a 4D cube vertices