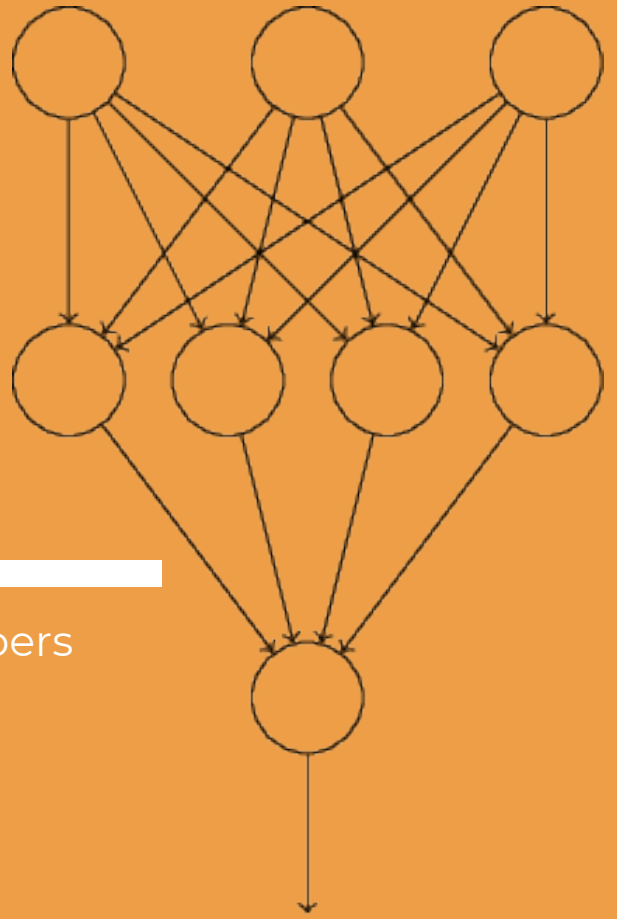
NEURAL NETWORKS

a **machine learning** classifier for handwritten numbers

by: sravya balasa



WHAT IS A **NEURAL NETWORK?**

- Take in a large amount of **training** examples
  - Greater # of training examples = Greater accuracy

- Develop a system that can learn from those examples
- Then evaluates **testing** examples

Neurons are in **layers** that **translate** information

MINIMUM VIABLE PRODUCT

1. To understand and implement a neural network (NN)
2. Using NN to classify the MNIST dataset of handwritten numbers (0-9)
3. Display the accuracy of the neural networks' classification
4. Implementation of successful **user input** for demo purposes
    a. Modifications to testing dataset

 **= 504192**

IMPLEMENTATION

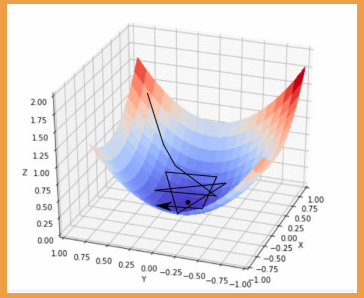
LIBRARIES

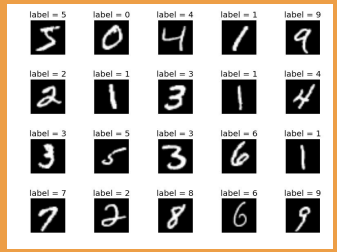
- random
- numpy
- pickle
- gzip
- sciPy → misc
- PIL → image
- json
- sys

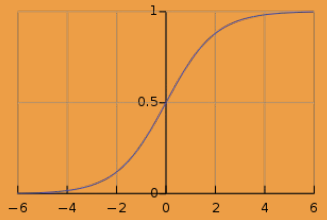
CHALLENGES

- Understanding the python functions and math used in the algorithm
- Finding the individual output from each image that contributes to the accuracy rate
- Discovering the exact pattern of the data structures of the input
- Implementing user input facility
- Implementing new libraries

RESULTS & METHODS





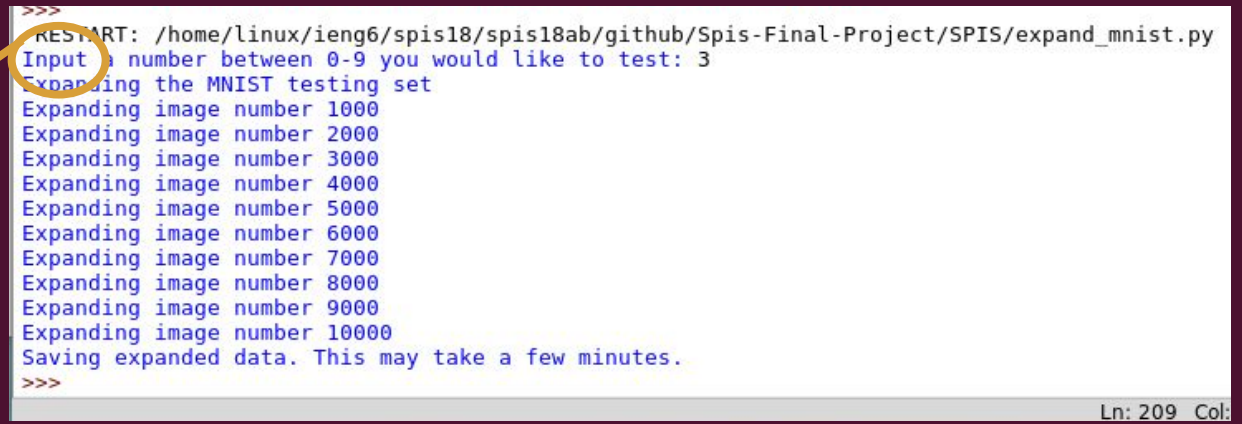


WHAT'S THE USER'S INPUT?

1. transforms an image to 28*28 image
2. transforms it to a numpy (matrix) array
3. adds it to the **testing data**
4. evaluates!

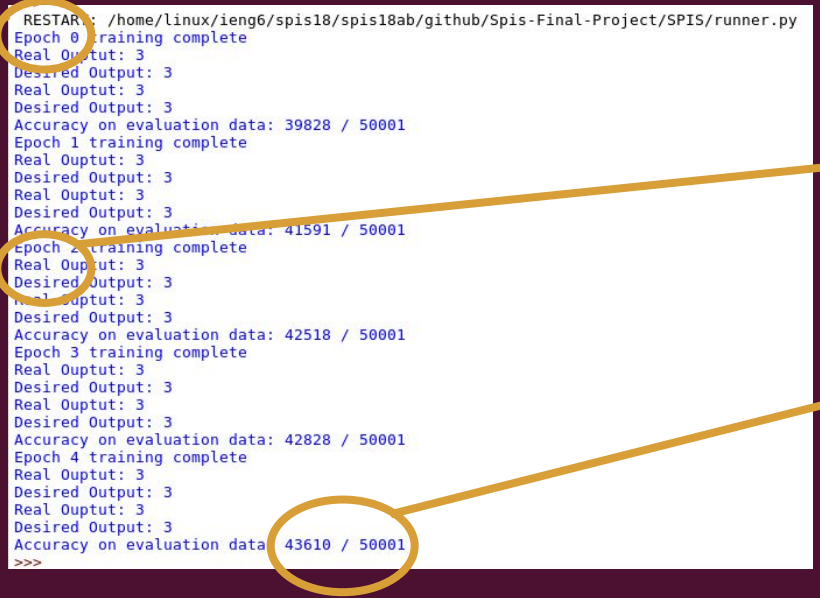
WHAT'S THE USER'S INPUT?

USER INPUT:
asks for which image should be processed



WHAT'S THE OUTPUT?

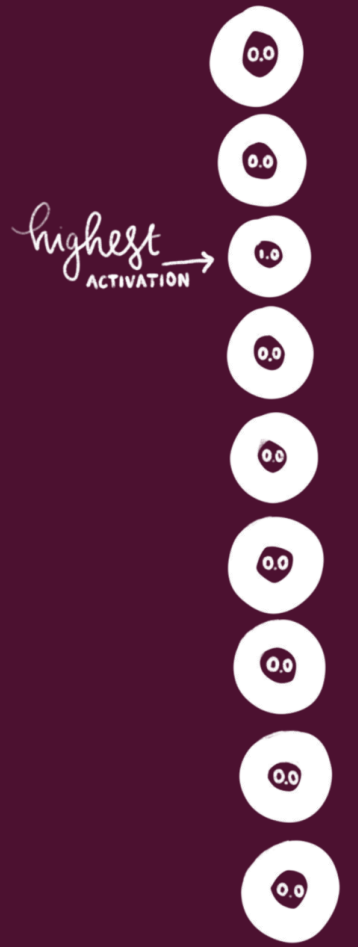
EPOCH:
one full training of network



REAL:
result of user's input after run through network
DESIRED:
user's predicted result

ACCURACY:
-number of testing data inputs classified correctly
-accuracy increases with each epoch

ACTIVATION ENERGIES



- Activation energies only adjusted by **weights** and **biases**
  - Allows for slow adjustment
- Last layer = desired output layer
- **Ex:** 3rd neuron has activation 1.0, inputted number is a 2

BACKPROPAGATION

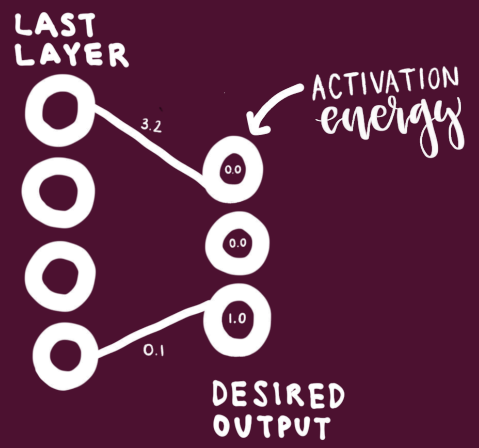
THE ALGORITHM ITSELF.

- How do changes in weights, biases, activation affect final activation?
- **Each layer depends on previous...**
  - Adding desired effects to get to the output layer
- Ex: $a_{desired} = a_{desired-1} * w$

POSSIBLE CHANGES: WEIGHTS, BIASES, INITIAL ACTIVATION

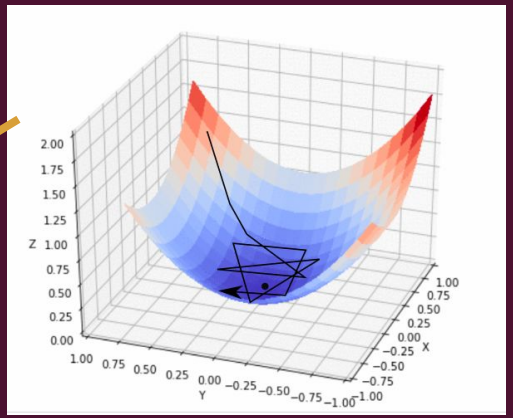
COST FUNCTION

- Measures **changes in activation** from one layer to the next
  - COST of this DIFFERENCE = GRADIENT
- Ex: Between last layer and desired output layer
- Goal:
  - Minimize the cost through **GRADIENT DESCENT**



STOCHASTIC GRADIENT DESCENT

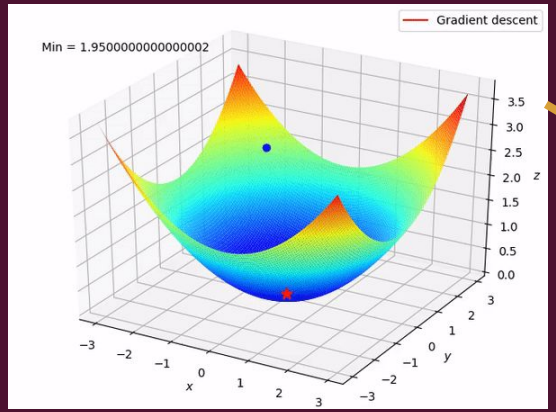
SGD



GRADIENT

=

Effect on cost



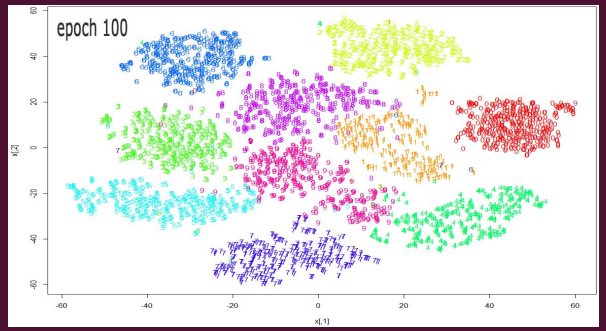
NORMAL

- Division into mini-batches
- **Effect on cost** computed each batch
- Total cost adjusted for each batch
- **Less accurate + Fast →** Local min

- **More accurate + slow →** Local min
- Careful because takes in ALL training data in one batch

FURTHER DEVELOPMENT

1. Segmentation of numbers and analyzing letters
2. Save and load a trained network
3. Creating a **tSNE** display



neural network