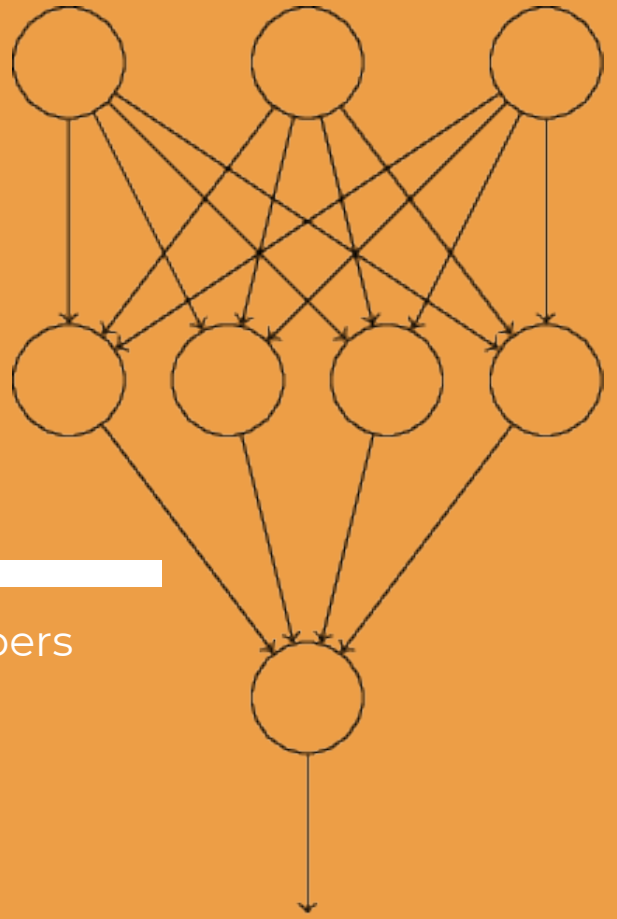
NEURAL NETWORKS

a **machine learning** classifier for handwritten numbers

by: sravya balasa



WHAT IS A **NEURAL NETWORK?**

- Takes in large amount of **training** examples
  - More examples = More accuracy
- ∴ system can train from those examples
- Then evaluates **testing** examples
- 30 neurons in hidden layer is optimal (accuracy + speed)

784 neurons in first layer → 9 neurons in last layer

MINIMUM VIABLE PRODUCT

1. Implement a neural network (NN)
2. **Use NN to classify the MNIST dataset of handwritten numbers (0-9)**
3. Display the **accuracy** of the neural networks' classification
4. Implement successful **user input**
    a. AKA modifications to testing dataset

 **= 504192**

CHALLENGES

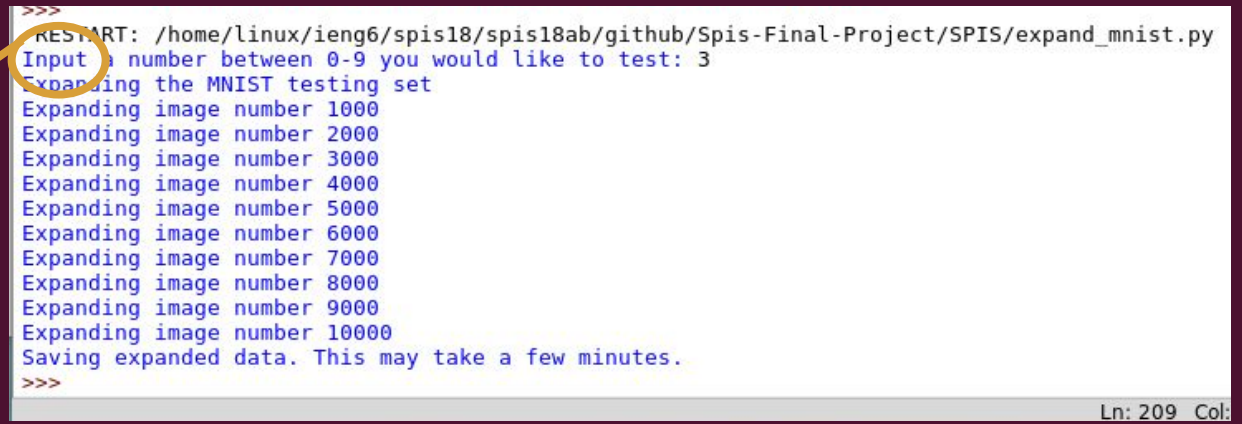
- Understanding the python functions and math used in the backpropogation algorithm
- Finding the individual output from each image that contributes to the accuracy rate
- Discovering the exact pattern of the data structures of the input
- Implementing user input facility
- Implementing new libraries

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!

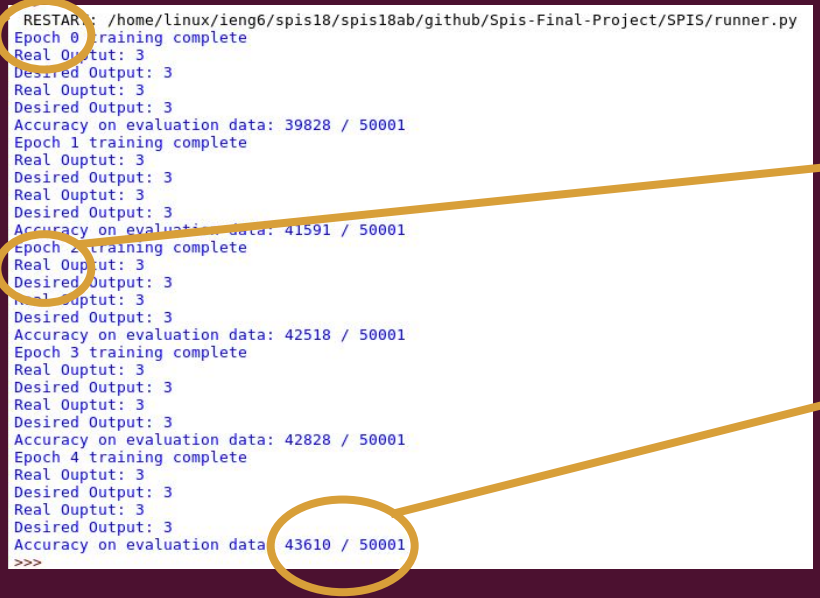
WHAT'S THE USER'S INPUT?

USER INPUT:
asks for which image should be processed



WHAT'S THE OUTPUT?

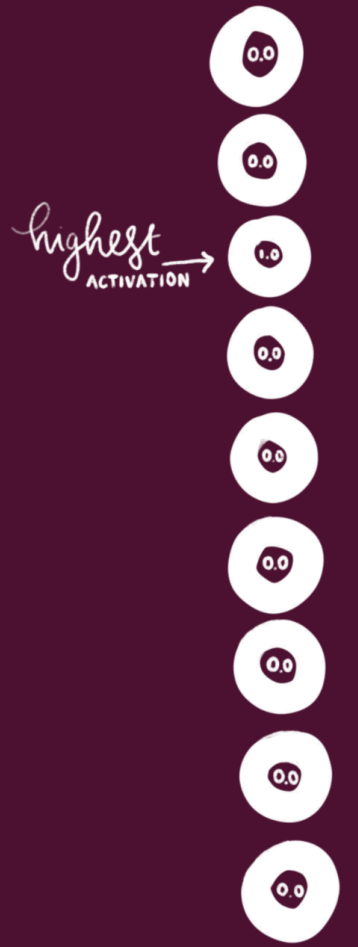
EPOCH:
one full training of network



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
- REACHED 98%

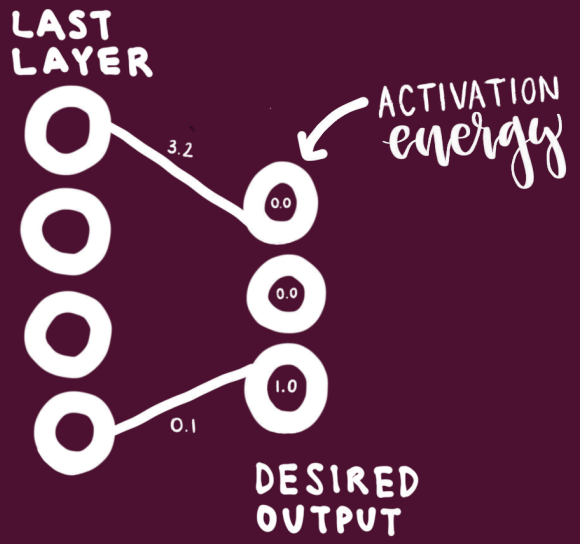
ACTIVATION ENERGIES



- Activation energies only adjusted by **weights** and **biases**
  - Allows for slow adjustment between each hidden layer
- Last layer = desired output layer
- **Ex:** 3rd neuron has activation 1.0, inputted number is a 2
  - Network aims to always add reach one highest value

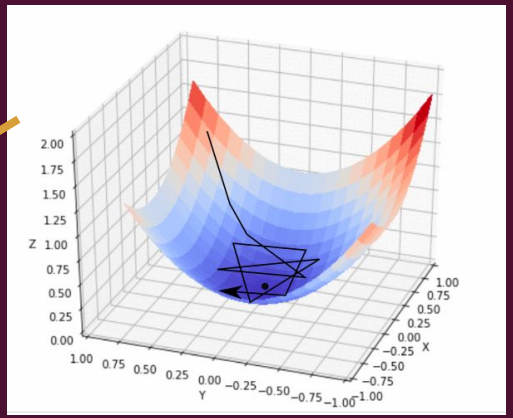
COST FUNCTION

- Measures **changes in activation** from one layer to the next
- Ex: Between last layer and desired output layer



STOCHASTIC GRADIENT DESCENT

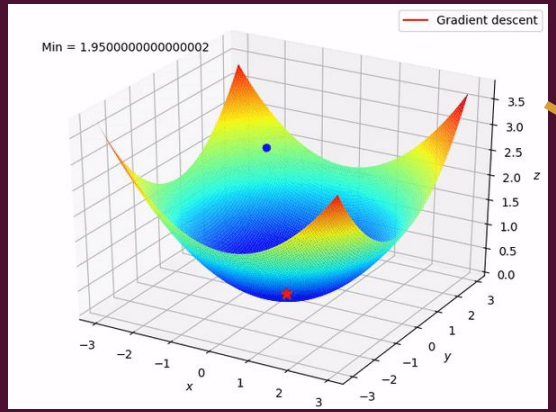
SGD



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

BACKPROPAGATION

THE ALGORITHM ITSELF.

- How do changes in weights, biases, activation affect final activation?
- Which **weight** changes decreases cost the most + gets our **final output**?

POSSIBLE CHANGES: WEIGHTS, BIASES, INITIAL ACTIVATION

CITATIONS

1. NIELSEN'S GITHUB
2. MNIST DATASET
3. NIELSEN'S BOOK: NEURAL NETWORKS AND DEEP LEARNING