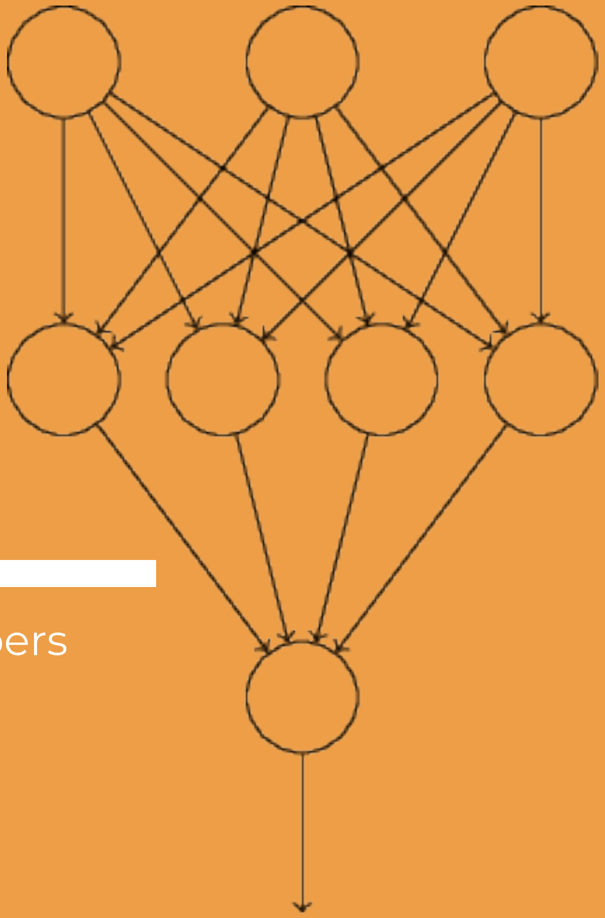
NEURAL NETWORKS

a **machine learning** classifier for handwritten numbers
by: sravya balasa



WHAT IS A **NEURAL NETWORK?**

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

784 pixels = # neurons in first layer → 10 numbers = # neurons in last layer

MINIMUM VIABLE PRODUCT

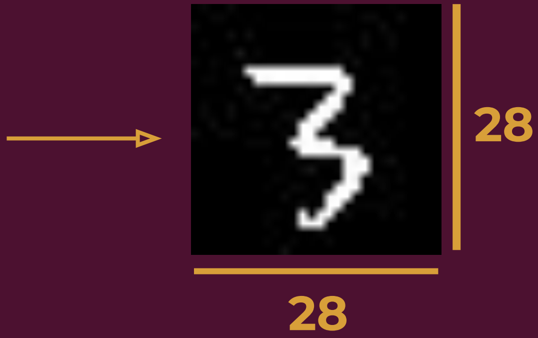
1. Implement a neural network (NN)
2. **Use NN to classify the MNIST dataset of handwritten numbers (0-9)**
3. Display the **accuracy** of the neural networks' classification
4. Implement <u>successful</u> **user input**
   a. AKA modifications to testing dataset

= 504192

CHALLENGES

- Understanding functions & math used in the algorithm
- Finding the individual output from each image that contributes to the accuracy rate
  - Mimicking structure of testing dataset
- Implementing user input facility
- Implementing new libraries

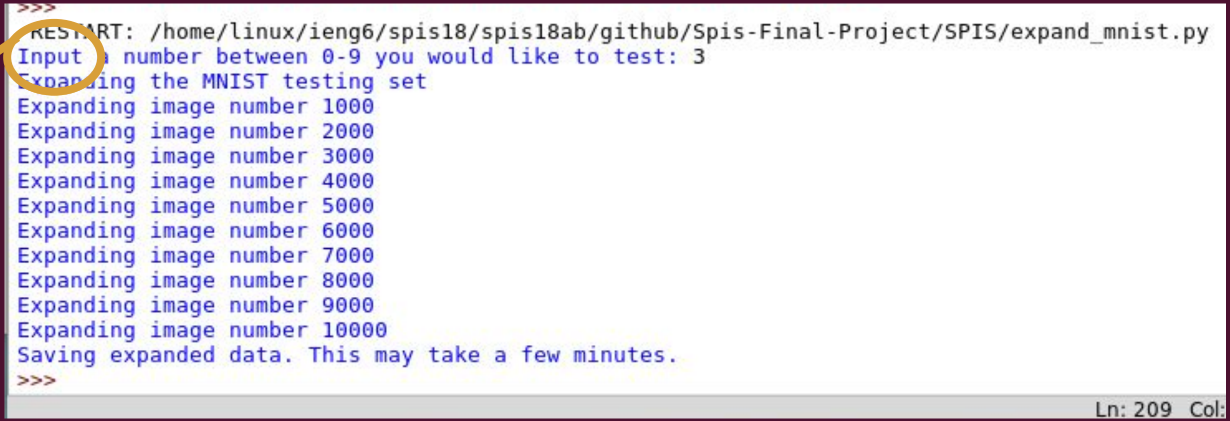
WHAT'S THE USER'S INPUT?



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?

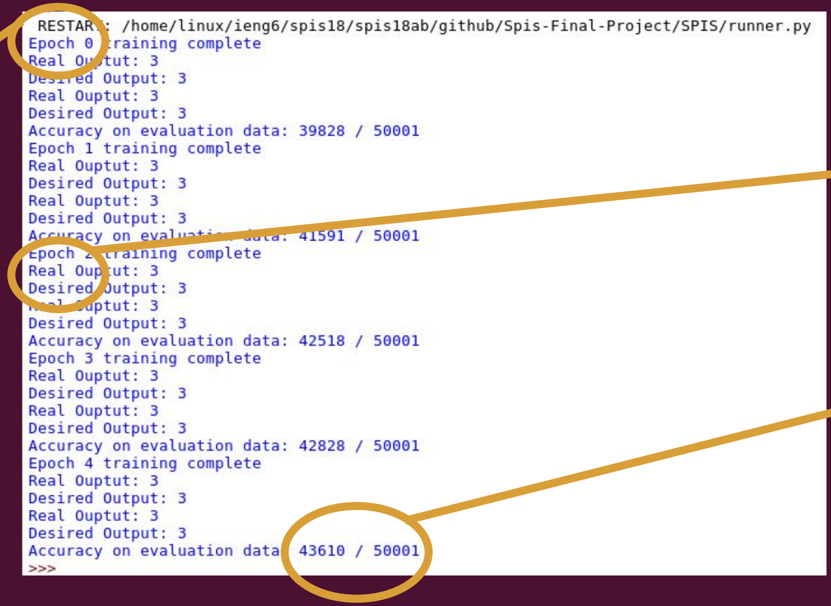
USER INPUT:
asks for which image
should be processed



```
>>>
RESTART: /home/linux/ieng6/spis18/spis18ab/github/Spis-Final-Project/SPIS/expand_mnist.py
Input a number between 0-9 you would like to test: 3
Expanding the MNIST testing set
Expanding image number 1000
Expanding image number 2000
Expanding image number 3000
Expanding image number 4000
Expanding image number 5000
Expanding image number 6000
Expanding image number 7000
Expanding image number 8000
Expanding image number 9000
Expanding image number 10000
Saving expanded data. This may take a few minutes.
>>>
```

WHAT'S THE OUTPUT?

EPOCH:
one full training of network

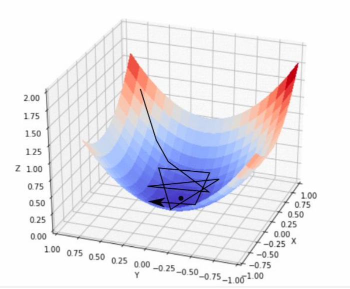


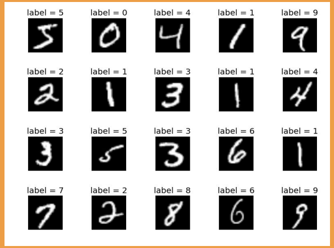
```
 RESTART: /home/linux/ieng6/spis18/spis18ab/github/Spis-Final-Project/SPIS/runner.py
Epoch 0 training complete
Real Ouptut: 3
Desired Output: 3
Real Ouptut: 3
Desired Output: 3
Accuracy on evaluation data: 39828 / 50001
Epoch 1 training complete
Real Ouptut: 3
Desired Output: 3
Real Ouptut: 3
Desired Output: 3
Accuracy on evaluation data: 41591 / 50001
Epoch 2 training complete
Real Ouptut: 3
Desired Output: 3
Real Ouptut: 3
Desired Output: 3
Accuracy on evaluation data: 42518 / 50001
Epoch 3 training complete
Real Ouptut: 3
Desired Output: 3
Real Ouptut: 3
Desired Output: 3
Accuracy on evaluation data: 42828 / 50001
Epoch 4 training complete
Real Ouptut: 3
Desired Output: 3
Real Ouptut: 3
Desired Output: 3
Accuracy on evaluation data: 43610 / 50001
>>>
```

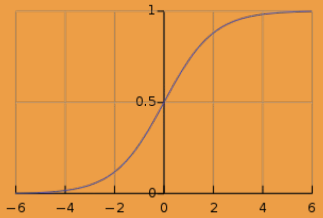
REAL:
result of user's input after run through network
DESIRED:
user's predicted result

ACCURACY:
-number of testing data inputs classified correctly
-accuracy increases with each epoch
- REACHED 98% !!

THE ALGORITHM

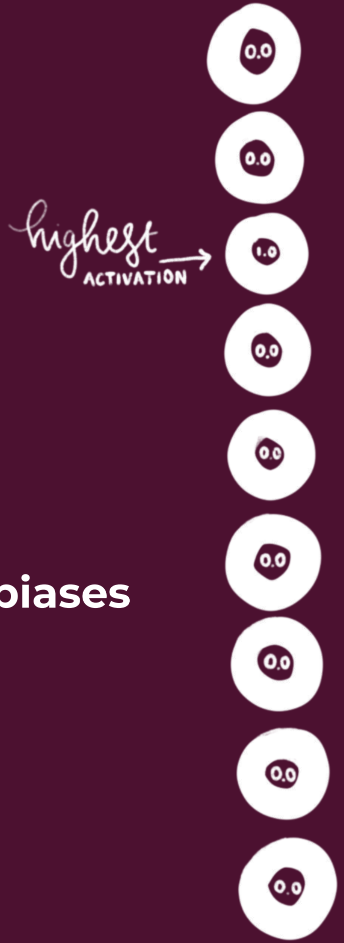






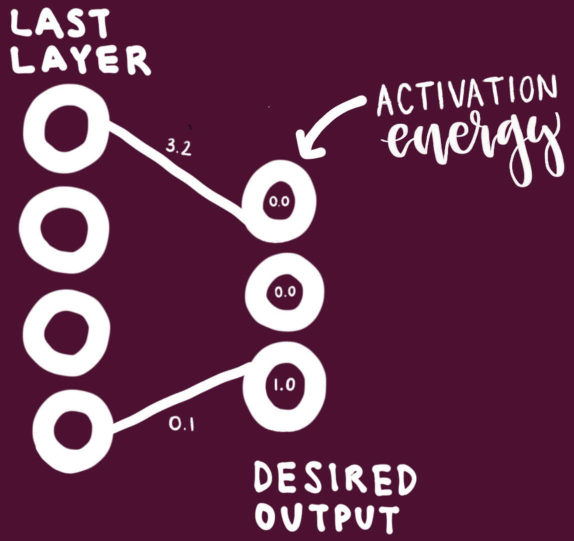
ACTIVATION ENERGIES

- Last layer = Determines network's output
  - 1.0 = highest ; 0.0 = lowest
  - Will output highest based on pixel analyzation
- Activation energies in layers slowly adjusted by **weights** and **biases**
- **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



BACKPROPAGATION

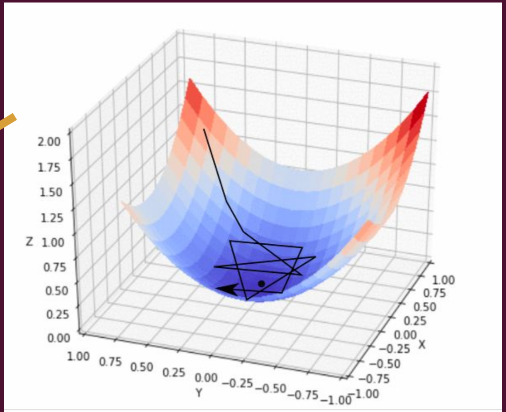
THE ALGORITHM ITSELF.

- How do changes in weights, biases, activation affect final activation?
- Which **weight** changes decreases cost the most + gets our **final output**?
- Therefore, retrace backwards to find the optimal W,B,A_i

POSSIBLE CHANGES: WEIGHTS, BIASES, INITIAL ACTIVATION

STOCHASTIC GRADIENT DESCENT

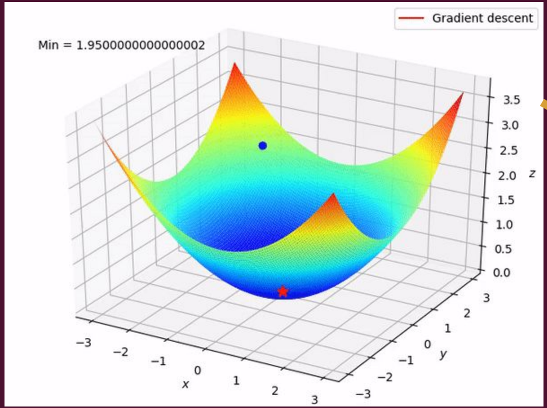
SGD



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

Machine Learning & Neural Network

CITATIONS

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