



University of Idaho

College of Engineering

UI DATA SCIENCE COMPETITION

BY: UMAIR MOHAMMAD

SCOPE



I Introduction

I Data Description

- Training Set
- Testing Set (Arithmetic Expressions for Grading)

I Neural Networks

- Loss Function
- Gradient Descent
- Stochastic mini-batch GD
- Convolutional Neural Nets and other Optimizers

I Results

I Conclusion

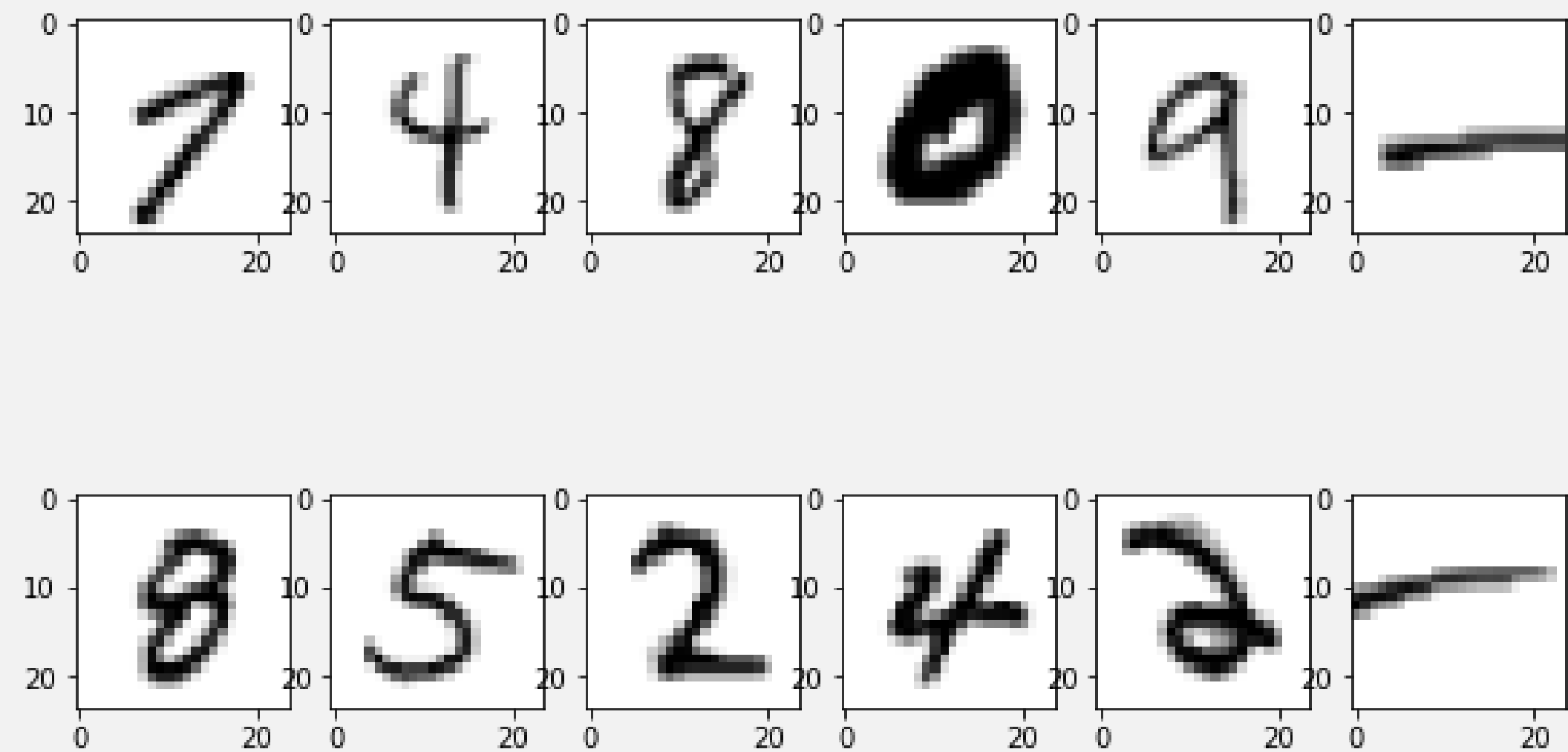
TASK DESCRIPTION



- UI Data Science Competition
- Correctly grade mathematical expressions
 - arithmetic sums and subtractions.
- Train using digits and arithmetic operators.
- Use the trained model to grade mathematical expressions accurately.
- The data was provided by the competition organizers and was available on the competition web-page
- <https://dscomp.ibest.uidaho.edu/>

DATA (TRAINING)

- Digits from 0-9 and arithmetic operators '+', '-', and '='
- Classes assigned from 0-9 and 10, 11, & 12, respectively.
- A total of 13 classes
- The correct class labels were provided as numbers
- Converted to one-hot vectors

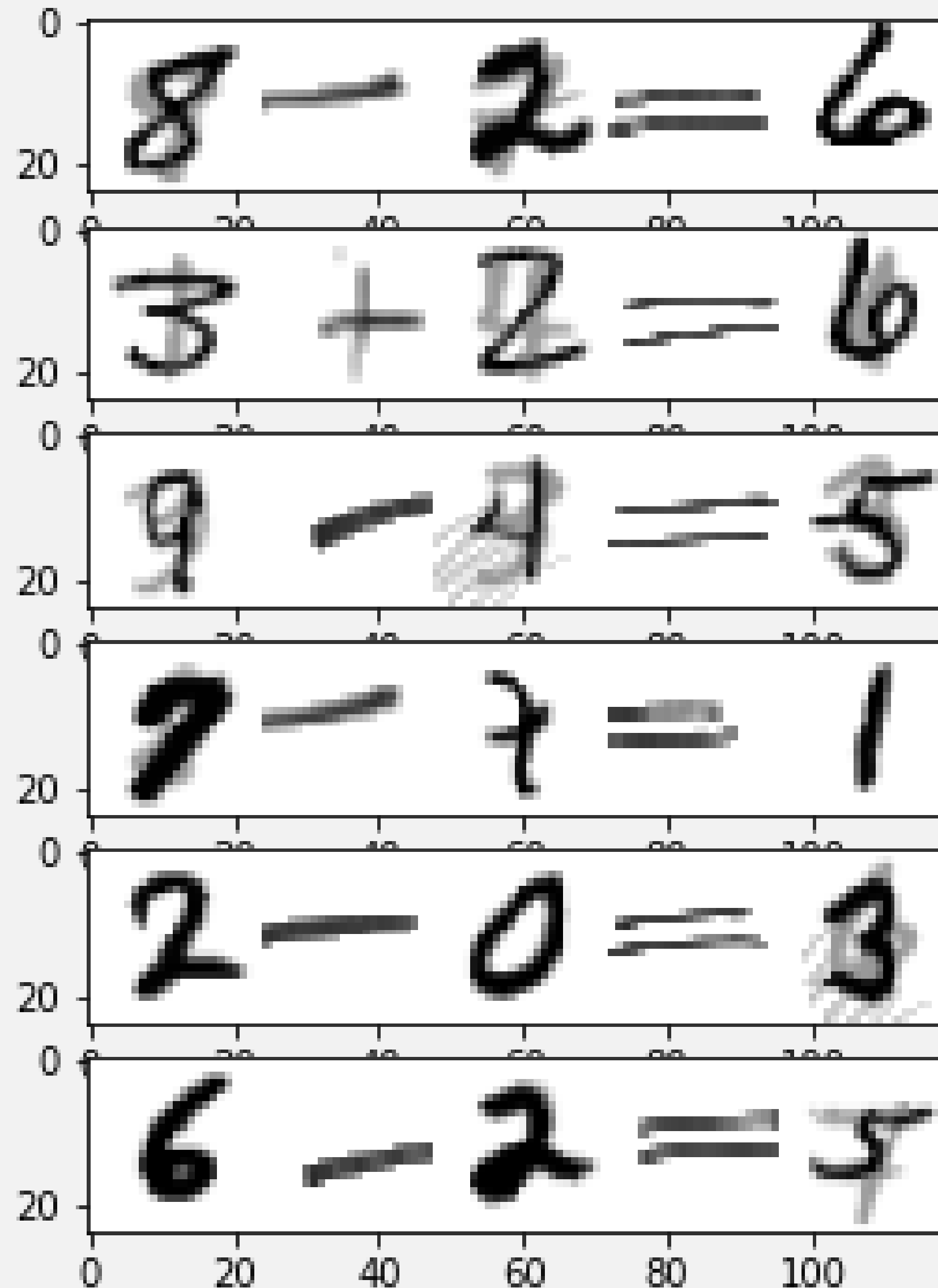
[illegible]

DATA (TESTING)



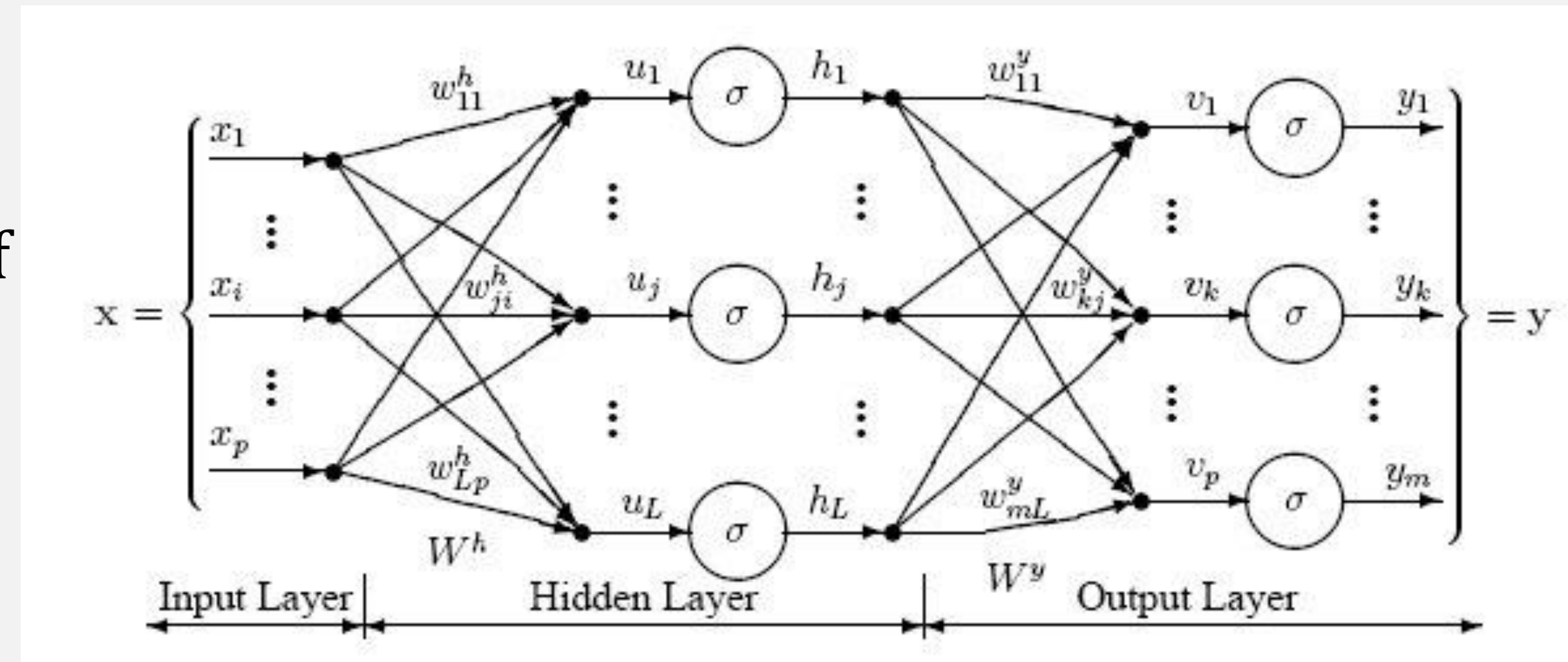
- Mathematical expressions
- In the forms
 - $a+b=c$
 - $a-b=c$
 - $a=b+c$
 - $a=b-c$
- Some correct (e.g. $5=4+1$) and some wrong ($5-3=1$)
- Based on the trained model, grade correct as 1 and grade wrong expression as 0

EXAMPLES OF EXPRESSIONS



USING NEURAL NETWORKS

- Contain an input layer, an output layer and multiple hidden layers
- The output at each stage is the sum of the weighted inputs passed through an 'activation function'
- Backpropagation to optimize the weights with gradient descent (GD)
- Inputs are gray levels
- Outputs equal number of classes
- Convolutional layers



<https://www.dtreg.com/solution/view/21>

MINI-BATCH STOCHASTIC GD (SGD) APPROACH



I Apply the GD approach to mini-batches of data

I The total dataset (epoch) is:

- randomly shuffled
- split into batches of size N
- The GD method is applied to each batch

I For training, we leave 20% for validation

I The cross-entropy is used due to the soft-max activation at the last layer (outputs probabilities)

I $\text{softmax}(x_n | x_n \in x = [x_1, \dots, x_n, \dots, x_N])$

$$= \frac{e^{-x_n}}{e^{-x_1} + e^{-x_2} + \dots + e^{-x_n} + \dots + e^{-x_N}}$$

TESTING STRATEGY



- I We know that images 1, 3 and 5 are digits
- I So, for any given test image
- I $Digit\ n = \arg \max P_{nm}^0, m = 0, \dots, 9$
- I $Operator\ n = \arg \max P_{nm}^0, m = 10, \dots, 12$
- I Take care of some other cases
- I Make sure that the expression is correct
- I Four cases: equal on position '2' or '4', '+' or '-' the other operators
- I Grade '0' if wrong and '1' if correct

RESULTS



- Best accuracy of 98.86%
- Using 2 convolutional layers and 1-layer neural network as a classifier with ADAM optimizer
- Gradient Descent alone gave up to 94% accuracy with 3-layers
- Momentum with CNN gave up to 96% accuracy
- ADAM with FCNN gave around 96% accuracy whereas three Convolutional layers and 1-3 FCNN gave around 98% accuracy
- The best accuracy came with augmentation
- CNN (3x3x36) + max. pooling (2) + CNN (3x3x48) + max. pooling (2) + CNN (3x3) + 12x288 + 288 x13

CONCLUSION



- Participated in the UI Data Science Competition as the course project
- Used neural nets, CNN's and various optimizers to classify and grade math quizzes correctly
- Focused on different optimizers for the course
- A best accuracy of 98.86% using augmented data

FUTURE WORK



- See how to get 1.0!!!
- Actual accuracy for just 20,000 equations was 99.41% (achieving this is a goal)
- Use an external server or GPU to try more complex networks
- Larger output channels in the CNN and apply dropout or batch normalization
- Try more sophisticated forms of augmentation

THANK YOU AND QUESTIONS

