GENETIC LEARNING

Deep Learning Model Optimization

Team members(SD_2)

- Timothy Schommer
- Carter wunsch
- Junran Zhang
- Malik Bulur
- Tao Li

Outline

1. Problem & Solution & History
2. Deep Learning & Genetic Algorithm
3. Software Tools
4. Implemented Algorithm
5. Demo
6. Results
7. Limitations
8. Future Works

The problem being addressed

CNN is one of the most effective and popular tool in fields of vision, recognition and others, But:

- Hard to pick the best hyperparameters
- Large amounts of computation
- Low efficiency

Proposed solution

- Algorithm Chosen: Genetic Algorithms
- Implement a genetic algorithm that can help finding optimal parameters for Deep Learning Convolutional neural networks.
- Selecting CNN
  - the number of convolutional layers
  - number of filters
  - Number of pulling layers
- Select function for ga
  - Validation loss of the model
  - Number of trained parameters

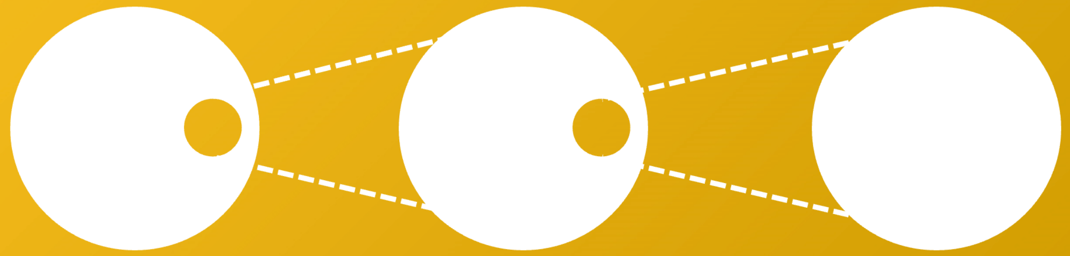
Our progress

- Week 1-2: Genetic Algorithm with one object function
- Week 3-4: Genetic Algorithm with two objects functions
- Week 5-6: Deep Learning
- Week 7-9: Integration of GA with DL (filters)
- Week 10-11: Increasing complexity of Algorithm (loss and parameters)
- Week 10-11: Testing on vermeer Data & Improving Algorithm

Deep Learning

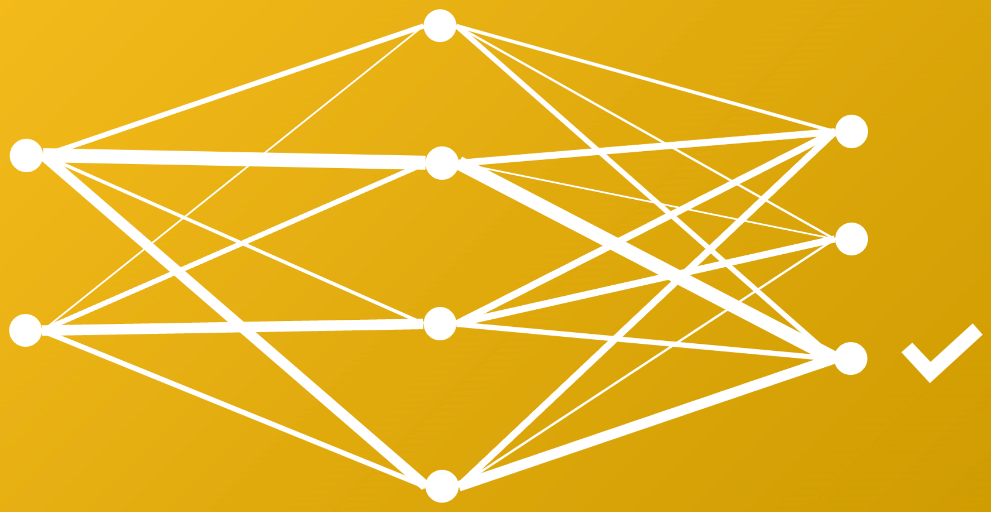
- AI, Machine Learning, Deep learning
- Artificial Neural Network

Artificial Intelligence Machine Learning Deep Learning



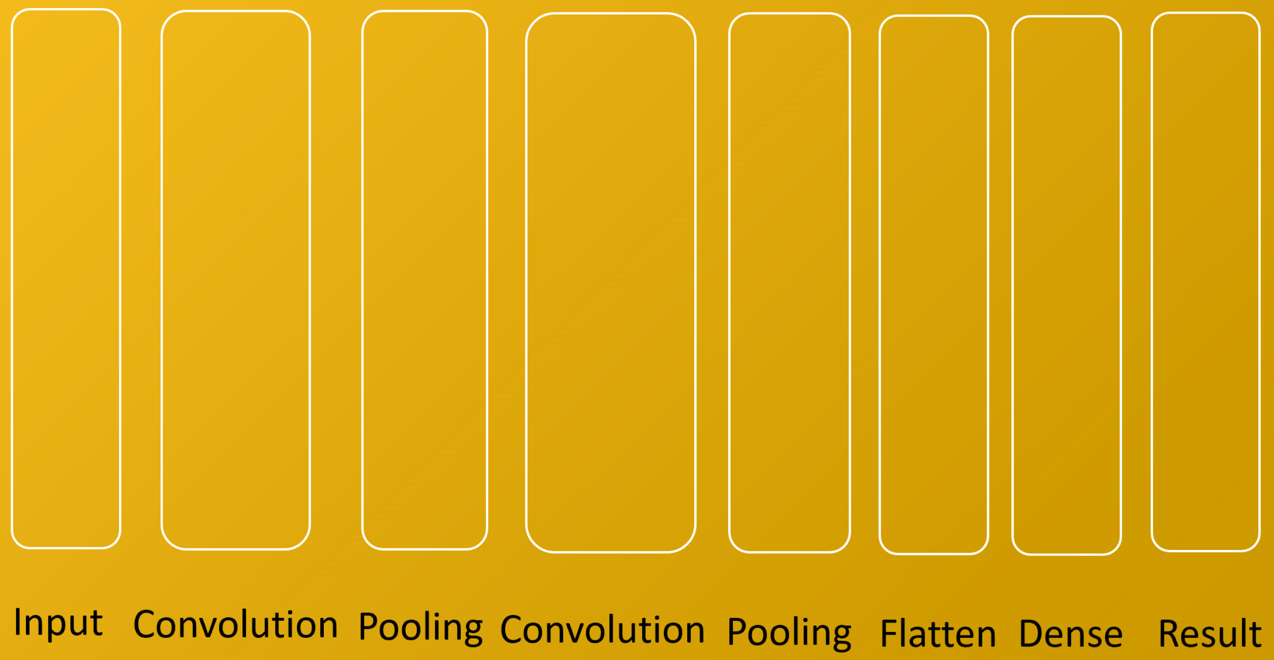
Deep Learning

- AI, Machine Learning, Deep learning
- Artificial Neural Network
    - Training



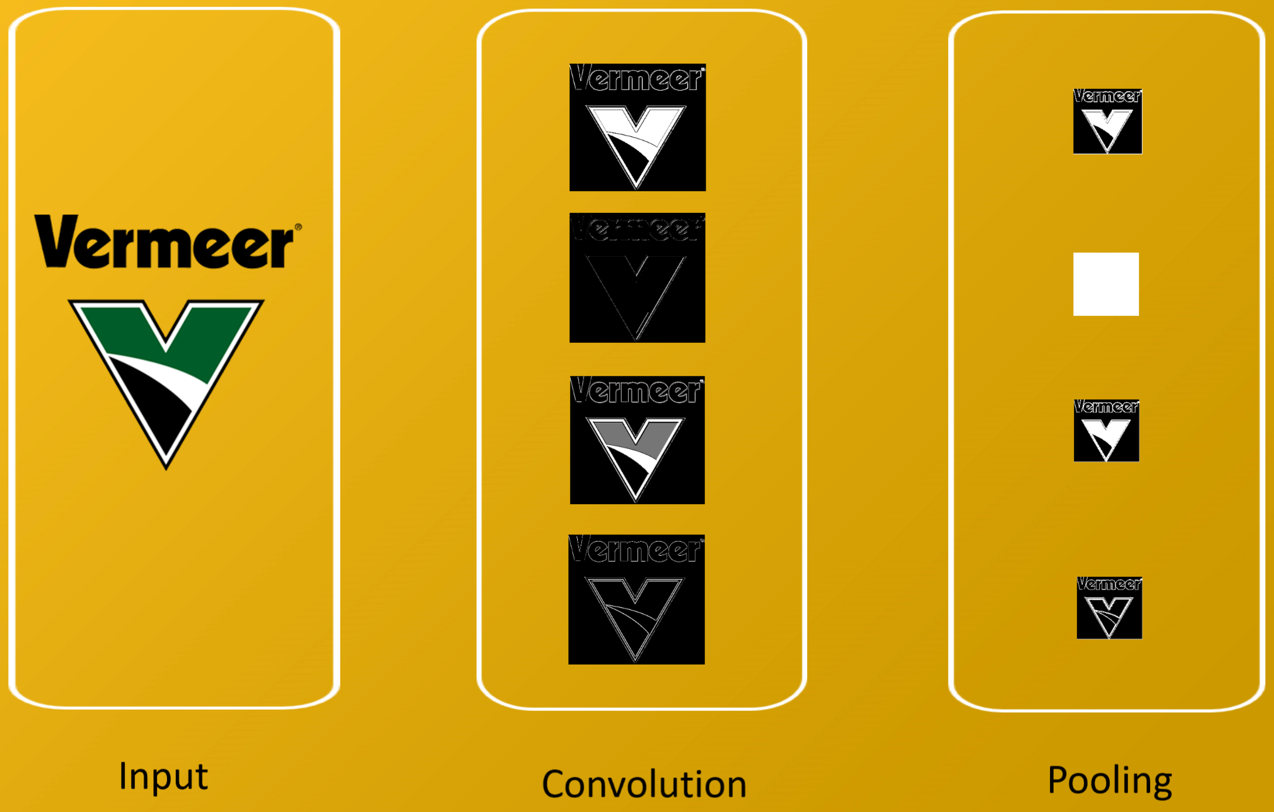
CNN

- Convolutional Neural Network (CNN)
- Classification problem using CNN



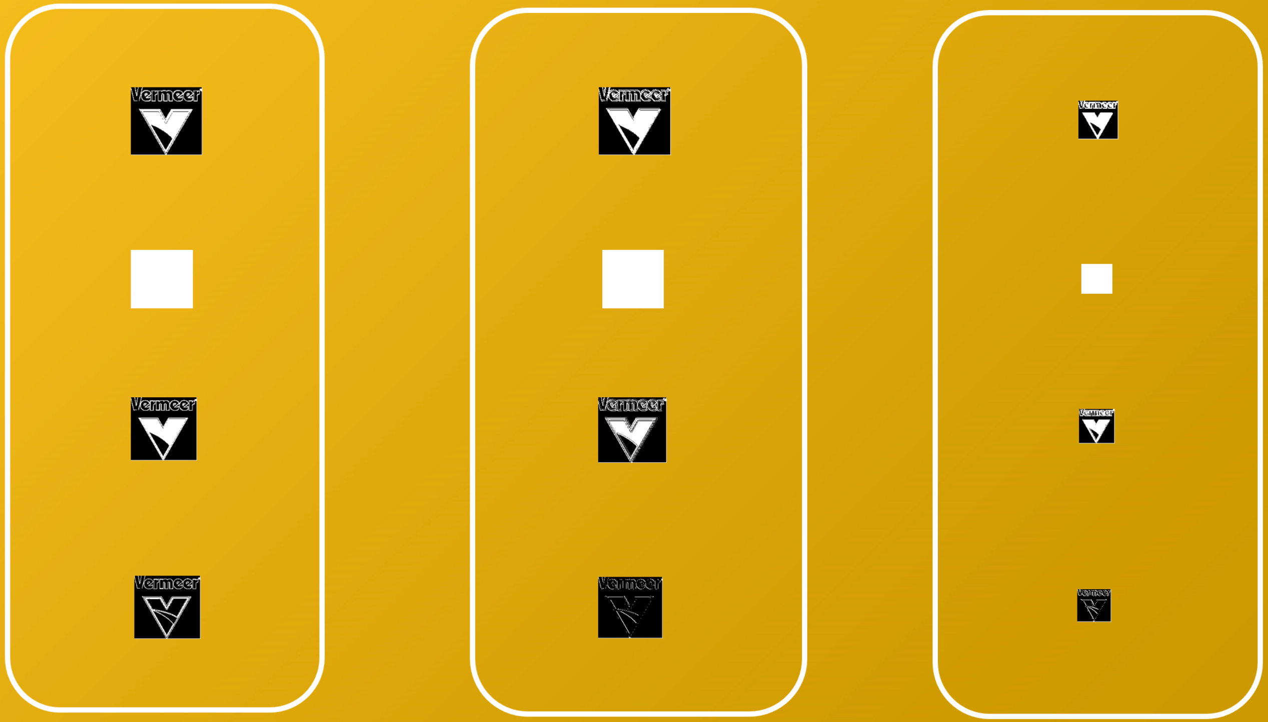
CNN

- Convolutional Neural Network (CNN)
- Classification problem using CNN



CNN

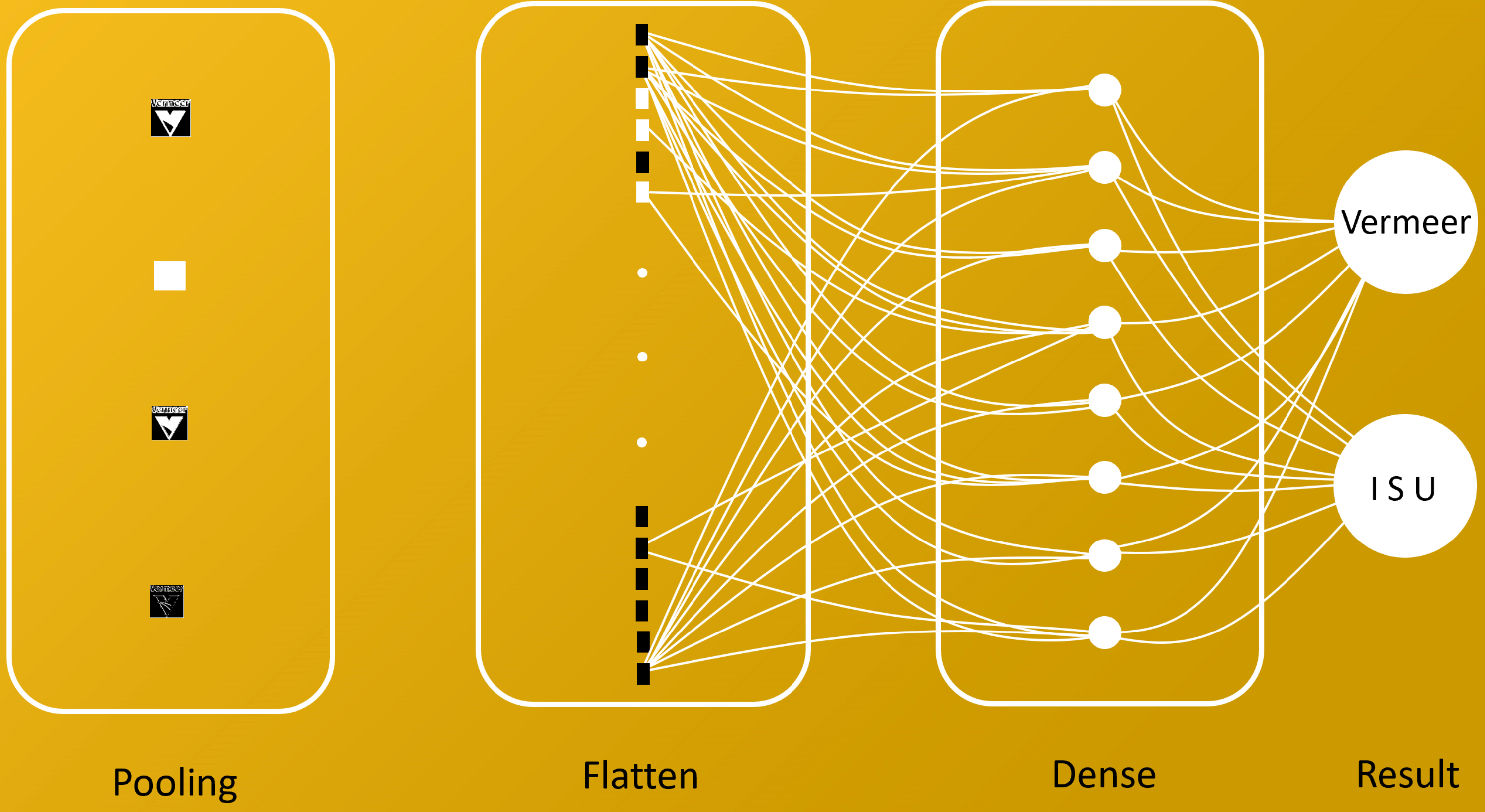
- Convolutional Neural Network (CNN)
- Classification problem using CNN



Pooling Convolution Pooling

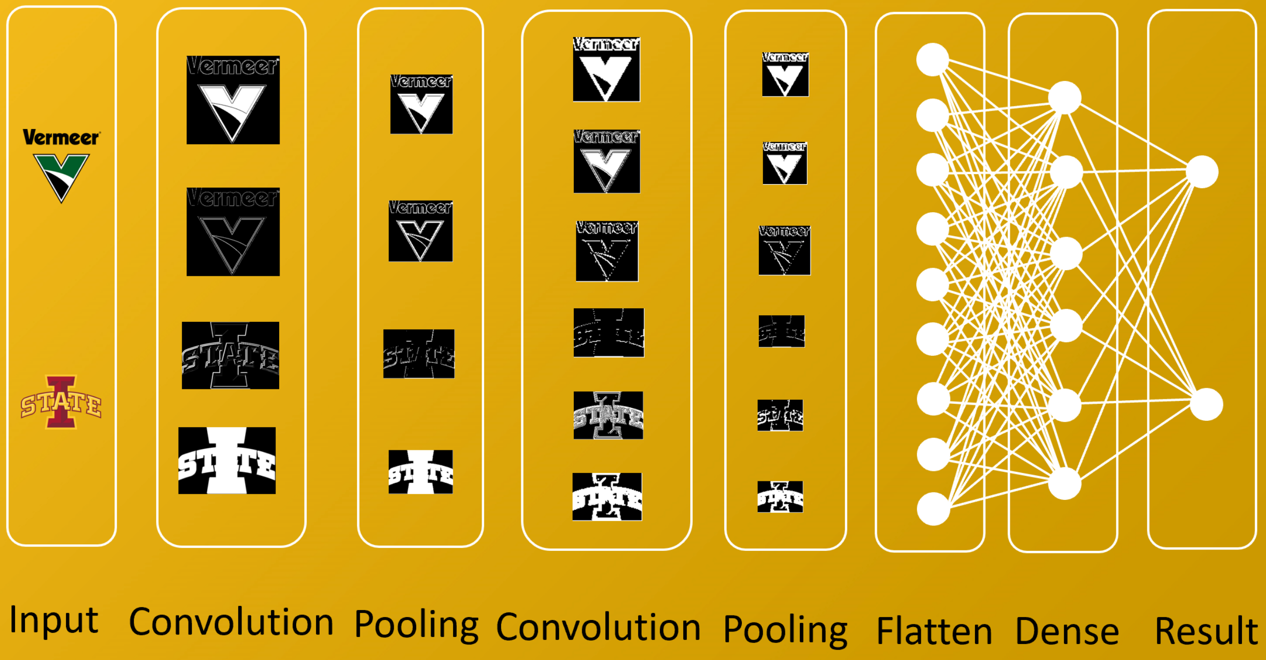
CNN

- Convolutional Neural Network (CNN)
- Classification problem using CNN



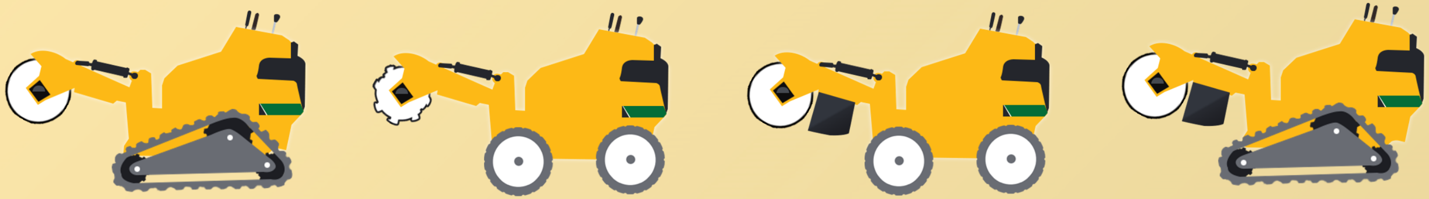
CNN

- Number of Filters
- Kernel Size
- Pool Size
- Number of Units



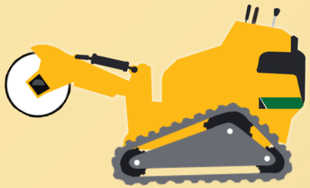
Genetic Algorithm

Initialization ➡ Evaluation ➡ Selection ➡ Crossover ➡ Mutation ➡ Done



Genetic Algorithm

Initialization ➡ Evaluation ➡ Selection ➡ Crossover ➡ Mutation ➡ Done



12%

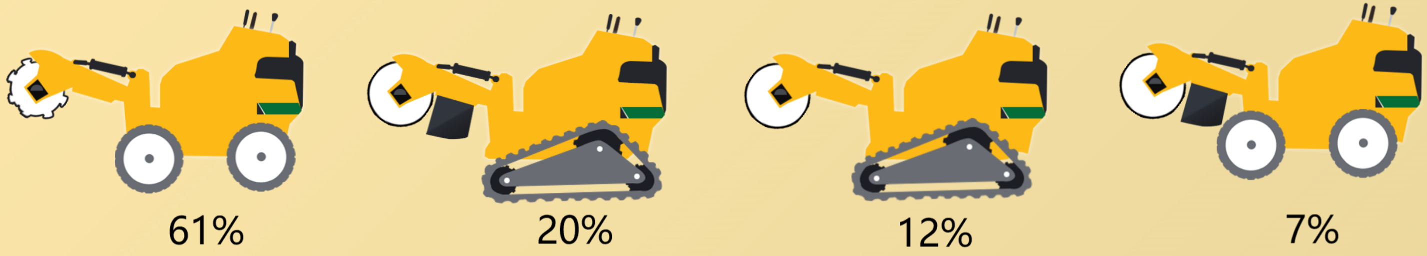
61%

7%

20%

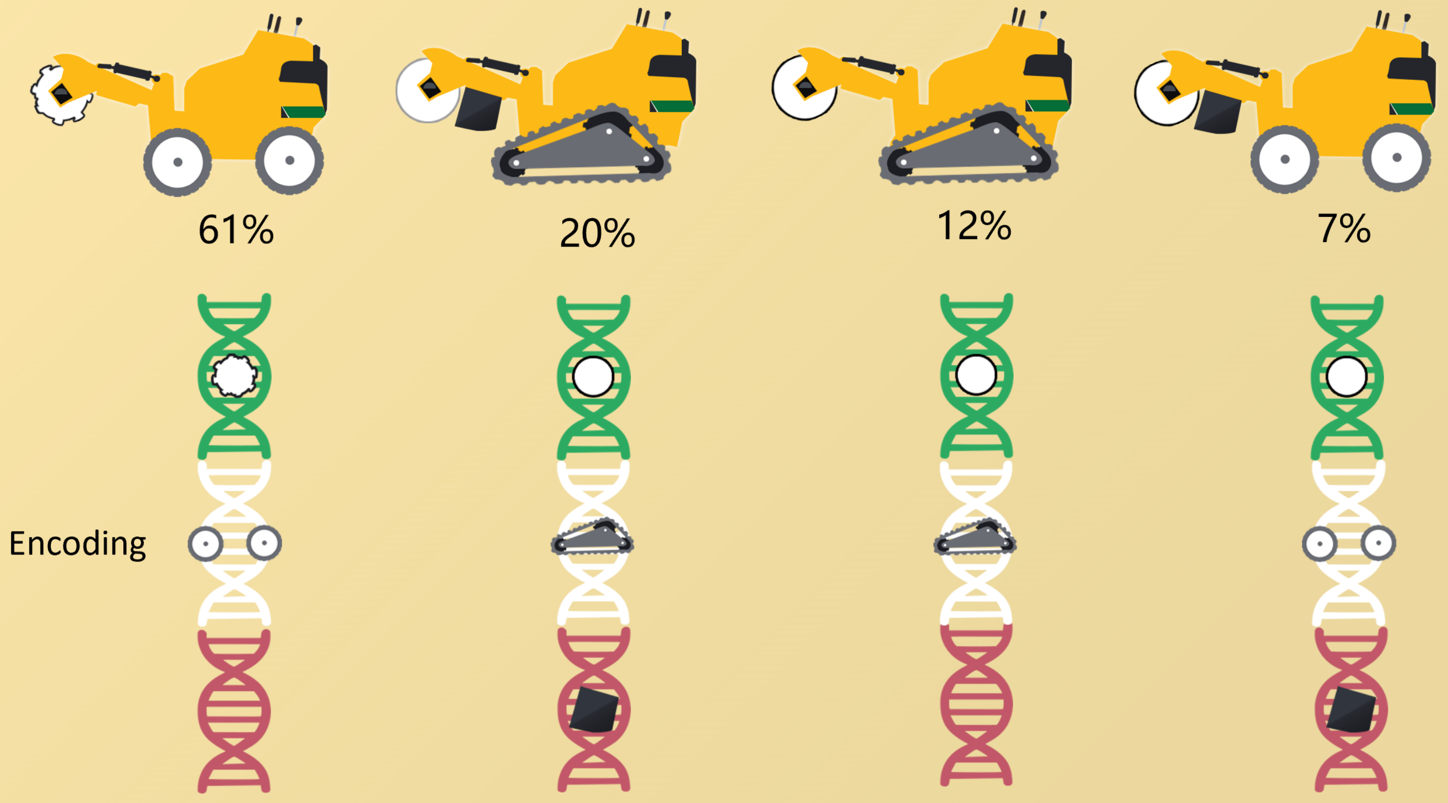
Genetic Algorithm

Initialization ➡ Evaluation ➡ Selection ➡ Crossover ➡ Mutation ➡ Done



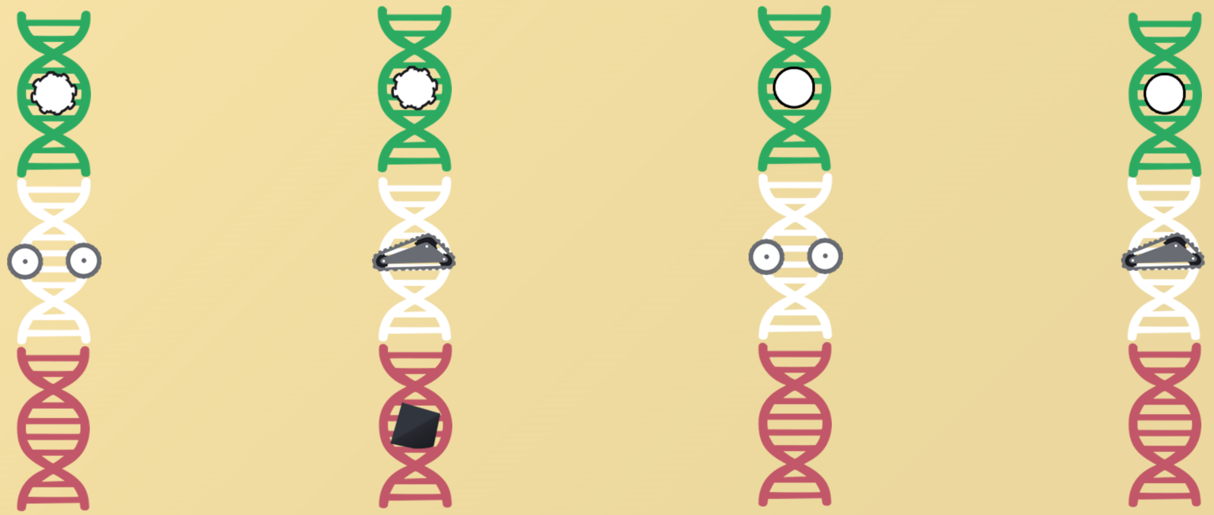
Genetic Algorithm

Initialization ➡ Evaluation ➡ Selection ➡ Crossover ➡ Mutation ➡ Done



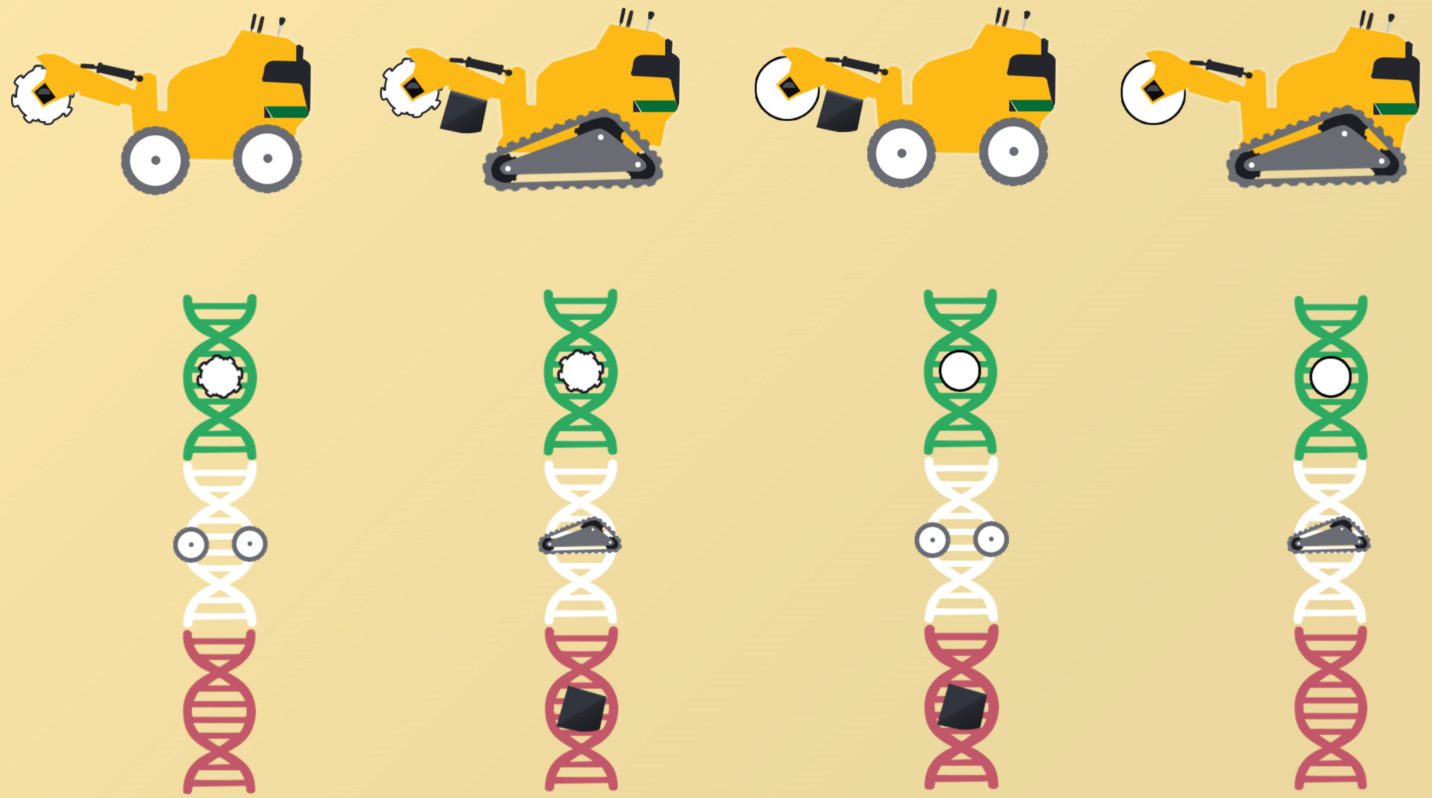
Genetic Algorithm

Initialization ➡ Evaluation ➡ Selection ➡ Crossover ➡ Mutation ➡ Done



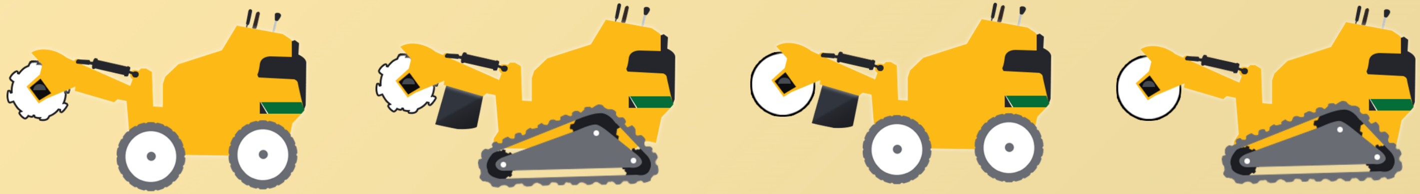
Genetic Algorithm

Initialization ➡ Evaluation ➡ Selection ➡ Crossover ➡ Mutation ➡ Done



Genetic Algorithm

Initialization → Evaluation → Selection → Crossover → Mutation → Done



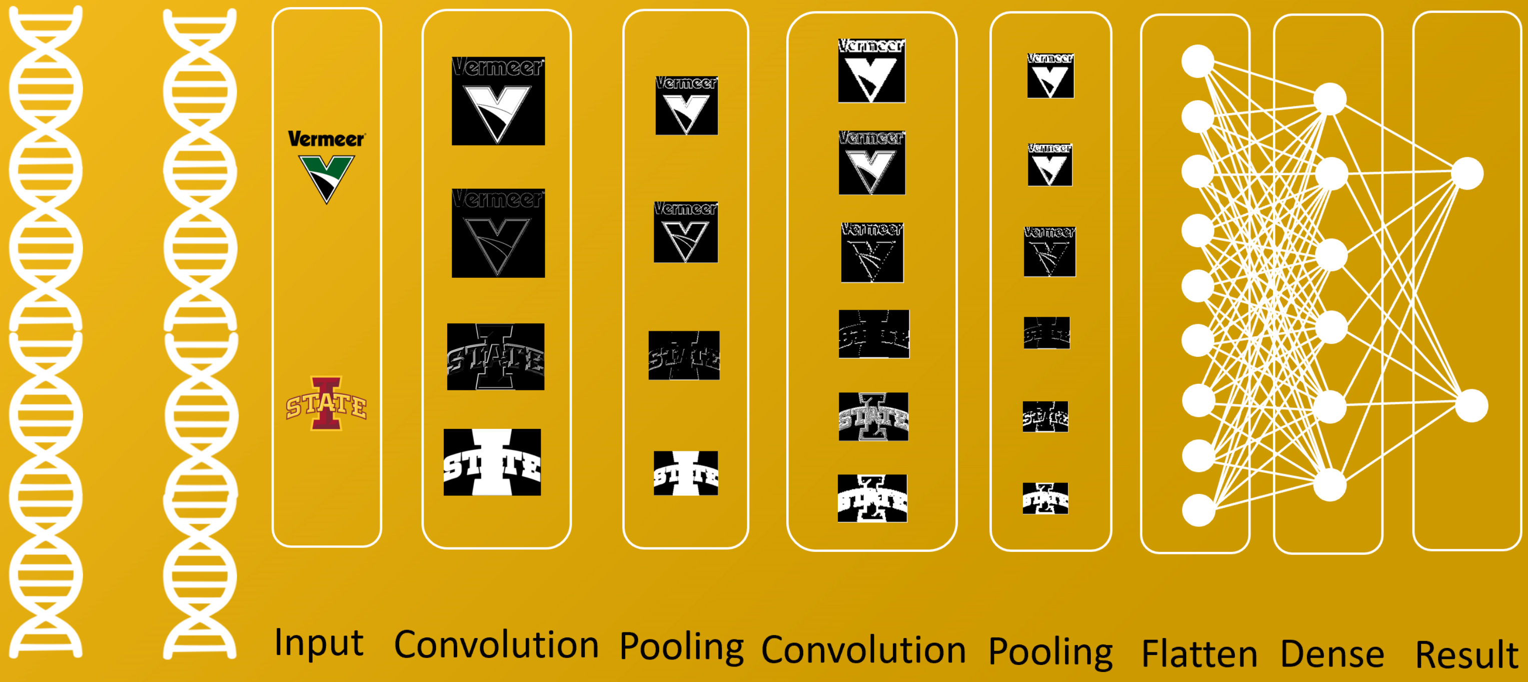
12% 80% 2% 6%

Genetic Algorithm

Initialization → Evaluation → Selection → Crossover → Mutation → Done

Genetic Learning

- Number of Filters
- Kernel Size
- Pool Size
- Number of Units

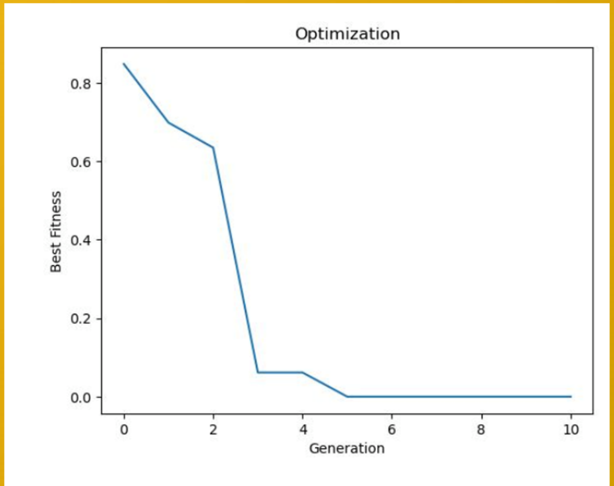


Software Design Principles and Tools

- Idea Pooling
- Component-by-component learning and development
- Python
- Tensorflow
- Keras
- Numpy

What was done?

- Single Objective Genetic Algorithm
- Multi-Objective Genetic Algorithm
- Deep Learning Model
- Single Objective Genetic Algorithm With Deep Learning
- Multi-Objective Genetic Algorithm With Deep Learning



Single Objective Function (Loss):

Single Objective Function:

$$\min \quad f(x) = x + 10sin(2x)$$

Subject to

$$0 \le x \le 10$$

Multi-Objective Function:

$$\min_{x_1, x_2} \quad \{\mu_1 = x_1^2 + 4x_2, \mu_2 = x_2^2 + 2\}$$

subject to

$$2x_1 + 3x_2^2 - 8 \le 0$$

$$x_1 + x_2 - \frac{7}{2} = 0$$

$$0 \le x_1 \le 10$$

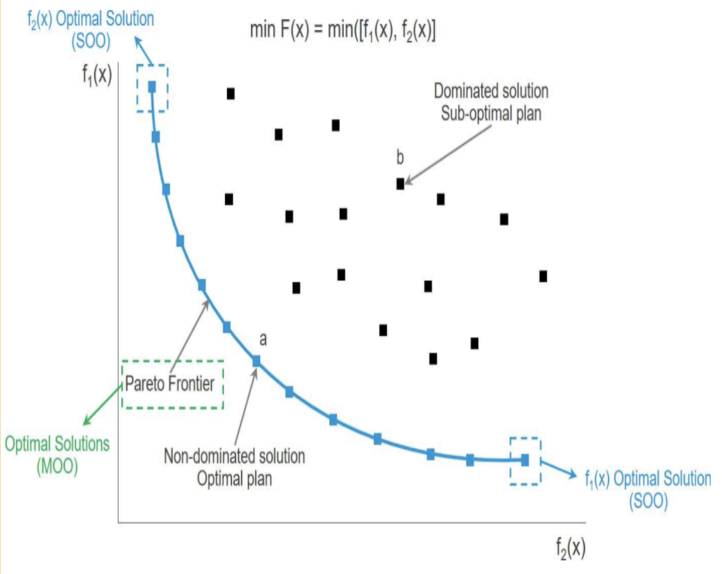
$$0 \le x_2 \le 5$$

Techniques

<u>Weighted Sum/Compromise Programming:</u>

- If n = 1 then weighted sum
- If n > 1 then compromise programming
- w_1 is the weight of function f_1
- $(1 - w_1)$ is the weight of f_2
- w_1 is between 0 and 1
- $J(x) = w_1 f_1(x)^n + (1 - w_1) f_2(x)^n$

<u>Pareto Frontier:</u>



Implementation

<u>Genetic Algorithm:</u>

generateRandomPopulation()

fitness(population)

select(population, fitnessScore)

crossover(population)

mutation(population, index)

main()

<u>Deep Learning Model:</u>

Conv2D(numberOfFilters, kernelSize,

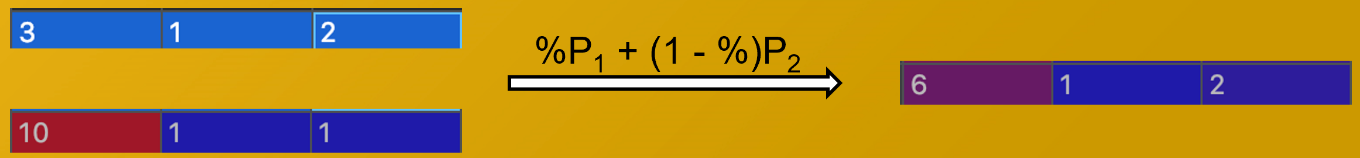
activation=”relu”)

MaxPooling2D(2)

Flatten()

Dense(2, activation=”sigmoid”)

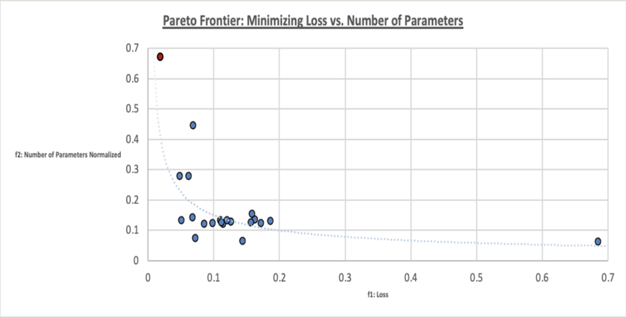
fit(Vermeer Dataset)



Results from Implementation

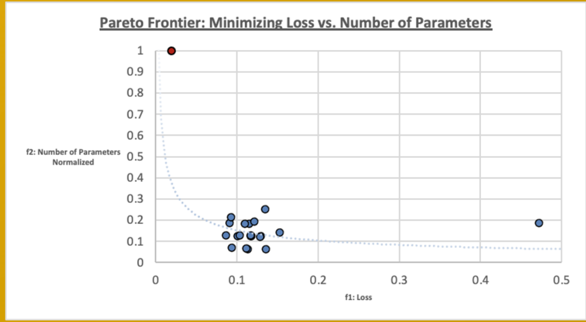
<u>Global Optimal Point (Red Dot):</u>

- F1: 0.01875
- F2: 0.671032
- Number of filters: 11
- Kernel Size: 4 x 4



<u>Global Optimal Point (Red Dot):</u>

- F1: 0.020141
- F2: 0.998088
- Number of filters: 12
- Kernel Size: 4 x 4



Limitations

- Without proper hardware, our model takes a long time to find the optimal deep learning hyperparameters
    - This forces us to use suboptimal values for population, number of generations, and number of epochs
- Variables with large domains can occasionally cause premature convergence
    - The algorithm can converge to some suboptimal value before the best value is found
- Our final results won’t be accurate if someone uses malicious images to attack our system

Future Works

- Improve the code to deal with premature convergence
  - Increase population size (requires better GPUs)
  - Implement uniform crossover
  - Favored replacement of similar individuals (crowding)
- Experiment with GA hyperparameters and functions
  - Find better population sizes, mutation rates, etc.
  - Implement different selection, crossover, and mutation functions
- Implement termination function
  - Our current GA algorithm runs the same number of generations each time
  - Termination condition can save time if the best value is found before the max number of generations is reached

Q & A