EECS 592: A Gene Expression Programming Approach to Designing CNN Architectures

Deepak Nagalla, Ishan Kapnadak, Naman Bhargava
{ndeepak, kapnadak, namanb}@umich.edu
University of Michigan, Ann Arbor

Introduction

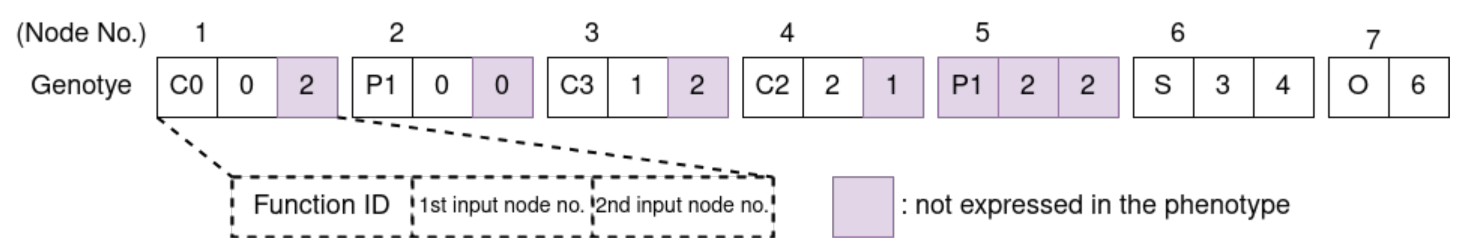
- CNNs have risen in popularity for a variety of learning tasks, but finding the best CNN architecture remains a huge challenge
- Instead of designing architectures by hand, can we automate the process? Use some kind of intelligent agent to pick the best CNN?
- Use of **evolutionary algorithms** is growing in popularity today! One common approach is Cartesian Genetic Programming (CGP)
- But CGP is expensive! We will also try to use another evolutionary approach, called **Gene Expression Programming (GEP)**
- We implemented CGP to act as a baseline and GEP as our proposed algorithm – the results seemed promising!

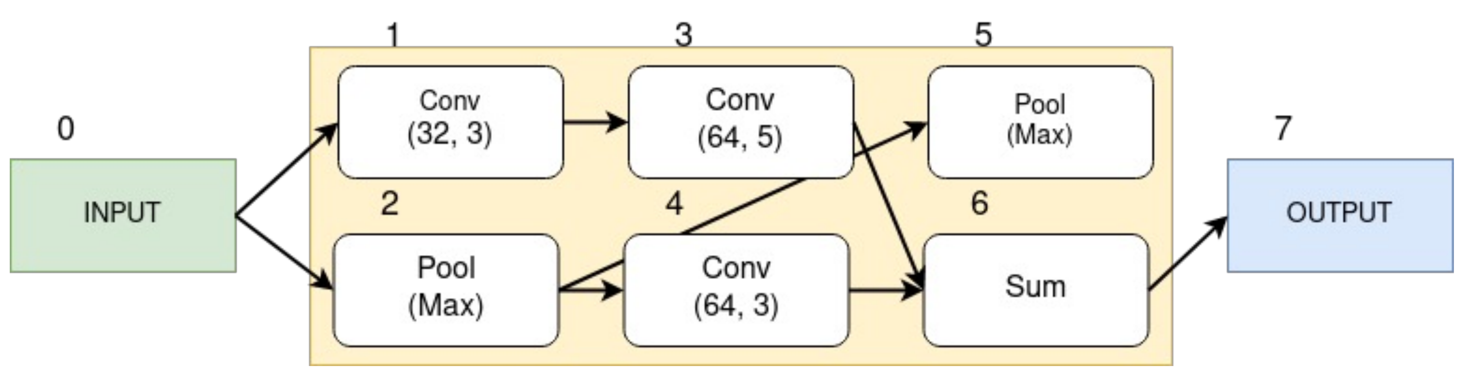
Problem Setup

- Image classification with the CIFAR-10 dataset
- Find the optimal CNN architecture for this dataset
- 32 x 32 pixel color images, categorized into 10 categories –
- 50,000 training images and 10,000 test images (equally balanced)
- Split into 45,000 training images and 5,000 validation
- *Pre-processing*: pixel values normalized to [0,1], image converted to grayscale, histogram equalization for enhancing contrast

Cartesian Genetic Programming

- Represents CNN architectures by 2-dimensional grids ($N_r \times N_c$)
- Uses highly functional blocks such as ConvBlock, ResBlock, Pooling, Summation, Concatenation.
- CNNs represented via genes that contain information about nodes of the grid – function that the node implements, nodes it is connected to, etc. – evolution via $(1+\lambda)$ algorithm, on the right

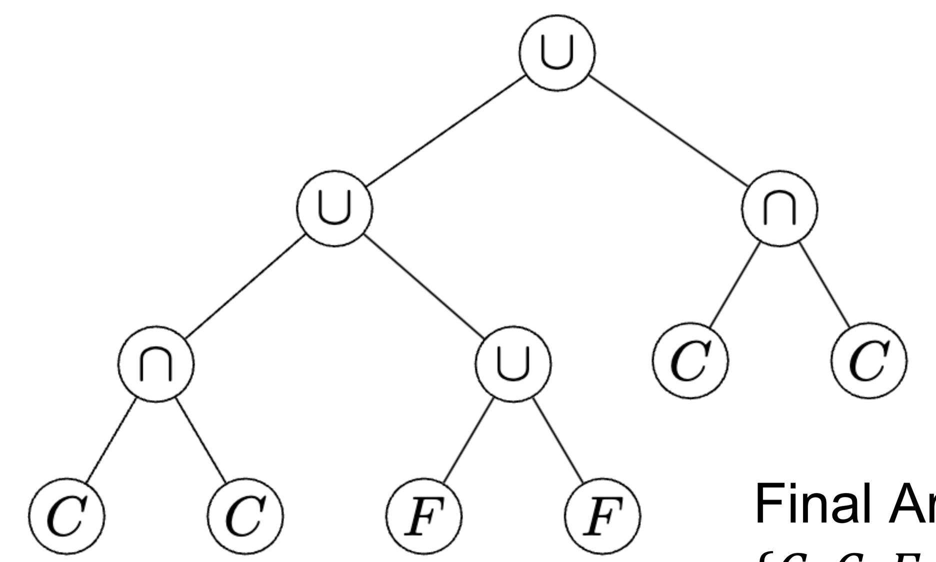




Gene Expression Programming

- Represent CNN architectures via "genes" – head & tail
- Head: {∩,∪}, Tail: $\{C, F\}$ (convolutional and fully connected layers)
- Convert gene to expression tree – then evaluate expression tree
- Order all convolutional layers first and then dense layers
- Add Max Pooling after each convolutional layer
- Experiment with and without Dropout

	0	1	2	3	4	5	6	7	8	9	10
G_1	∪	∪	∩	∩	∪	C	C	F	F	C	C
	Head					Tail					

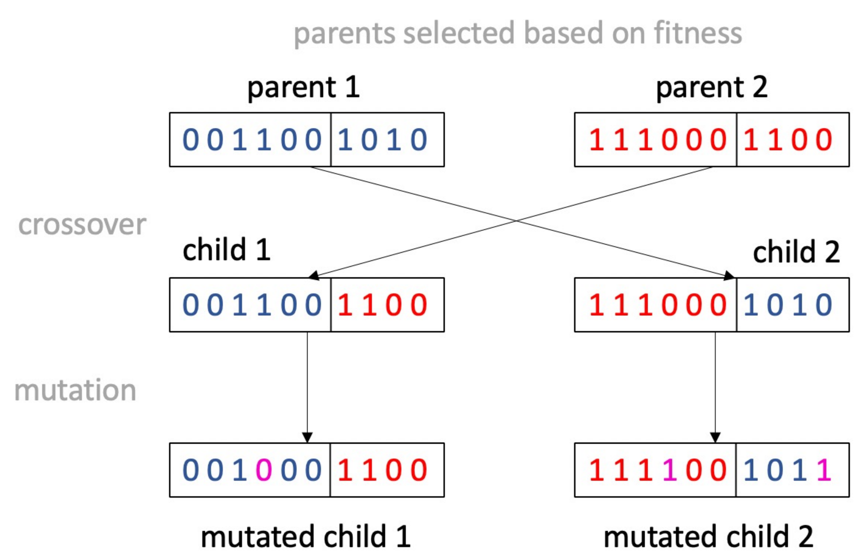


Final Architecture: $\{C, C, F, F\}$

Genetic Algorithms

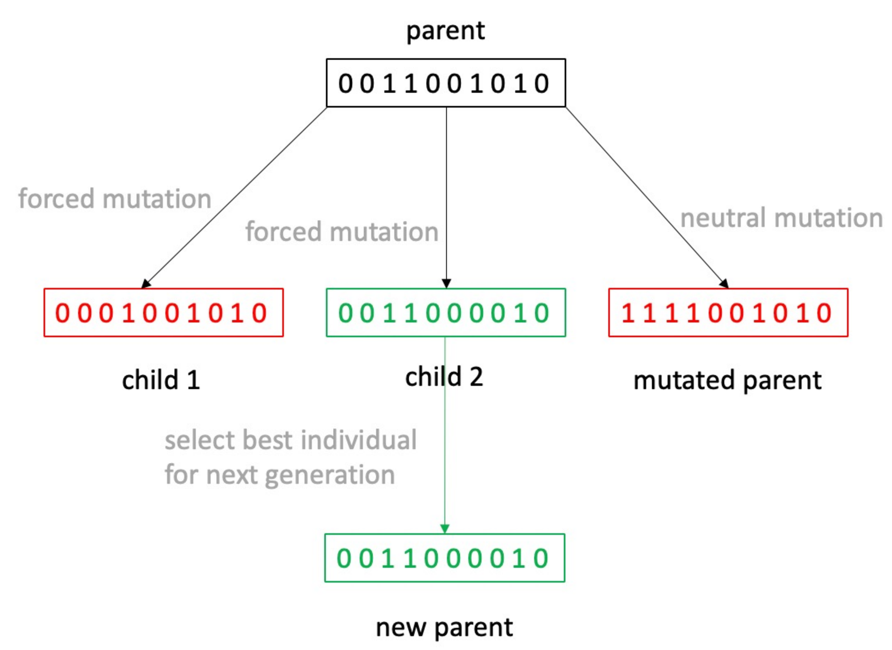
Crossover Mutation

- Population size of 10, pick parents randomly in accordance with fitness (validation accuracy of CNN)
- Generate children via crossover and mutation (mutation rate = 0.1)
- Keep evolving for some fixed number of generations (max = 50)
- At the end of the algorithm, return optimal gene (CNN) and retrain!



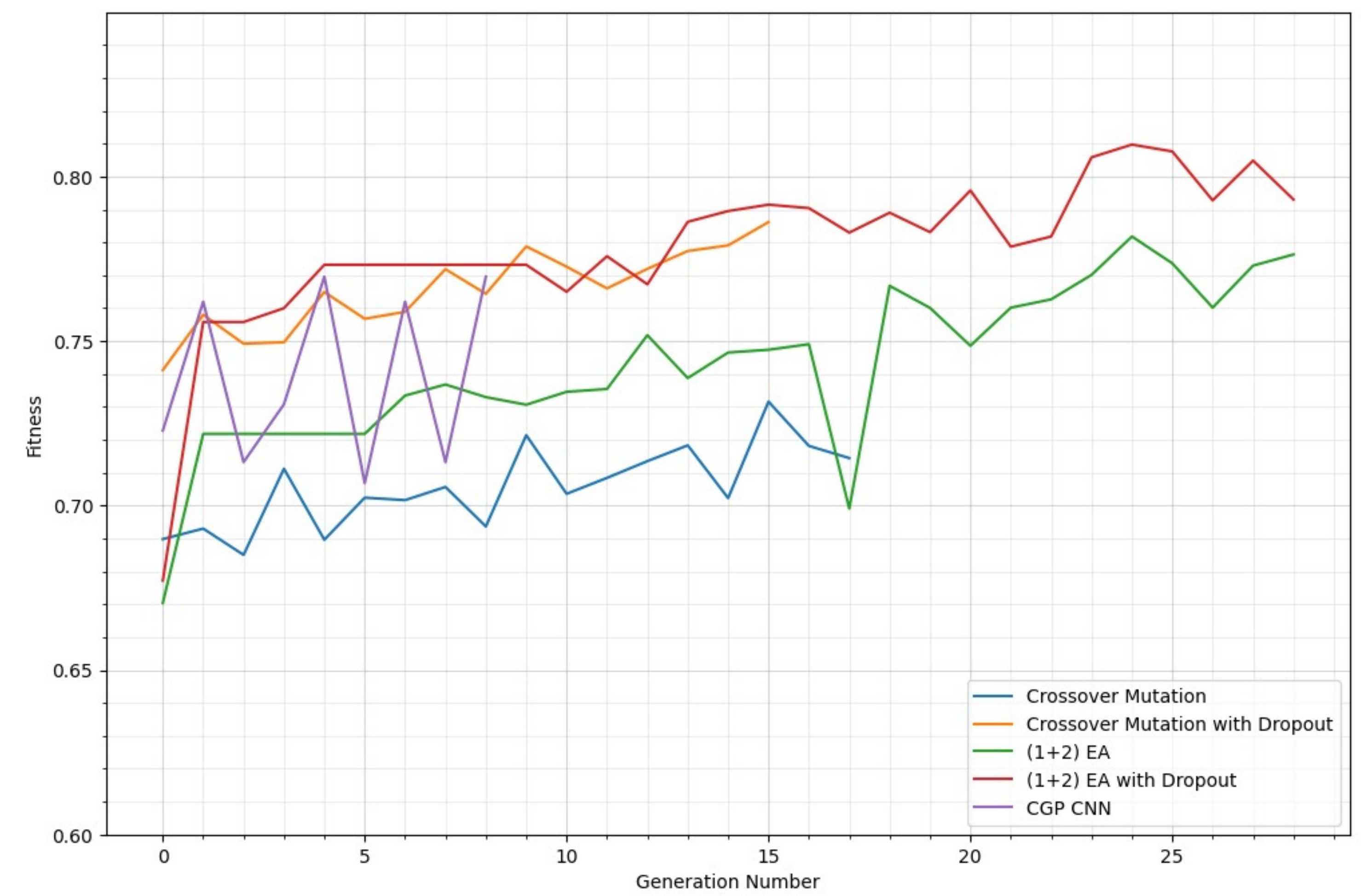
$(1+\lambda)$ Evolutionary Algorithm

- Maintain just one parent and generate λ children via forced mutation (forces mutation in one bit)
- Also mutate the parent by randomly flipping all bits (neutral mutation)
- Out of these $1+\lambda$ individuals, pick the best one as the next parent
- We use $\lambda = 2$,mutation rate = 0.1, and max no. of generations = 50



Results

Method	Accuracy
GEP with Crossover Mutation	73.16%
GEP with Crossover Mutation and Dropout	78.62%
GEP with $(1+2)$ EA	78.18%
GEP with $(1+2)$ EA and Dropout	80.97%
CGP-CNN	75.22%



Conclusion and Future work

- GEP with (1+2) EA and Dropout performed the best
- Dropout starkly increased performance in all cases
- All results suffered due to lack of computational resources
- Several improvements possible – train for more epochs, evolve for more generations, experiment with population size and value of λ
- Can also include more high-level blocks in GEP
- Overall, GEP is a promising avenue of efficient evolutionary algorithms for CNN architecture design

Significant References

- M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, (New York, NY, USA).
- D. Song, X. Yuan, Q. Li, J. Zhang, M. Sun, X. Fu, and L. Yang, "Intrusion detection model using gene expression programming to optimize parameters of convolutional neural network for energy internet," *Applied Soft Computing*, vol. 134, p. 109960, 2023.