A COMPARATIVE STUDY OF OPTIMIZATION ALGORITHMS IN PYTHON FOR NEURAL ARCHITECTURE SEARCH

DEPARTMENT OF APPLIED INFORMATICS, UNIVERSITY OF MACEDONIA

MSC IN APPLIED INFORMATICS, COMPUTER SCIENCE & TECHNOLOGY

MSC THESIS OF LIANTZAKIS KONSTANTINOS

PRESENTATION OUTLINE

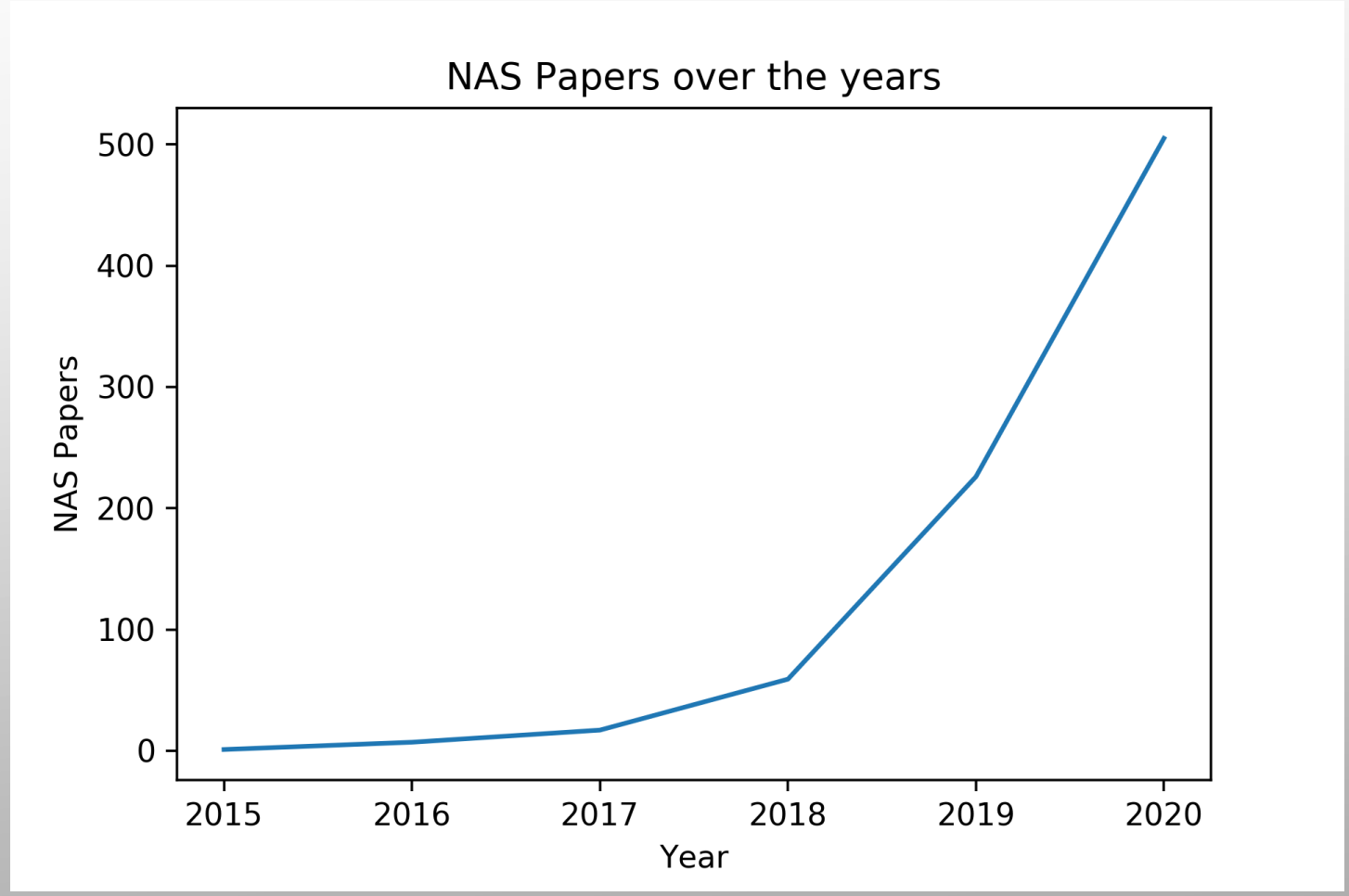
- Neural Architecture Search
- Implementation approach and tools
- Deep Q-Learning
- Evolutionary Algorithm
- Metaheuristic Algorithms using PyGMO
- Conclusion
  - Additional Work

NEURAL ARCHITECTURE SEARCH (NAS)

NAS INTUITION

- Process of automating the construction of good neural architectures
- Search spaces
  - Global (or macro)
  - Cell (or micro)
- Optimization methods
  - Reinforcement Learning
  - Metaheuristic Algorithms
- Candidate evaluation methods

NAS PROGRESSION



IMPLEMENTATION APPROACH AND TOOLS

TOOLS

- Python programming language
- Keras
- NORD
- PyGMO

G. Kyriakides and K. Margaritis, "NORD: A python framework for Neural Architecture Search," *Softw. Impacts*, vol. 6, p. 100042, Nov. 2020, doi: 10.1016/j.simpa.2020.100042.

F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: pagmo," *J. Open Source Softw.*, vol. 5, no. 53, p. 2338, Sep. 2020, doi: 10.21105/joss.02338.

CONVENTIONS AND LIMITATIONS

- Using benchmark NASBench-101 and a NASNet-like search space
  - Up to 5 hidden layers per network
  - Up to 9 connections between the layers
  - Using only 1x1 convolution, 3x3 convolution and 3x3 max-pooling layers
  - Best accuracy in the benchmark: 95.15%

C. Ying, A. Klein, E. Real, E. Christiansen, K. Murphy, and F. Hutter, "NAS-BENCH-101: Towards reproducible neural architecture search," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 12334–12348, 2019.

R. Miikkulainen *et al.*, "Evolving deep neural networks," *Artif. Intell. Age Neural Networks Brain Comput.*, pp. 293–312, 2018, doi: 10.1016/B978-0-12-815480-9.00015-3.

DEEP Q-LEARNING

DEEP Q-LEARNING INTUITION

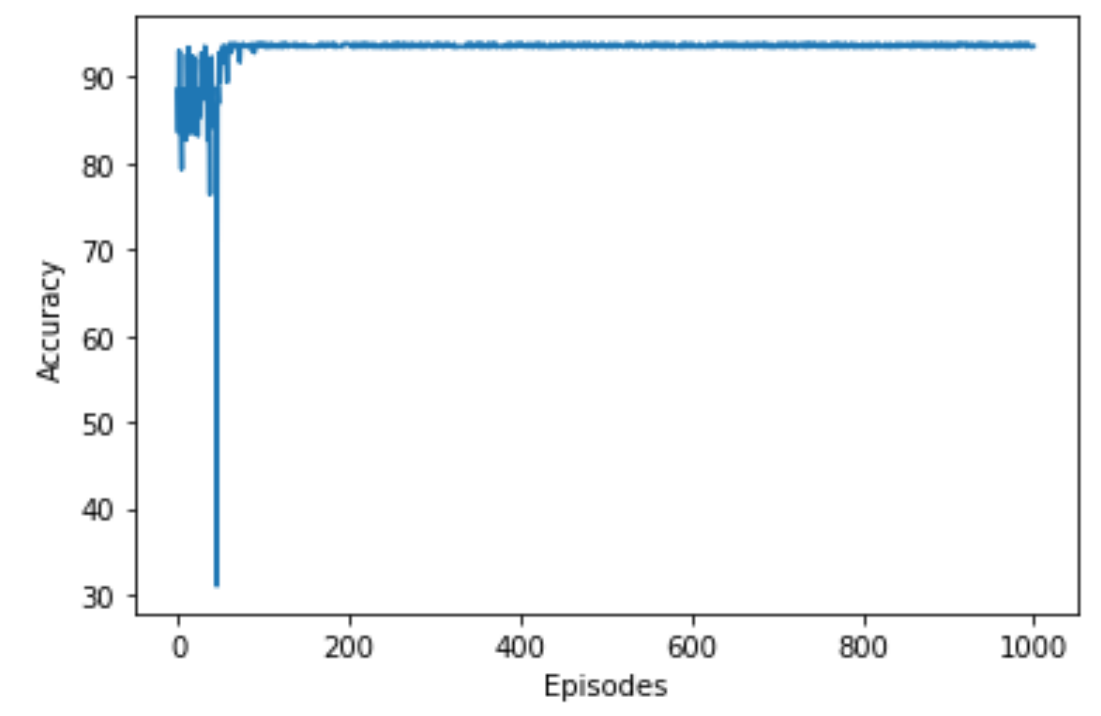
- Q-Learning returns a Q-value for a state-action pair
- Deep Q-Learning tackles the problem of larger search spaces
  - For each state input, it returns an **approximation** for each action's Q-value
- The agent that estimates the Q-values is a neural network
- Two neural networks are used: A **prediction** network (always trained) and a **target** network (updated every C iterations)
- The loss to train the prediction network is based on the different between the predicted and the target Q values
- Mechanisms to improve performance:
  - ε-greedy strategy (exploration-exploitation tradeoff)
  - Experience Replay Memory *(s, a, r, s')*

IMPLEMENTATION SETUP AND COMPARISON

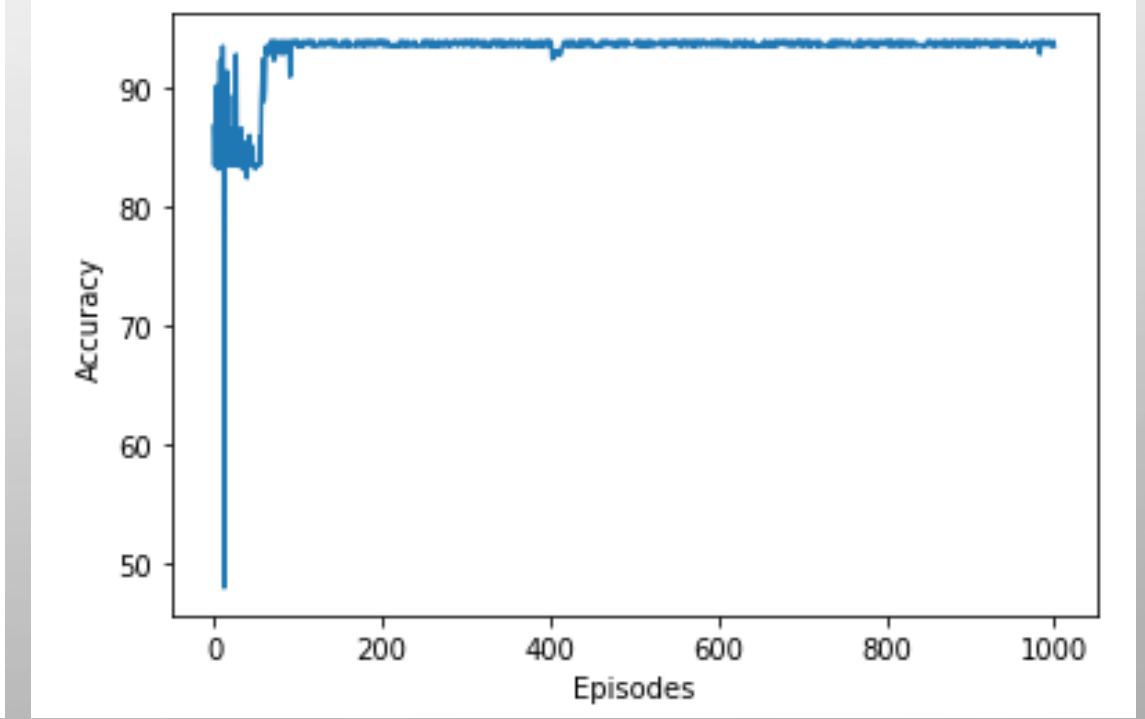
	1	2	3
Controller type	Dense	LSTM	LSTM
Episodes	1000	1000	2000
Search space	363	363	182947 (503x greater)
Best accuracy found	93.9%	93.9%	94.72%
Converges at best accuracy found	✔	✔	✘
Noticeable difference between 1-network and 2-network executions	✘	✘	✔

EXPERIMENTAL RESULTS WITH A SMALL SEARCH SPACE

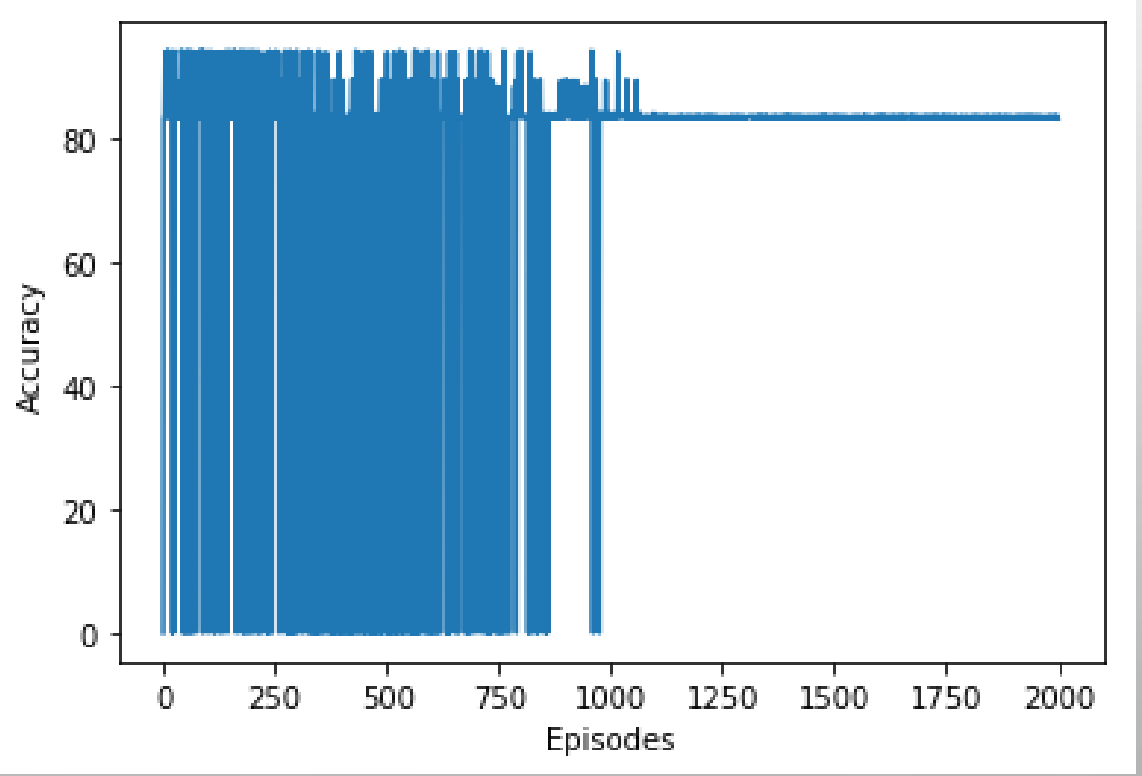
DENSE-LAYER CONTROLLER



LSTM CONTROLLER



EXPERIMENTAL RESULTS WITH A WIDER SEARCH SPACE



EVOLUTIONARY ALGORITHM

PARAMETERS TO DEFINE

- *population*: empty queue of size *P*
- *history*: Ø of size *C*
- *sample*: Ø of size *S*
- Mutation related:
  - *add_node_rate*
  - *add_connection_rate*

E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized Evolution for Image Classifier Architecture Search," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 4780–4789, 2019, doi: 10.1609/aaai.v33i01.33014780.

THE ALGORITHM (1/2)

- Initialize the population
  1. Create a random neural architecture
  2. Train and evaluate it
  3. Add it to the end of the *population*
  4. Add it to *history*
  5. Repeat steps 1-4 for *P* (population size) amount of times

THE ALGORITHM (2/2)

- After the population initialization, in every iteration:
  1. Get a random *sample* of size *S* from the *population*
  2. Get the highest accuracy model from the chosen *sample*, which is the parent
  3. Perform a mutation to the parent to produce a child
     - A mutation is either an addition of a layer or the connection of two existing ones
  4. Train and evaluate the child
  5. Add the child to the end of the *population*
  6. Add the child to *history*
  7. Remove the oldest member of the *population* (from the beginning of the queue)
  8. Repeat steps 1-7 for *C-P* amount of times
- Return the highest-accuracy model in *history*

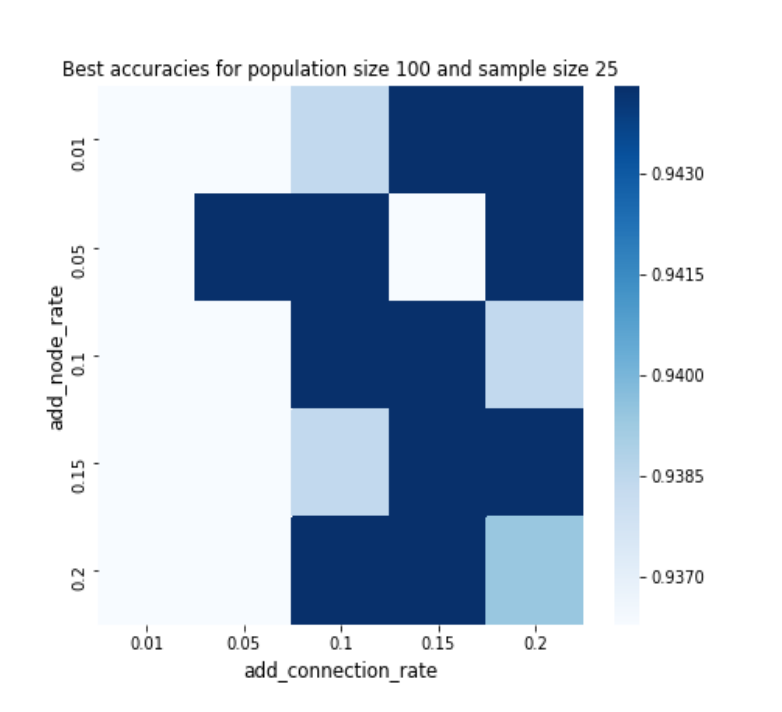
EXPERIMENTAL SETUP

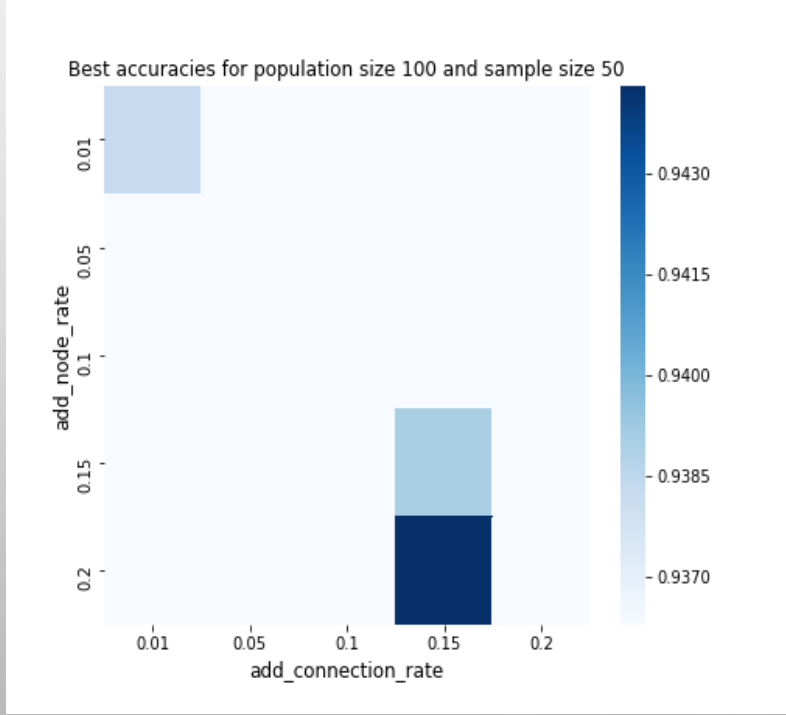
- *history* size: 1500
- *population* and *sample* size combinations *(P/S)*:
  - 100/2, 100/50, 20/20, 100/25, 64/16
- *add_node_rate*
  - 0.01, 0.05, 0.10, 0.15, 0.20
- *add_connection_rate*
  - 0.01, 0.05, 0.10, 0.15, 0.20
- *Experiments:* 10

EXPERIMENTAL RESULTS

Highest Accuracy found in all experiments: 94.43%

Worst performances have been met for the lower *add_connection_rate* values.





METAHEURISTIC ALGORITHMS USING PYGMO

PYGMO REQUIREMENTS

- An optimization problem is required as input for PyGMO
- Formulating our NAS problem results in:

max: Neural network evaluation (accuracy)

$$\textbf{subject to:} \quad \sum_{i=6}^{26} x_i \leq 9$$

$$\sum_{i=21}^{26} x_i \geq 1$$

based on the fact that all layer types and their inputs fit in a vector of 26 positions.

SELECTED ALGORITHMS

- Extended Ant Colony Optimization (GACO)
- Particle Swarm Optimization (PSO)
- Artificial Bee Colony (ABC)

EXPERIMENTAL SETUP

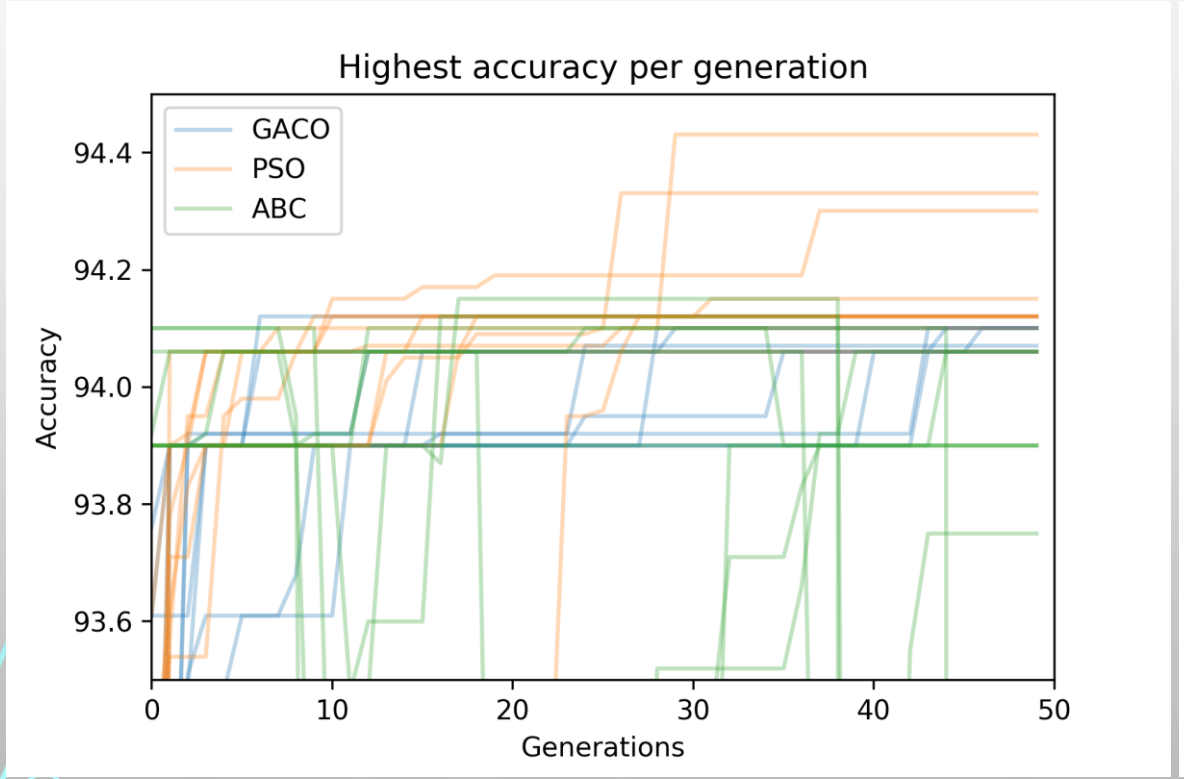
- Parameter experimentation for each algorithm
- Generations = 50
- Population size = 30
- 10 executions per algorithm (and Random Search)

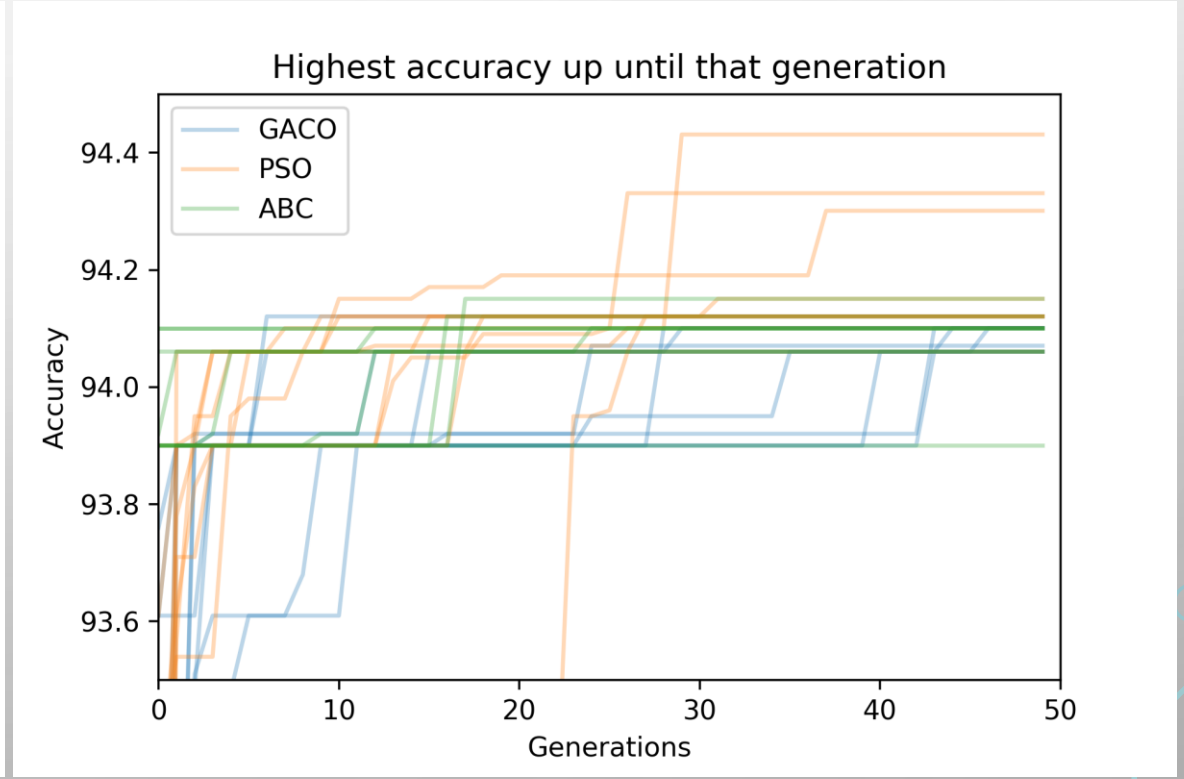
EXPERIMENTAL RESULTS

For every generation, the best candidate of the population is used for the comparison.

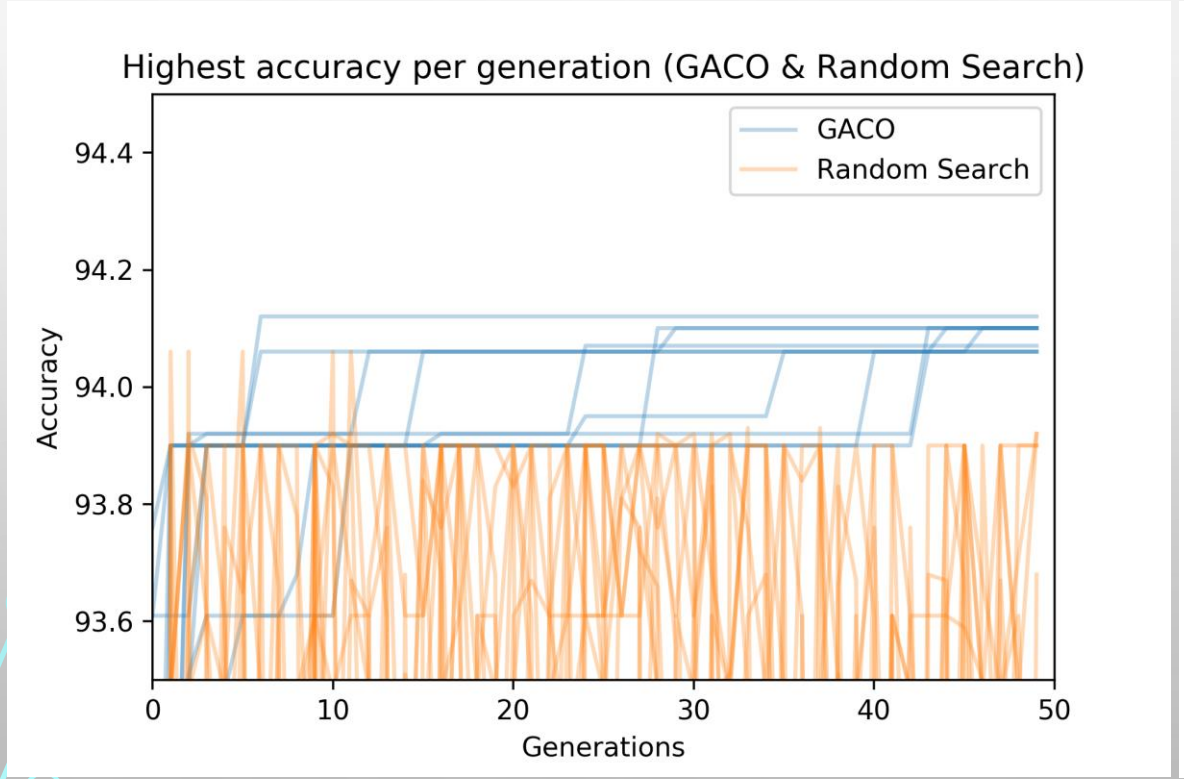
	Highest Accuracy	Highest Mean (per experiment)	Mean	Mean of Highest Accuracies (per experiment)	Average Generation in which the highest was first met
GACO	94.121%	94.017%	93.731%	94.088%	31.0
PSO	**94.431%**	**94.179%**	**94.029%**	**94.186%**	19.5
ABC	94.151%	94.018%	87.706%	93.847%	**9.0**
Random Search	94.061%	92.591%	90.109%	93.964%	18.7

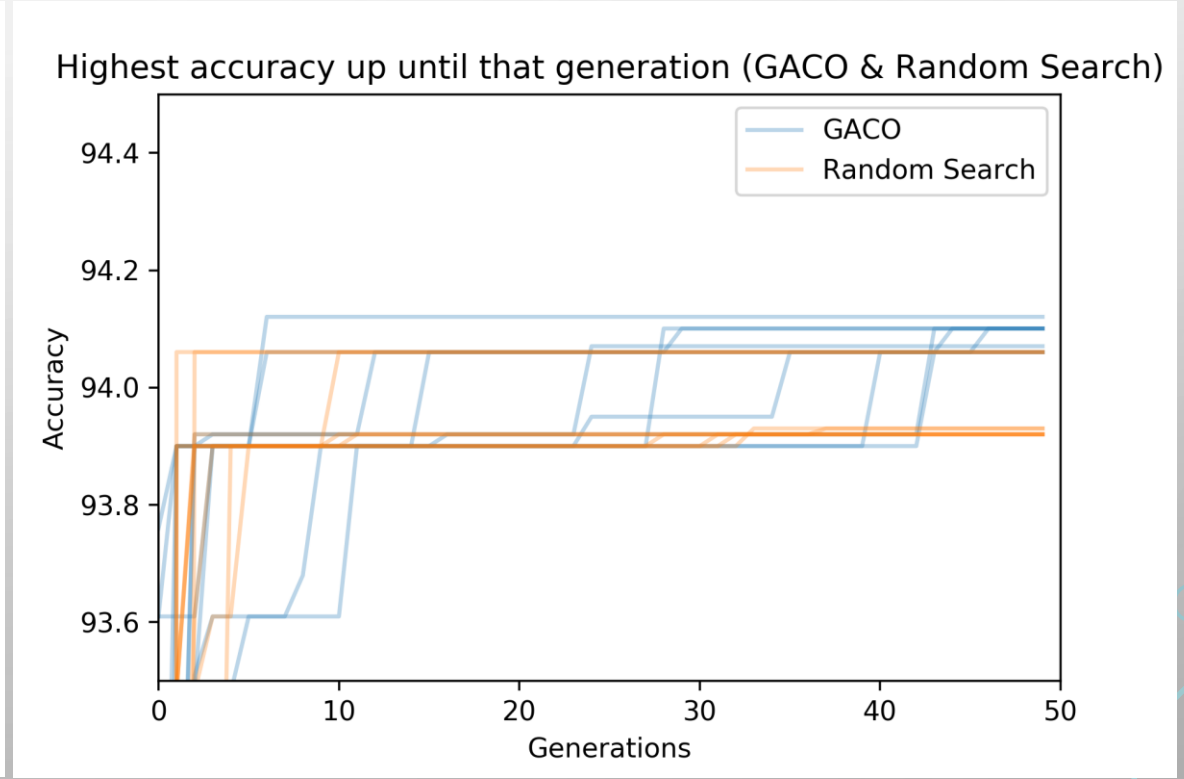
EXPERIMENTAL RESULTS (ALL ALGORITHMS)



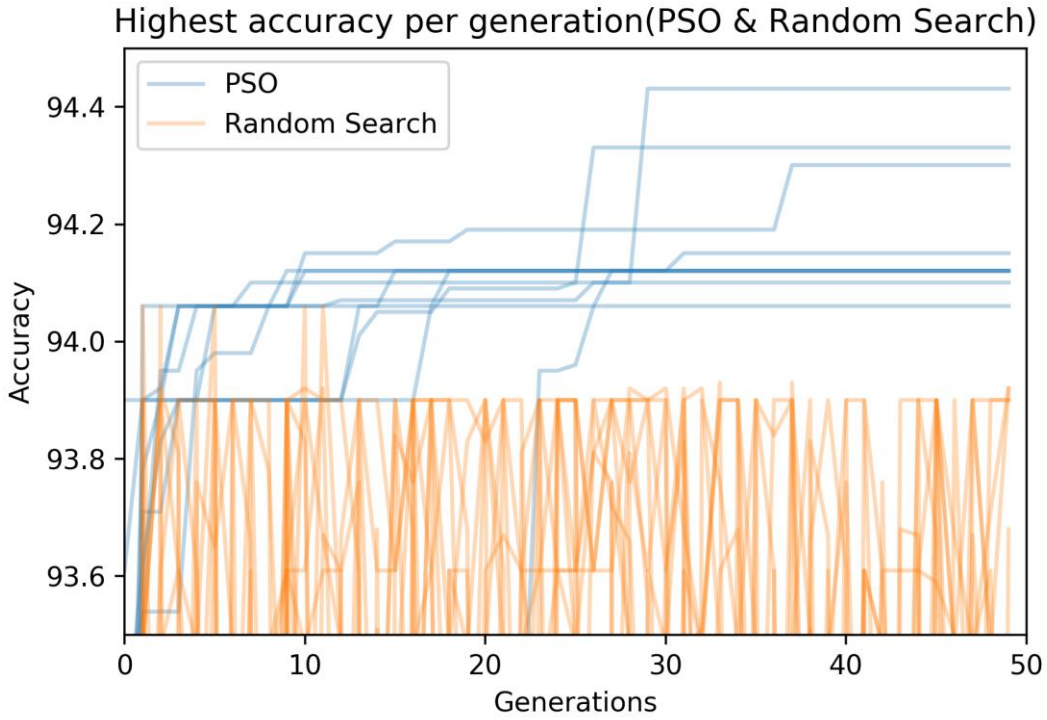


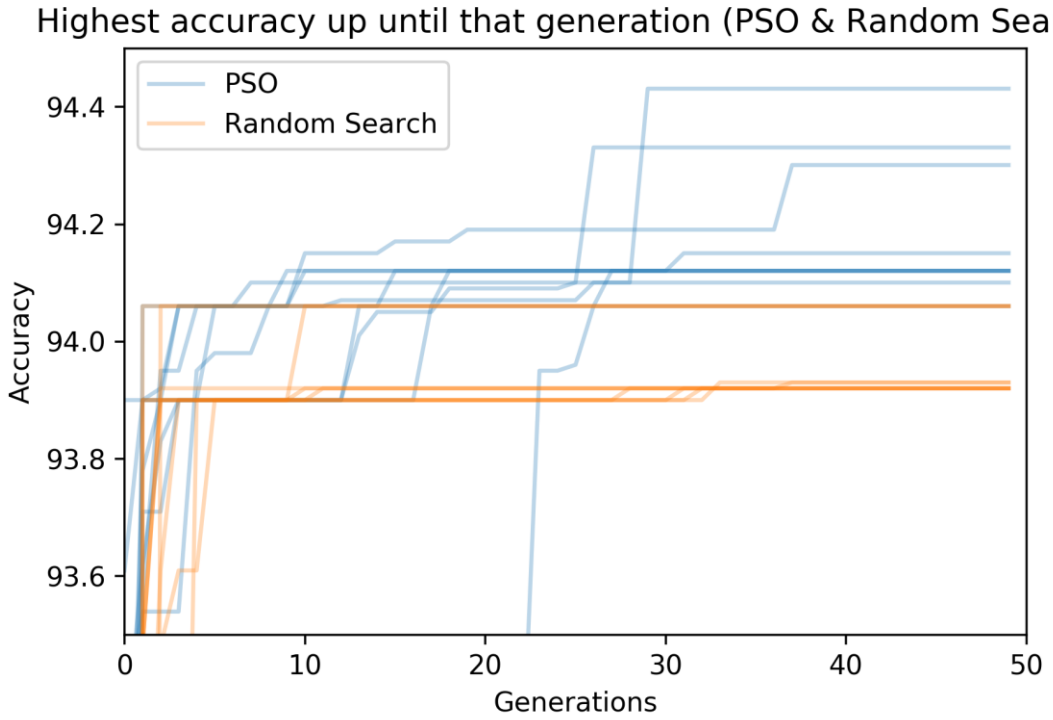
EXPERIMENTAL RESULTS (GACO)



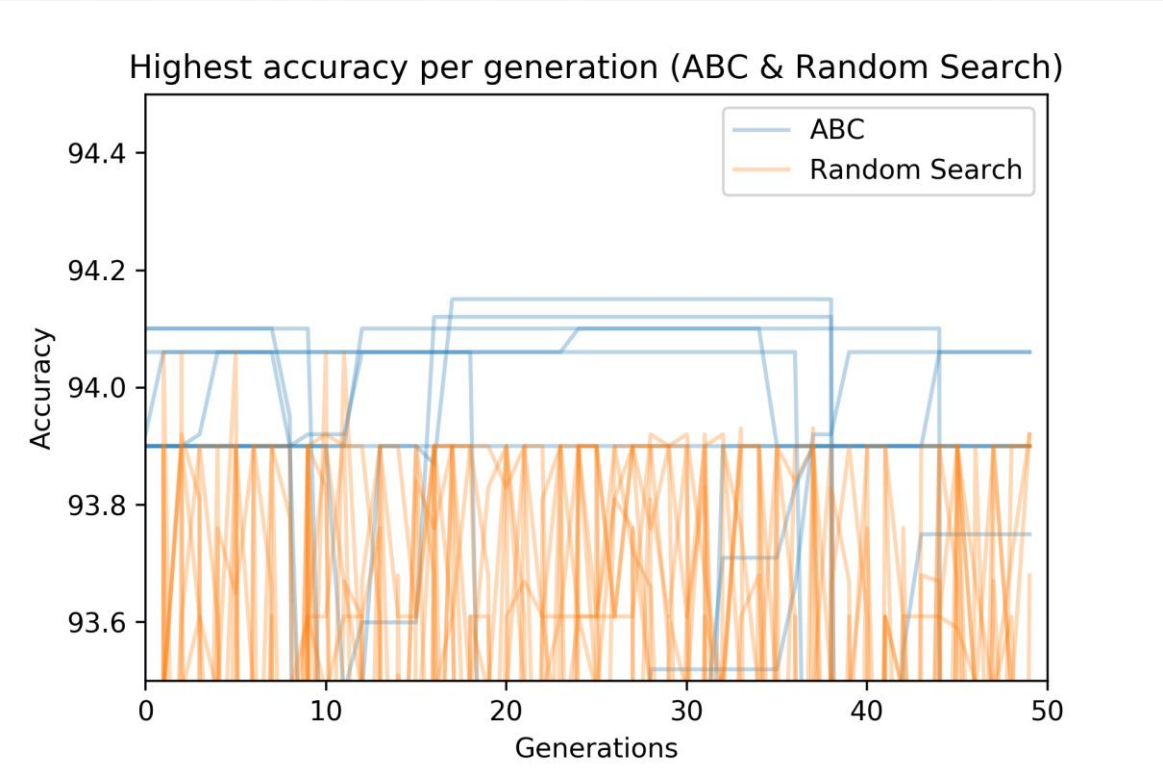


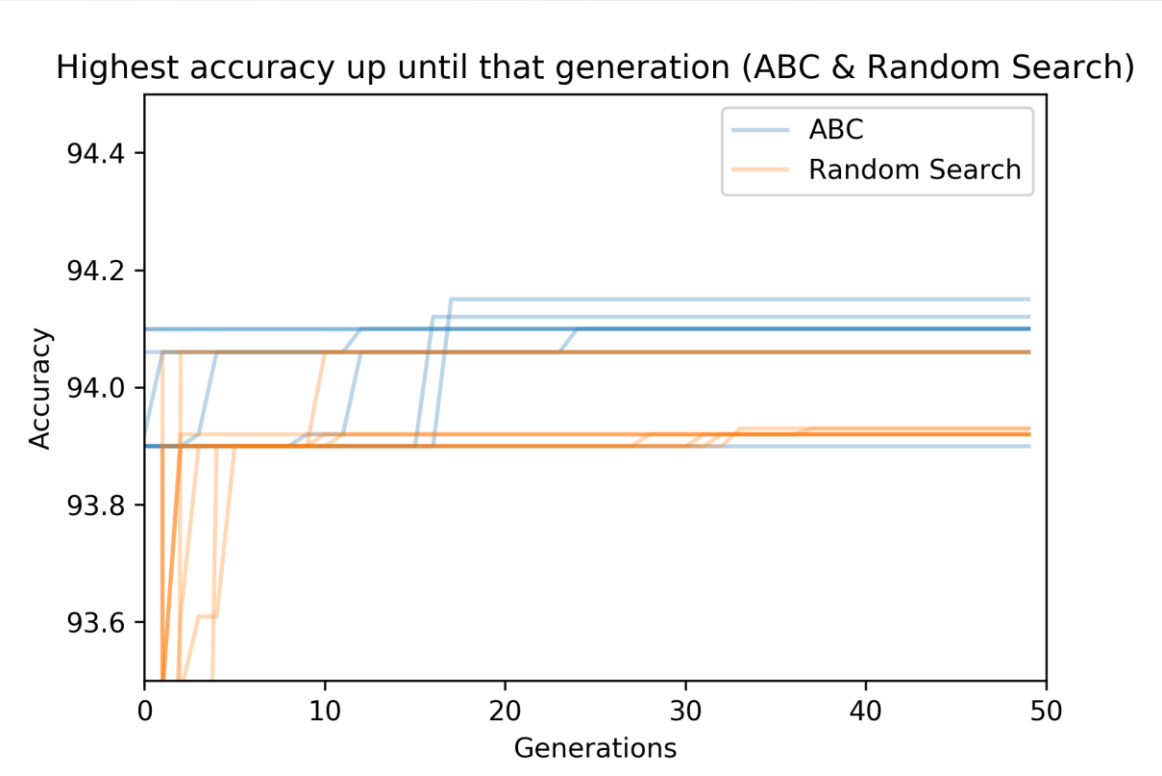
EXPERIMENTAL RESULTS (PSO)





EXPERIMENTAL RESULTS (ABC)





CONCLUSION

CONCLUSION

	DQN (small)	DQN (wide)	Evolutionary	GACO	PSO	ABC
Highest Accuracy	93.9%	94.72%	94.43%	94.12%	94.43%	94.15%

- Deep Q-Learning inadequacies for wider search spaces
  - Either add “attention” or switch to PPO
- The evolutionary algorithm and PSO outperform GACO, ABC and Random Search
- Artificial Bee Colony shows major signs of instability

ADDITIONAL WORK

- Fashion-MNIST (highest accuracy: 96.91%)
- Experiments with PyGMO (PSO)
  - Global search space (added functionality to use greater number of hidden layers)
    - 5, 7 or 9 hidden layers
    - Layer types (conv3x3, maxpool2x2)
    - Output dimension of the convolutional layers
  - Used NAS to find good architectures (only 10-20 epochs of training)
  - Trained the best architectures found for more epochs (e.g. 50, 108)
- Highest accuracy ≈ 92%
- Improvement ideas:
  - More options for the number of hidden layers and layer types
  - Environment to generate more models (current results are based on max. 200 models instead of 1500)

THANK YOU FOR YOUR ATTENTION!

- Any questions?