Exploration of Genetic Algorithms with Fast Growing Topology for Profitability in Stock Trading System

AIST 4999 TJ01 from CUHK

CHONG Tsz Wing (1155159426)

Under the Guidance of:

Prof. Benjamin W. Wah, Research Professor, Department of CSE, CUHK

ABSTRACT

This study explores an innovative approach of leveraging genetic algorithms to augment the neural network topology in a fast-growing way, named MEANT (Modularly Enhanced Augmenting Neural Topology). Extending the traditional methods, such as NEAT, which augment the neural network topology by adding single node and edge, MEANT advances by integrating entire modules. This approach aims to address the limitations of NEAT's slower, incremental growth, and the inability to match the complexity of modern, large-scale models. MEANT investigates the feasibility of autonomously determining optimal network structure, particularly in complex domains like stock trading.

1 Introduction

Traditionally, neural network design relies on human expertise and are determined by trial and errors and trained in a predefined architecture and input parameters. Some other methods, like NEAT, attempt to adjust the complexity of a neural network incrementally through techniques of augmenting topologies. However, such techniques fall short when scaling up to the complexity required for modern application. This project introduces a method that extend the idea of augmenting topology in NEAT by employing a modular approach to the network topology. It potentially accelerates the evolutionary process and enhances the adaptability of the neural network, making it more compatible for complex tasks like predictive analysis in stock trading. The motivation for this research stems from the limitations observed in the " A Deep Neural-Network Based Stock Trading System Based on Evolutionary Optimized Technical Analysis Parameters" paper, where a heuristic approach using basic genetic algorithms demonstrated significant simplification in handling stock trading conditions.

2 Methodologies

In this study, we propose a method extending the idea of augmenting topologies of neural network structure proposed in the original NEAT paper. Instead of adding nodes or edges one by one, we allow the neural network structure to add or remove a whole module in the mutation process. So, the model would be able to grow from the minimal structure, which only connects the input to the output, to a much more complex structure composed of different modules in a timely manner.

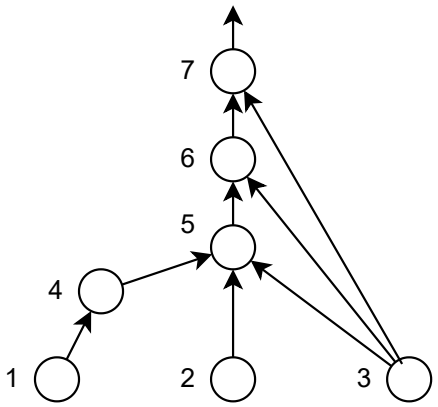


Fig 1: Example model structure from NEAT, where each node represents a perceptron

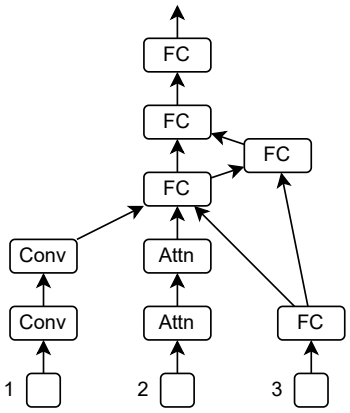


Fig 2: Example model structure from our approach, where each node represents a module

A model is treated as a acyclic directed graph of modules, where each node represents a module, and each edge directs the output of a module to the input of another module. To make the mutation process possible, which involves connecting two nodes in an alternative path with matching input and output data shape, it is necessary to specify the input shape, output shape and spatial property of each module. A genome would store the topological structure of the modules and also their weights. Then, a phenotype will be generated from a genome following the structure stored in it. A generation of a group of phenotypes generated from a gene pool will undergo natural selection. Winners survive and their offspring created from mutation would be added to the gene pool, losers die and are removed from the gene pool.

2.1 Genome

A genome in NEAT composes Node Genes and Connect Genes. Each node in Node Genes is defined only by its type (input, hidden or output) as it just acts as a perceptron in the network,

adding up all inputs and feed into a given activation function. However, this is not the case in MEANT, where a node acts as a module and a connection acts as a data channel. Knowing the shape and spatial property of data is crucial to establish a legal connection between two nodes. Therefore, each node is specified by its input shape, output shape, input spatial property, and a node index. The node index is designed to be able to determine also the node type.

NEAT Genome Properties

Node: Node index, Node type

Connection: In node, out node, weight, availability, innovation

MEANT Genome Properties

Node: Node index, module, input shape, output shape, input spatial property, weights

Connection: In node, out node, weight

2.2 Phenotype

A genotype and a phenotype are one-to-one mapped. The phenotype constructs the actual graphical structure denoted by the genome. Starting from the input nodes, each node will be activated in the topological ordering of the graph, ensuring that each node processes the inputs received from its predecessors before passing its output to subsequent nodes. This sequential activation allows the orderly propagation of data through the network. As a node activates, it computes its output based on its defined module and the sum of all its input, then send the output to all its subsequent nodes. Note that the output of each node is handled by an activation function.

Node Genes

index:	in_0	**index**:	0	**index**:	1	**index**:	2	**index**:	3	**index**:	out_0
module:	None	**module**:	Flatten	**module**:	Linear	**module**:	Conv1d	**module**:	Attn	**module**:	Softmax
in shape:	None	**in shape**:	16, 10	**in shape**:	160	**in shape**:	16, 10	**in shape**:	16, 10	**in shape**:	2
out shape:	16, 10	**out shape**:	160	**out shape**:	2	**out shape**:	16, 10	**out shape**:	16, 10	**out shape**:	None
out spatial:	True	**out spatial**:	False	**out spatial**:	True	**out spatial**:	True	**out spatial**:	True	**out spatial**:	False

Edge Genes

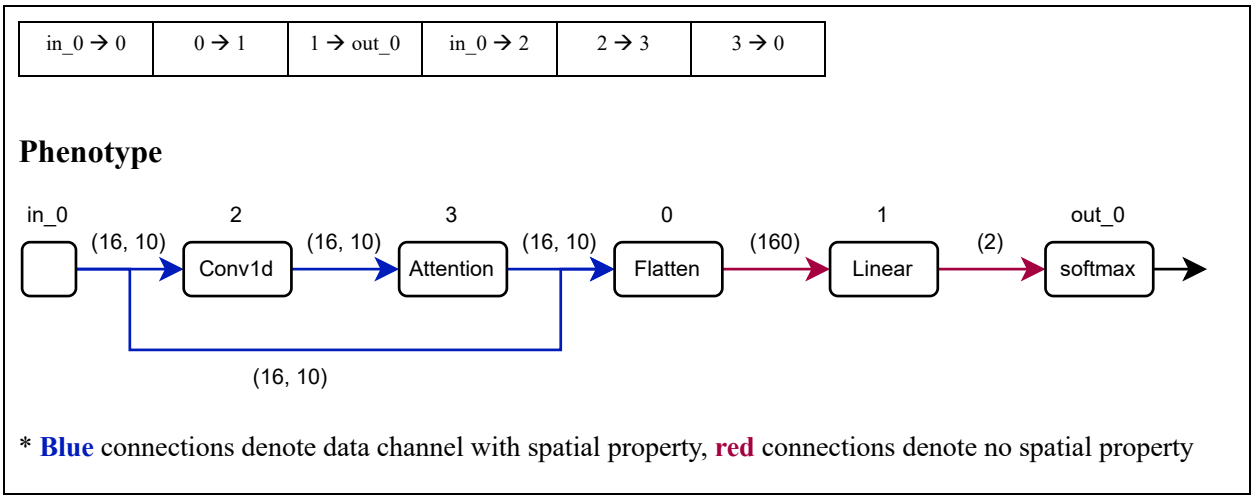


Figure 3: Genotype to phenotype mapping example

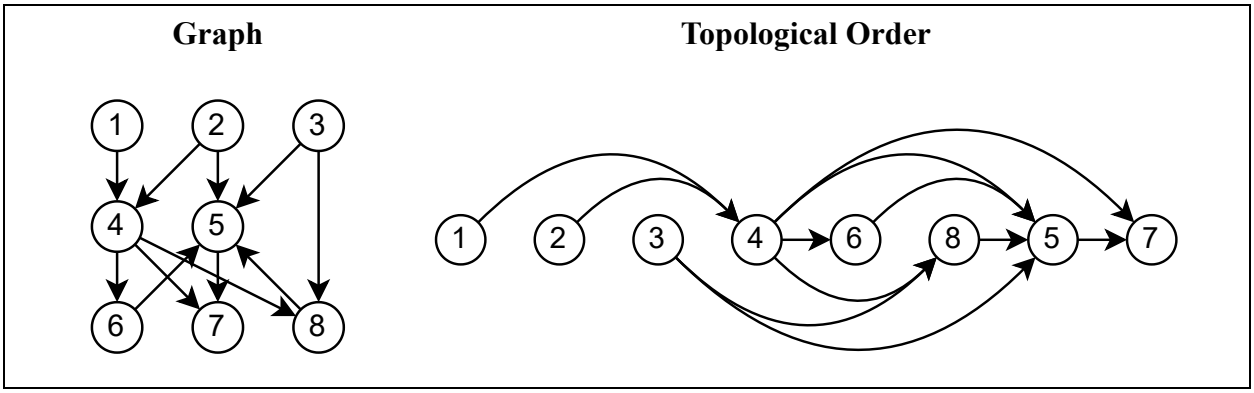


Figure 4: Topological order of nodes in a directed acyclic graph

2.3 Mutation

A mutation in genotype would lead to a structural change in the corresponding phenotype. The change in number of nodes will alter the number of parameters in the model, i.e. the dimensionality of the objective space, while the change in node connection would alter the values in the objective space. To ensure a gradual change in the objective space, the mutation must be smooth and gradually adapted by the phenotype. So, new nodes or new edges will be weighted a small value first, and as a result the mutated phenotype would have a similar objective space with its ancestor at the beginning.

In the mutation process, a replica of the original genome is first created, with itself marked as the ancestor. As only the structural data of the network is stored in the genome but not the weights, the

replica can reference the same module objects in the original genome. Multiple genome can reference the same module for its structure, which greatly reduce the storage size for the genomes.

On the other hand, it is of vital importance that the mutation rules should allow all possible legal (acyclic) topologies to emerge. Also, data channel should have the ability to increase its size to fit a larger module, allowing a structure with dimensionality higher than the maximum of input and output to emerge. A node replication mutation is added to fasten the mutation process.

2.3.1 Adding One Node

This mutation method would add a new node connecting two existing nodes as its initial in node and out node. The “in shape”, “out shape” and “out spatial” property of the new node is defined by the in node’s “out shape”, the out node’s “in shape” and the in node’s “out spatial” property. Then, based on this “in shape”, “out shape” and “out spatial” configuration, a node factory would generate a list of candidate modules which all can take data of “in shape” to output data of “out shape”. The resulting node being added is randomly chosen among this list, configured to correctly adapt the data shapes.

The node pair chosen to insert the new node is not arbitrary. It has to be ensured that adding the new node would not create a cycle in the phenotype structure. Also, it is not allowed for the case that the in node does not have a spatial property while the out node does, as the new connection would destroy the spatial property of the out node. To achieve this, we can filter the set of all possible edges with a reachability map, indicating what other nodes can a node reaches in the graph. Then, check the remaining to align with the spatial property inheritance criteria.

To ensure the mutation leads to a gradual change of the phenotype’s objective space, the outgoing edge of the new node is initially weighted with a small number.

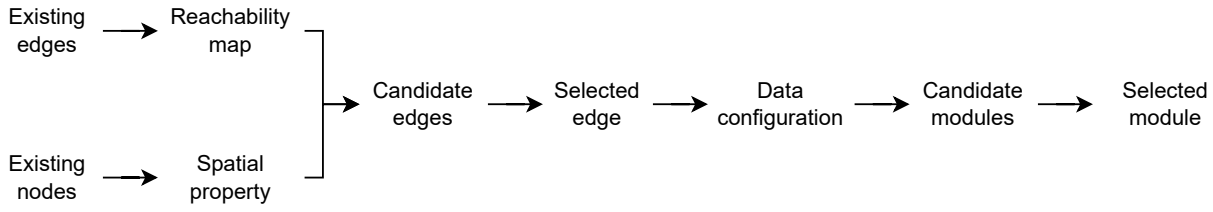


Fig 4: Overview of the process of adding one node

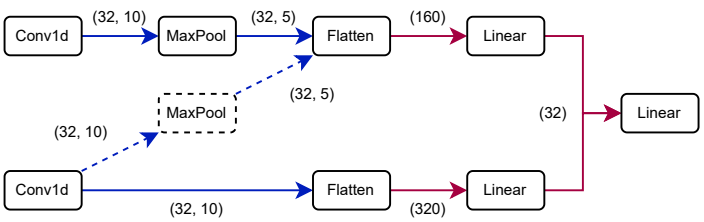


Fig 5: Illustration of the change in phenotype undergoing the mutation of adding one node

2.3.2 Adding Two Nodes

The mutation of adding one node doesn't allow the augmentation of data dimensionality. For example, a simple sequential model of a single hidden layer neural network, with the hidden layer's size larger than the input and output, cannot never emerge via the mutation of adding one node. This is because the data shape is predefined and thus capped by the existing data shapes in the genome. Therefore, it is necessary to allow a large data shape to emerge, which can be achieved by adding two nodes simultaneously.

The process is roughly the same with adding one node. A node pair is selected, and the two new node would be connected sequentially between them. The "in shape" of the prior node and the "out shape" of the later node is predefined by the node pair. However, the data shape between the prior and later nodes remains flexible. Based on the "in shape", "out shape" and "spatial" properties, a shape factory can generate a list of candidate "mid shapes" that is compatible with them. Then, a random "mid shape" is selected, and used to undergo two times of adding one node, prior and later.

To ensure the mutation leads to a gradual change of the phenotype's objective space, the outgoing edge of the new later node is initially weighted with a small number.

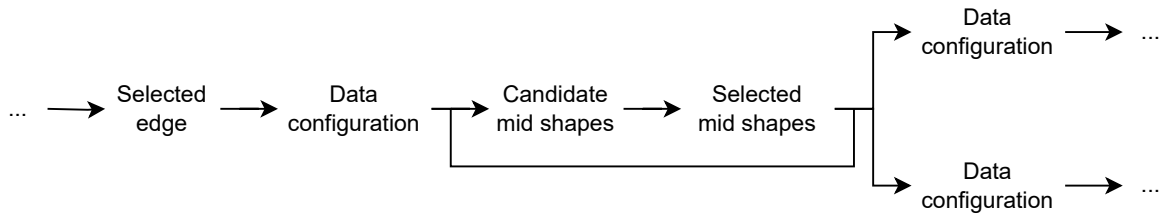


Fig 6: Overview of the process of adding two nodes

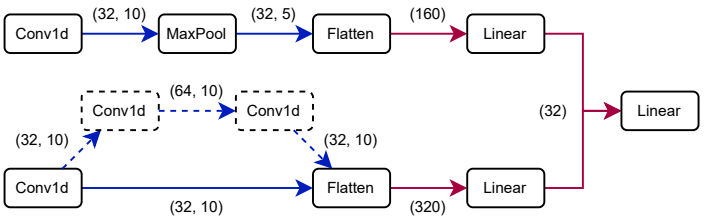


Fig 7: Illustration of the change in phenotype undergoing the mutation of adding two nodes

2.3.3 Adding Edge

This mutation method establishes a new connections between a node pair. The rule of choosing the node pair is stricter than that for adding nodes. As the output of the in node is directly passed to the out node, the in node's "out shape" and out node's "in shape" must exactly match. The weight of the new edge is also initially set to a small number to ensure smoothness change in objective space.

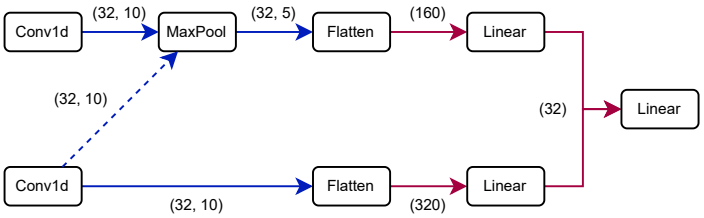


Fig 8: Illustration of the change in phenotype undergoing the mutation of adding an edge

2.3.4 Removing Edge

An edge is eligible to be removed if its weight is small enough, indicating its ineffectiveness in altering the model output. As the weight of the edge being removed is small, the change in objective space remains gradual. Removal of an edge might create dead end, where a node has no outgoing edge. This node would remain in the genome to allow future possible reconnection by mutation.

Also, this mutation method allows a complete replacement of an alternative path created with the above mutation methods between a node pair, which is equivalent to inserting a node in an edge on the graphical structure perspective.

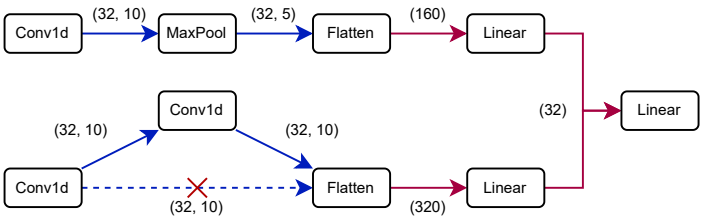


Fig 9: Illustration of the change in phenotype undergoing the mutation of removing an edge

2.3.5 Node Replication

This mutation method is not necessary to make all legal topologies possible to emerge, but it helps fasten the model growing speed by creating a twin identical to a selected node. The node being replicated can be chosen arbitrary in the genome. All connections of the selected node will be replicated to the replicated node.

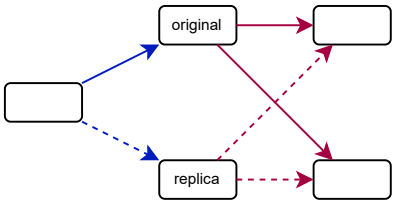


Fig 10: Illustration of the change in phenotype undergoing the mutation of replicating a node

2.4 Natural Selection

A population is a collection of phenotypes constructed from a genomes in a genome pool. Starting from the initial genome of the minimal possible topology that only maps the model input shape to the output shape, some mutations will be generated from the original genome and added to the population. Then, in each generation, the population would be trained, evaluated and selected. Only genomes with the best performance can survive and create offspring.

As the expected performance of the surviving genomes must be at least as good as the expected performance of the whole population in a generation. The expected performance throughout generations is expected to increase monotonically.

2.4.1 Training

The population is trained on the training dataset. Each phenotype would undergo backpropagation and use gradient descent to fit the data. The parameters trained in phenotype is reflected in the genome, which stores also the structural information of the model. So, if a genome survives in this generation, its weights would be passed to its successors. This ensure that the mutants are changing gradually to potential changes.

2.4.2 Evaluation

In the evaluation process, the trained population would perform back testing on the validation dataset. This is crucial as overfitting or underfitting models in the training process would likely to have a worse performance in this separate unseen dataset. Thus, we can avoid the complexity of the genomes growing infinitely, as an overcomplicated genome would cause overfitting and get eliminated in the selection phase.

2.4.3 Selection

Based on the evaluation result, the genomes would be ranked accordingly. Top-ranked genomes would be selected to remain, while all other genomes would be eliminated genome pool. Remaining genomes would create random mutations and added back to the genome pool.

2.5 Trading Strategies

The genome output is set to be two values summing up to 1. The first value indicates the model's confidence of price going up the next day, and the second value indicates the price going down. If any of the values exceed a predefined threshold, a corresponding trading signal would be produced. The model will hold a long position with all its capital if it expects the price to go up, and a short position with all its capital if it expects the price to go down. Otherwise, it will clear its position and stay out of the market.

An alternative long-position-only strategy can be adopted for ablation study.

3 Data

Since MEANT is a highly adaptive algorithm, the data shape used to train it can be arbitrary. For simplicity, in this paper, we will m features of n days to create an $m \times n$ matrix as the input data

shape. The features would be augmented and calculated from the raw OHLCV (Open, High, Low, Close, Volume) data fetched from yahoo finance API. One hot encoding would be adopted for the label, where [1, 0] and [0, 1] indicate the price goes up and down respectively in the next day.

Features			
Open	Volume	Log Close	RSI 21
High	Log Open	Log Volume	EMA 9
Low	Log High	RSI 7	EMA 21
Close	Log Low	RSI 14	EMA 55

Table 1: Feature list

The following list of tickers would be used to evaluate the models:

Tickers				
AAPL	DD	INTC	MRK	TRV
AXP	DIS	JNJ	MSFT	UNH
BA	GE	JPM	NKE	VZ
CAT	GS	KO	PFE	WBA
CSCO	HD	MCD	PG	WMT
CVX	IBM	MMM	RTX	XOM

Table 2: Ticker list

Data from 01/01/2013 to 31/12/2022 will be used. The dataset is split into the training dataset, validation dataset and testing dataset in a 60%, 20% 20% ratio.

Each phenotypes would be trained with 20 epochs on the training dataset, then perform a back testing on the validation dataset.

4 Performance Evaluation

The performance of a model would be evaluated based on cross entropy loss, returns (PnL), Sharpe ratio, Sortino ratio, max drawdown, hit rate, and model complexity, compared within a generation and across different generations.

4.1 Equations

Sharpe Ratio

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}$$

where:

R_p = Return of portfolio

R_f = Risk-Free rate

σ_p = Standard deviation of portfolio's excess return

Sortino Ratio

$$\text{Sortino Ratio} = \frac{R_p - R_f}{\sigma_d}$$

where:

R_p = Return of portfolio

R_f = Risk-Free rate

σ_d = Standard deviation of negative returns (downside)

Maximum Drawdown (MDD)

$$\text{MDD} = \frac{\text{Trough value} - \text{Peak value}}{\text{Peak value}}$$

4.2 Performance over Generations on Individual Stocks

After training the population over 5 generations on the training dataset of individual stocks and applying selection with the validation dataset, we select the model that performs the best in the last validation back testing. This best model will then be used to trade on the unseen testing dataset to evaluate its final performance.

Company	Price %	Return %	Outpfm	APY	# of L	# of S	# of H	Hit rate	Sharpe	Sortino	MDD
AAPL	0.74%	56.83%	56.09%	25.57%	303	120	75	56%	1.00	0.86	20%
AXP	22.05%	27.41%	5.37%	13.04%	299	121	78	50%	0.47	0.40	19%
BA	-7.88%	0.16%	8.05%	0.08%	272	144	82	52%	0.16	0.15	24%
CAT	23.57%	29.16%	5.58%	13.82%	272	147	79	50%	0.48	0.40	20%
CSCO	5.03%	21.63%	16.61%	10.42%	275	155	68	54%	0.54	0.52	23%
CVX	95.95%	-21.66%	-117.61%	-11.62%	287	124	87	51%	-0.37	-0.30	40%
DD	-18.28%	50.31%	68.58%	22.90%	269	150	79	45%	1.65	-0.90	23%
DIS	-51.49%	-6.62%	44.87%	-3.40%	288	137	73	50%	-0.01	-0.01	32%
GE	-8.53%	-7.03%	1.50%	-3.62%	262	169	67	49%	0.04	0.04	40%
GS	16.86%	0.76%	-16.10%	0.38%	291	135	72	51%	0.17	0.16	37%
HD	16.45%	31.06%	14.61%	14.67%	301	125	72	49%	0.72	0.64	26%
IBM	14.61%	27.64%	13.03%	13.14%	258	159	81	50%	0.73	0.71	13%
INTC	-48.72%	118.45%	167.17%	48.50%	274	159	65	52%	1.46	1.67	19%
JNJ	10.84%	35.41%	24.57%	16.58%	291	132	75	47%	1.29	1.12	25%
JPM	-2.86%	23.49%	26.35%	11.27%	292	141	65	50%	0.56	0.59	30%
KO	26.71%	-10.74%	-37.46%	-5.59%	287	130	81	50%	-0.27	-0.27	31%
MCD	23.01%	-4.80%	-27.81%	-2.46%	310	131	57	52%	-0.03	-0.03	24%
MMM	-27.41%	-1.71%	25.70%	-0.87%	289	148	61	53%	0.07	0.06	31%
MRK	36.79%	15.96%	-20.84%	7.78%	295	130	73	50%	0.48	0.49	30%
MSFT	10.27%	-19.62%	-29.89%	-10.46%	298	141	59	52%	-0.35	-0.30	34%
NKE	-20.43%	2.74%	23.17%	1.38%	291	137	70	52%	0.19	0.17	36%
PFE	35.66%	10.28%	-25.38%	5.08%	277	132	89	52%	0.37	0.34	35%
PG	9.95%	-13.95%	-23.90%	-7.32%	284	136	78	47%	-0.60	-0.62	39%
RTX	44.58%	26.07%	-18.51%	12.44%	274	136	88	52%	0.63	0.58	21%
TRV	36.43%	25.95%	-10.47%	12.38%	277	133	88	47%	0.73	0.66	18%
UNH	46.97%	2.08%	-44.89%	1.05%	311	116	71	48%	0.17	0.17	27%
VZ	-31.43%	-14.54%	16.89%	-7.64%	283	137	78	50%	-0.42	-0.37	25%
WBA	-21.68%	-18.05%	3.62%	-9.58%	277	151	70	51%	-0.27	-0.23	33%
WMT	-3.73%	95.88%	99.62%	40.53%	297	118	83	54%	1.70	2.00	17%
XOM	135.48%	112.63%	-22.85%	46.48%	264	150	84	51%	1.86	1.75	11%
Total	12.32%	19.84%	7.52%	8.50%	284.93	138.13	74.93	50.58%	0.44	0.35	27.78%

Table 3: MEANT Dow 30 Result ("Outpfm" = Outperform, "APY" = Annualized, "# of L" = Number of Long, "# of S" = Number of Short, "# of H", "MDD" = Max Drawdown)

Below is the result of the best model in later generation trading on the stock AAPL:

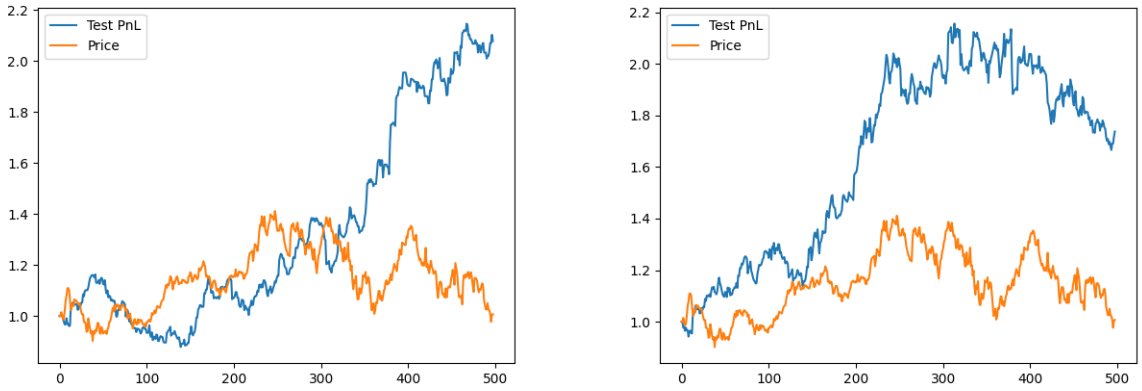
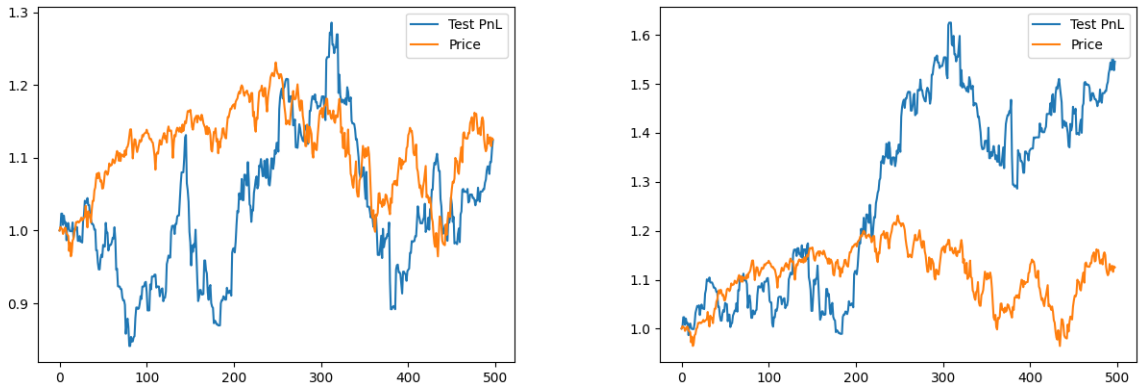


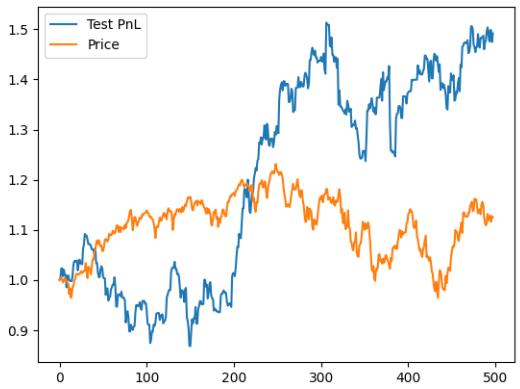
Fig 11: Sample of best performing model trading on stock AAPL

4.3 Performance over Generations on Portfolio

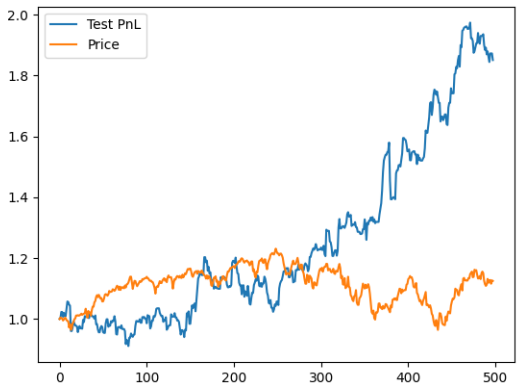
The performance of the best model using this evolutionary algorithm increases dramatically when trained with data from all stocks available. The best model would trade on a portfolio constituted from the 30 stocks with equal weights. Then, fluctuation and noises in a single stock would be hedged out in the portfolio, leading to a more stable performance with lower volatility.



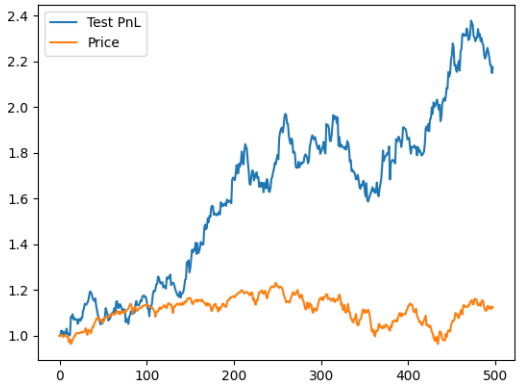
Gen 0 **Gen 1**



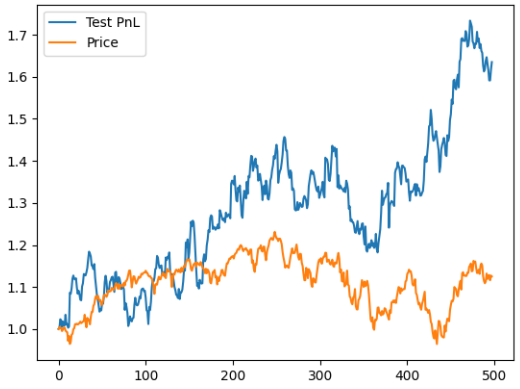
Gen 2



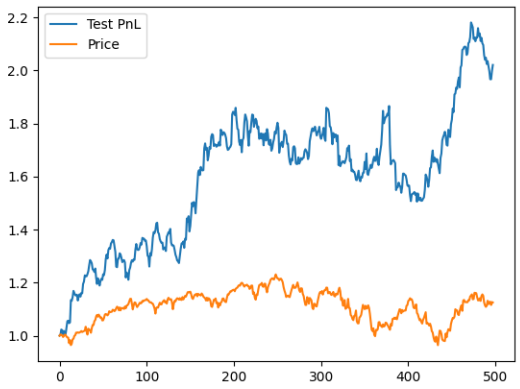
Gen 3



Gen 4



Gen 5



Gen 6

Gen 7

5 Future Works

Current studies focus most on the performance of the models. Yet, it is also valuable to track the model complexity trend and trace the genealogy. We would also compare the performance of this approach with the original method mentioned in the papers that we base on. Ablation studies would be conducted, and different threshold would be tested to evaluate the optimal value. A larger population size and more generations would be adopted.