A Novel Approach to Optimising Convolutional Neural Networks based on the Ant Colony Optimiser

Ted Michael Arlidge [redacted]
Computer Science, 3rd Year
Guildford, United Kingdom
ta01051@surrey.ac.uk

Saul Roche [redacted]
Computer Science, 3rd Year
Guildford, United Kingdom
sr01396@surrey.ac.uk

Pascal Duncan Lek Hou U [redacted]
Computer Science, 3rd Year
Guildford, United Kingdom
pu00064@surrey.ac.uk

***Abstract*—This paper explores the application of Continuous Ant Colony Optimisation (CACO) for optimising Convolutional Neural Networks (CNNs) and compares its performance to popular gradient descent and population-based optimisation algorithms. The study evaluates these optimisation techniques in the context of training CNNs to classify images from the CIFAR-10 dataset, a standard benchmark widely used for training in machine learning problems. Our analysis shows that gradient descent continues to outperform population-based in terms of efficiency and accuracy. However, CACO achieves the best loss due to its superior ability to reach generalised solutions, and we discuss the relative advantages and limitations which lead to this result.**

I. Introduction

Image classification is a fundamental task in computer vision which aims to categorise images into predefined classes. Although this is a trivial task for humans, achieving similar performance in artificial systems is challenging due to the extreme variability of objects within the same class. Gradient descent is the most common algorithm for optimisation problems, however some optimisation problems are too complex and gradient descent methods can be insufficient in finding the optimal solution. Population-based optimisation algorithms such as Genetic Algorithm and Ant Colony Optimisation can be employed instead to solve this problem. This article presents a modified Continuous Ant Colony Optimisation and its performance compared to Genetic Algorithm and Adam, a baseline gradient descent method. In section III, we give a detailed overview of the Convolutional Neural Network architecture we chose to evaluate these algorithms on and why it was chosen. Thereafter, in section IV we provide a description of Continuous Ant Colony Optimisation with details of our implementation and justification for our decision to use it. In section V, we provide numerical results comparing the three algorithms, with an explanation of the relative advantages and limitations which lead to those results. Finally, in section VI we attempt to implement NSGA-II to train the model as a bi-objective problem, and discuss whether this is effective for our specific problem.

II. Literature Review

A. Convolutional Neural Networks (CNNs) and the CIFAR-10 dataset

CNNs are designed to resemble the human visual system, and operate by extracting features from input data and using them to classify the image.

The Tiny Images dataset [1] was created in 2006 and contains 80 million 32x32 images classified into 53,464 groups. These groups are each a noun, which were used to automatically search and download the corresponding images available from the internet at the time. These were collected by downloading images from search engines available at the time, using every English noun in the WordNet database [2]. Due to the nature of online image search technology, the given labels in the Tiny Images dataset are extremely unreliable and can only be used as a rough label. Additionally, as the Tiny Images dataset has never been carefully examined, there are several classes labelled after offensive nouns such as ethnic slurs and derogatory terms [3]. For this reason, as of 2020, the Tiny Images dataset has been formally taken offline and MIT has requested for the community to refrain from continuing to use it in the future. The CIFAR-10 dataset [4] is a labelled subset of the Tiny Images dataset consisting of 60,000 images categorised into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class contains 6000 images, split into 5000 for training and 1000 for testing. It was created by a group of students manually filtering out mislabelled images using the rough labels given. Although the CIFAR-10 dataset was derived from the Tiny Images dataset, it is not affected by its shortcomings due to each image being manually checked.

ResNet-50 is a 50 layer deep convolutional neural network architecture which has been trained on large datasets to achieve state-of-the-art results in image classification. The average performance of ResNet-50 on the CIFAR-10 dataset is 78.10%, with the lowest being 61.90% on the cat class, and the highest being 90.80% on the airplane class [5]. These results were obtained by using a pretrained version of ResNet-50 [6],

which was trained on 1.28 million images in 1000 classes from the ImageNet dataset [7]. Therefore, these results can be significantly improved by using the ResNet-50 architecture to train a model on the CIFAR-10 dataset, and this performance can not be used to compare with our models results.

B. The Adam Optimiser

Gradient descent is the most common algorithm for optimisation problems, where the goal is to minimise the loss function by calculating the gradient and iteratively updating the model's weights to follow the gradient towards the optima. Adaptive Movement Estimation (Adam) is a gradient descent optimisation algorithm which combines the benefits of two gradient descent methods, Momentum, and Root Mean Square Propagation [8] [9].

Momentum takes the exponentially decaying average of past gradients, which is used to accelerate the convergence gradient descent by effectively dampening the change in the gradient. In Adam this is given by:

$$m_t = \beta_2 \cdot m_t + (1 - \beta_2) \cdot (\delta L / \delta w_t)$$

Root Mean Square propagation (RMSprop) takes the exponentially decaying average of past squared gradients. This term V scales the learning rate according to the gradient, allowing for the benefits of a decaying learning rate while not suffering from the risk of the learning rate being permanently decayed. This is given by:

$$v_t = \beta_2 \cdot v_t + (1 - \beta_2) \cdot (\delta L / \delta w_t)^2$$

As both m_t and v_t are initially set to zero, during the initial time steps while decay rates are small, they are biased towards zero so the bias-corrected $\hat{m}_t$ and $\hat{v}_t$ are computed:

$$\hat{m}_t = m_t \div (1 - \beta_1^t)$$

$$\hat{v}_t = v_t \div (1 - \beta_2^t)$$

These are used to yield the Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

C. Population-based optimisation algorithms (POAs)

For complex optimisation problems involving multimodality, discontinuity, and noise, traditional gradient-based methods are no longer sufficient, and different kinds of optimisation algorithms must be employed instead [10]. POAs are often used to solve these problems due to their ability to avoid getting trapped in local optima and more effectively explore the solution space due to the inclusion of randomness in its operators. POAs begin with an initial population of individuals which search the solution space and eventually converge into a solution.

There are two main categories of POAs: Evolutionary Algorithms (EAs) and Swarm Intelligence Algorithms (SIAs). Genetic Algorithms (GAs) and Differential Evolution are popular EAs which try to mimic biological evolution by using selection, mutation and reproduction. Whereas SIAs mimic the collective behaviour of decentralised systems in nature, with examples being Particle Swarm Optimisation and Ant Colony Optimisation.

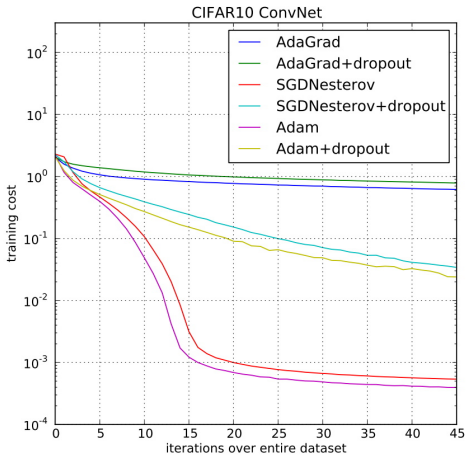

Fig. 1. Training cost of different optimisers on the CIFAR-10 dataset over 45 epochs [9]

D. Ant Colony Optimisation (ACO)

ACO is a POA developed by Marco Dorigo, inspired by the behaviour of foraging ants [11]. Initially used for solving combinatorial problems, there have been commendable attempts to extend the algorithm toward continuous domains [12]. The outline of ACO is that ants choose to move from node x to y based on a probability calculated using;

1) **Trail Level:** This is controlled by the pheromones deposited by previous ants on each edge.
2) **Path Attractiveness:** Determined based on a heuristic that indicates the *a priori* desirability of that move.

A high level overview of this algorithm is given in Algorithm 1. Where each iteration is seperated into 3 phases.

Algorithm 1 Ant Colony Optimiser

```
while termination conditions not met do
    generateSolutions()
    daemonActions()
    pheromoneUpdate()
end while
```

1) **GenerateSolutions():** encapsulates the part of the program that initialises ants, trail levels and attractiveness. Different applications of ACO will have different initialisations.
2) **daemonActions():** is the algorithm moving ants from node x to y corresponding to a more complete solution, based on the probability p_{xy}^k for each ant k given by

$$p_{xy}^k = \frac{(\tau_{xy})^\alpha (\eta_{xy})^\beta}{\sum_{z \in \text{Allowed } y} (\tau_{xz})^\alpha (\eta_{xz})^\beta}$$

where τ_{xy} and η_{xy} are trail level and attractiveness of the edge xy respectively. α and β are parameters to control the influence of τ and η respectively. τ_{xz} and η_{xz} represent other possible state transitions.

3) **pheromoneUpdate():** Trails usually update when all ants have completed their solutions, and vary by different implementations, but commonly make use of a pheromone evaporation coefficient which allow unfit solutions to be less likely to be selected over time.

E. Our Contribution

This project compares various existing optimisation methods for optimising a CNN against a modified ACO algorithm. Our proposed algorithm is modified to treat ant solutions as points in continuous space and uses a dynamic probability distribution modelled after pheromone calculations in traditional ACOs. However, we utilise a fitness function to determine probabilities instead of pheromones and attractiveness.

III. Selected Architecture

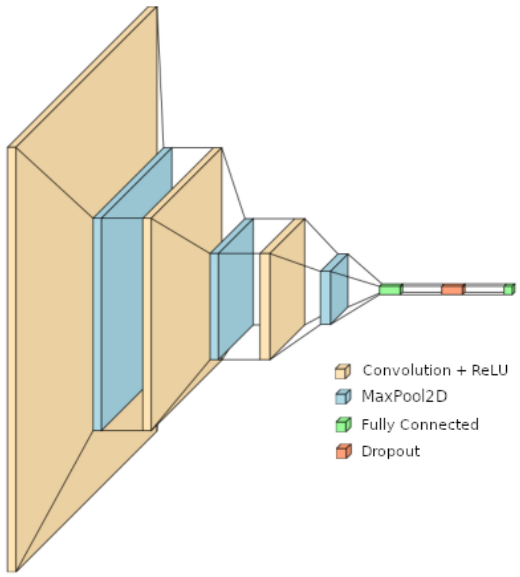

Fig. 2. Diagram of CNN Architecture

The architecture selected for this project is a Convolutional Neural Network. A Convolutional Neural Network was chosen due to its high accuracy on image classification problems while being simple to implement. A detailed overview of the architecture is defined as:

1) Convolutional Layers:
   a) The first layer applies 32 filters of size 3x3 over the input image.
   b) The second layer applies 64 filters of size 3x3 over the output of the previous layer.
   c) The third layer applies 128 filters of size 3x3 to the output of the previous layer.
2) Pooling Layers:
   a) After every convolutional layer, a max-pooling is applied, defined by a 2x2 filter with a stride of 2 for downsampling.
3) Fully Connected Layers:
   a) The first layer takes the flattened feature map of size 2048 and maps it to 256 neurons, followed by a ReLU activation and dropout of 0.5.
   b) The second layer takes the output from layer 1 of size 256 and maps it to 10 output neurons for the 10 classes in CIFAR-10.
   c) Softmax is then applied to the output to give a probabilistic output of the class predictions.

The configuration of the model is illustrated using the layer diagram in Figure 2

This configuration proved to be effective for image classification in the CIFAR-10 dataset with a high accuracy and minimal overfitting. A dropout of 0.5 was included to avoid overfitting and improve generalisation. The three convolutional layers were sufficient to process the image to a ready state for the fully connected layers.

IV. Training Approach

Our chosen algorithm is an adaptation of Continuous Ant Colony Optimisation (CACO). [13] It starts with a uniformly distributed population of individuals (ants) as a point in n dimensional space within a set of bounds defined in the hyperparameters, and calculates the fitness of each individual. After every individual is evaluated, a probability distribution function is generated with respect to the fitnesses of each ant and the global best ants of previous iterations using multivariate normal distributions.

The algorithm we implemented is based on the algorithm presented in Algorithm 2.

The next generation of ants is produced randomly according to the probability distribution function, with a high density of new solutions in high-fitness areas of the search space, and a lower density of new solutions in lower-fitness areas.

The global best ant of each iteration is kept in a list to allow the effect of that ant to persist across generations to reduce the risk of losing a global optimal solution. However, the probability distribution function of each global best evaporates over each iteration to make more recent global optima more likely to be chosen to avoid focusing too much on less optimal global optima.

Our chosen algorithm differs from the paper by making the probability distribution function using all ants and scaling their contribution based on the fitness of each ant, making fitter areas in the search space more likely than less fit areas. This creates a balance of exploration and exploitation by focusing the search in local optima while allowing for some solutions to explore outer areas. This will reduce the convergence speed, but allows each individual a chance to explore their area in the search space. After each iteration, the covariance matrix (cov) is reduced, this allows for the probability of a chosen new ant

Algorithm 2 Continuous Ant Colony Optimisation (CACO)

Require: An objective function $f(x) \in \mathbb{R}$, where $x \in \mathbb{R}^n$

ants $\leftarrow$ Initialise n ants according to a uniform distribution
cov $\leftarrow$ identity covariance matrix
best_ants $\leftarrow \emptyset$
for each iteration **in** epoch_count **do**
    for each batch **in** train_loader **do**
        fitnesses $\leftarrow$ evaluate all ants
        best_ant $\leftarrow$ iteration best solution
        add best_ant to best_ants
        for $i = 1$ **to** n **do**
            ant_temp $\leftarrow$ choose a random ant according to the fitness of the ants
            ant $\leftarrow$ sample ant randomly according to a multivariate Gaussian of ant_temp with covariance matrix cov
        end for
        scale down the value of cov
    end for
end for
$S_G^{\text{best}} \leftarrow$ find best solution in best_ants **return** S_G^{best}

to be closer to other existing ants, thus allowing exploitation in later generations.

Our implementation of CACO may have some limitations in high-dimensional search spaces. The method of sampling the next generation involves generating a random multivariate normal in the same dimensionality as the search space. The sampling method requires generating a multivariate Gaussian with a high number of dimensions (2570 in the case of our example model), which uses the Cholesky decomposition with time complexity of $O(n^3)$ where n is the number of dimensions. This is manageable with our example by utilising multiprocessing, but may become intractable for higher-dimensional problems. Multivariate normal distributions are very flexible, allowing for a vast range of shapes using different covariance matrices, and allows for a continuous, smooth shape without cutting off suddenly. However, a possible area to improve the performance of the generation of new individuals is to have a simpler probability distribution function, such as a radial distribution which may have significantly better runtime performance at the cost of losing the benefits of the normal distribution.

The CACO algorithm requires sensitive problem-specific fine-tuning of hyper parameters and implementation for some problems. The function for reducing the covariance matrix in the multivariate normal distributions worked well for our image classification task, but may need to be adjusted for simpler or more complex problems.

CACO was chosen as the optimiser for the convolutional neural network because of its flexibility in implementation, and ability to converge to global optima. With our variation of the implementation of CACO, it has a balance of exploration and exploitation throughout the training process by fine-tuning the Gaussians for each ant by reducing the covariance matrix. In our experiments, CACO outperformed the GA we implemented with the same population size, demonstrating better convergence for the image classification task. Additionally, our implementation is highly generalisable, capable of handling problems with any dimensionality, provided it is not excessively large (e.g. many thousands of dimensions). For extremely high-dimensional problems, alternative simpler probability distributions than the multivariate normal may be considered to maintain computational feasibility. Furthermore, the CACO algorithm is not constricted to image recognition tasks, it can be adapted to any fitness function, making it a possible choice for a wide range of optimisation problems.

V. Results

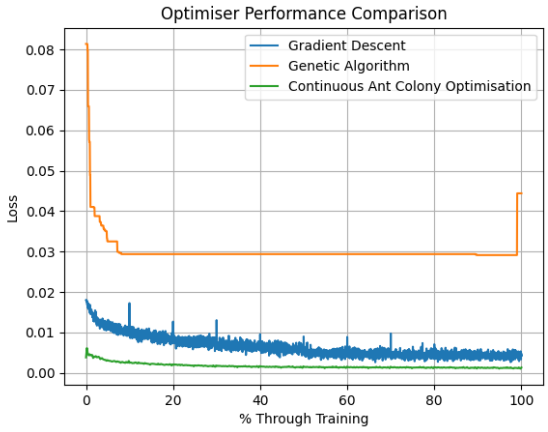

Fig. 3. Loss calculated over training CNN with each optimiser

The results in table I are generated using values extracted from training the model with 3 different optimisers. GA and CACO are only applied to the final fully connected layer of our model to save computation for this project, further work may involve extending these metaheuristic algorithms to optimise the whole model.

Method	Accuracy	Total Loss
Adam	**0.7569**	55.36522344
GA	0.3945	479.9376783
CACO	0.6162	**5.262152837**

TABLE I
Performance of Optimisation Methods

In line with our expectations, we found that Adam outperforms both the GA and CACO optimisers with an accuracy of 75.69%, GA and CACO getting accuracies of 39.45% and 61.62% respectively. Our CACO optimiser achieves the best loss even surpassing Adam, which is not far from our expectations because of the optimiser's ability to reach generalised solutions. As the cross entropy loss function calculates loss based on the probability output of the model, it suggests that CACO makes small errors in probability for wrong

classifications, but at a larger frequency than that of Adam. This makes CACO a candidate solution for implementations that require very low overfitting.

GA is the lossiest optimiser in this comparison which can be attributed to the nature of mutation and selection of candidate individual solutions. There is no objective function that an individual can optimise from, solutions converge merely by selection and have no means of minimising loss directly. GA is likely to converge to higher accuracies and lower loss with more generations. However, doing so will require large computational requirements and extending this toward multiple layers of the model will be impractical for simple use cases.

VI. Training as a Bi-Objective Problem

The results shown in Figure 4 are derived from training our model on a bi-objective problem. For this project we used the DEAP library to implement the NSGA-II algorithm. Using the parameters set in table II, we run the algorithm for 15 generations, a population of 64 and with a crossover probability of 0.9. The dimensions of each individual correspond to the number of weights in the final layer and are arbitrarily bounded to -0.5 and 0.5 to prevent individuals from being initialised in too large a search space. These parameters are chosen to minimise computational intensity and encourage exploitation of optimal solutions.

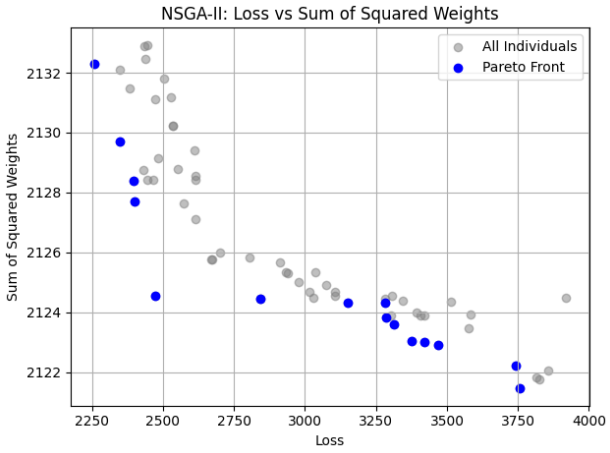

Fig. 4. Results of NSGA-II Training

Parameter	Value
NGEN	15 (Number of generations)
MU	64 (Number of individuals in population)
CXPB	0.9 (Crossover probability)
NDIM	size of fc2 weights (Number of dimensions in individual)
BOUNDS	$[[-0.5, 0.5]$ for each dimension in NDIM]

TABLE II
Parameter settings for NSGA-II

The training problem is formulated to minimise both loss and the sum of squared weights. The loss calculated is the sum total loss calculated over the training and has virtually the same usage as average loss. The Gaussian Regulariser is the minimisation of the sum of squared weights. The reason for doing so is justifiable when weights have significant impact to the model overfitting and having bad generalisation. Minimising loss (and by proxy increasing accuracy) on the training set may lead to overfitting when not taking into account the ability of the model to generalise. When weights tend toward higher absolute values, the model risks suffering from overfitting.

With the results presented in section V, our metaheuristic models have a good ability to generalise and do not tend to overfit, which may be attributed to the ability of metaheuristic algorithms being able to compute global minima and not being stuck in local optima. In the case of Adam, most gradient descent optimisers (including Adam) address this issue using weight decay which performs ridge regression on weights, improving generalisation. Therefore, we believe that the training problem is better formulated as a single objective optimisation problem.

VII. Conclusion

This study established that gradient descent remains the most effective optimiser for image classification, demonstrating better performance metrics and less computation time compared to alternative approaches. However, the CACO algorithm proved to be significantly more effective population-based metaheuristic optimiser than our GA. Given its capacity to optimise high-dimensional and non-differentiable search spaces, the CACO algorithm can be used in a vast range of optimisation problems. Further experimentation is necessary to determine the extent at which this algorithm can be used in optimisation problems.

A. Further Work

The complexity of the CACO algorithm provides opportunities for refinement and generalisation. Automation of some hyperparameters, such as linking the initial bounds of the search space with the covariance matrix in the normal distributions or correlating the covariance matrix to the population size, could improve usability and performance. In particular, modifications to the covariance matrix reduction function is a point of interest as the function is critical for balancing exploration and exploitation of the algorithm.

Our CACO algorithm will require more testing against other metaheuristic optimisers to determine its performance in the state of the art. Further experimentation is necessary to determine the extent at which this algorithm can be used in other optimisation problems.

This project uses the DEAP library to implement the GA and NSGA-II Algorithms, parallelising the algorithm with multiprocessing or SCOOP can speed up computation. Further work in finding optimal parameters and running for more generations may likely find more optimal solutions and will benefit from parallelised computation.

REFERENCES

[1] [Online]. Available: https://groups.csail.mit.edu/vision/TinyImages/

[2] [Online]. Available: http://wordnet.princeton.edu/

[3] A. Birhane and V. U. Prabhu, "Large image datasets: A pyrrhic win for computer vision?" in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2021, pp. 1536–1546.

[4] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[5] N. Sharma, V. Jain, and A. Mishra, "An analysis of convolutional neural networks for image classification," *Procedia computer science*, vol. 132, pp. 377–384, 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[8] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[9] [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=Adam

[10] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms–a survey," *Swarm and evolutionary computation*, vol. 44, pp. 695–711, 2019.

[11] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[12] K. Socha and C. Blum, "An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training," *Neural computing and applications*, vol. 16, pp. 235–247, 2007.

[13] K. Socha, "Aco for continuous and mixed-variable optimization," in *International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer, 2004, pp. 25–36.

APPENDIX

Summary of Contribution For Each Member of our Group: The Introduction and Abstract summarising the background of the problem and content of our work was written by Ted. The Literature Review was also largely written and researched by Ted while Subsection II-D describing the Ant Colony Optimiser was written and researched by Pascal. The closing paragraph in II-E summarising our contribution was largely written by Pascal with help from Ted and Saul.

Saul designed and implemented the architecture of our model and wrote Section III describing it. Saul also implemented the Adam and GA optimisers, designed the chosen algorithm and wrote Section IV detailing and describing it. Saul wrote the code that generated the visual performance metrics shown in Figure 3 and the values obtained in Table I. Section V is collaboratively written between Ted and Pascal.

The code for the NSGA-II implementation that produced Figure 4 was written and implemented by Pascal for Section VI, which was also written by Pascal. Section VII concluding our work and discussing future directions for our work was written collaboratively between Saul and Pascal. The final version of this report was collated and produced by Pascal, with proofreading done by Ted and Saul.

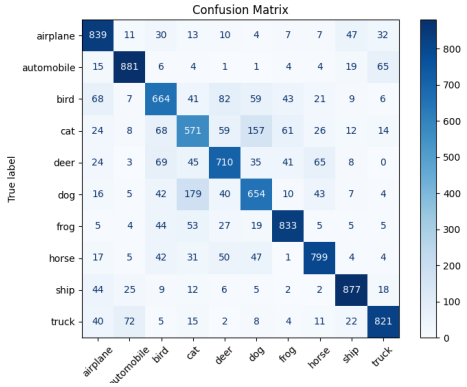

Fig. 5. Confusion Matrix: Adam

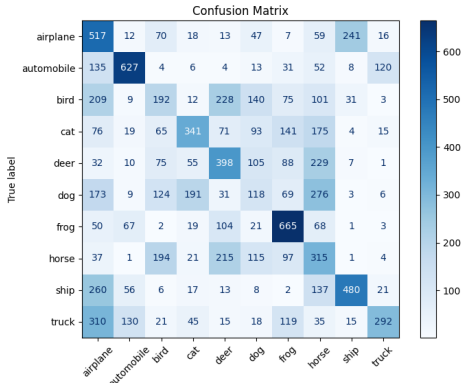

Fig. 6. Confusion Matrix: Genetic Algorithm

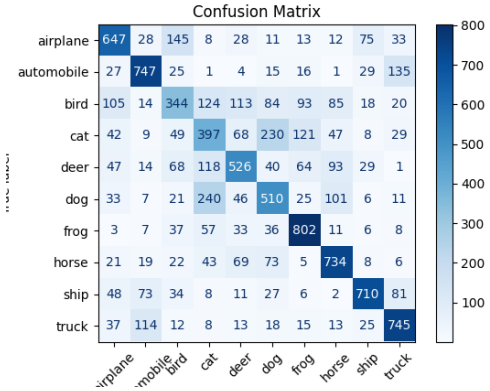

Fig. 7. Confusion Matrix: Continuous Ant Colony Optimiser