Using Population-Based Algorithms to Fine-Tune a CNN for Image Classification

Author Name, 0000000
Department of Computer Science
University of Something
Something, ST
email@something.edu

Author Name, 0000000
Department of Computer Science
University of Something
Something, ST
email@something.edu

Author Name, 0000000
Department of Computer Science
University of Something
Something, ST
email@something.edu

***Abstract*—In this report, we compare the performance of a convolutional neural network for the classification of images in the CIFAR-10 dataset using three different optimisation algorithms for the minimisation of the error metric in the final layer, where previous layers have been pre-trained using stochastic gradient descent (SGD). Using SGD and social learning particle swarm optimisation (SL-PSO) as benchmark optimisation algorithms, we find our chosen algorithm (a modification of SL-PSO) performs better than SL-PSO, but fails to reach the accuracy level of SGD. We also compare the performance of the derived model when viewed as a single- and bi-objective optimisation problem (adding a Gaussian regulariser as the second objective), finding the single-objective solution performs better than the bi-objective solution.**

***Index Terms*—optimisation, classification, neural networks, particle swarm, gradient descent**

I. Introduction

Neural networks are a subset of machine learning models that aims to approximate a function that best fits the dataset. Their design is based on the complex neurological pathways found in the human brain of that enables them to learn by making millions of connections between other neurons. These neural networks can learn in various ways, reinforcement learning, semi-supervised learning, and supervised learning. In this paper we will discuss only supervised learning. In supervised learning, the ground truth labels of the dataset are available, and we can directly train the network directly based on the difference between its predictions and the actual value, calculating a resulting loss value. This allows for methods such as backpropagation to teach a network to perform well on image classification. This project aims to optimise neural network training by comparing the performance of single-objective (SGD, SL-PSO, SSPSO (our own algorithm)) and multi-objective (NSGA-II) optimisation methods. The primary objective is to maximize prediction accuracy for unseen images, and we explore simultaneously minimising regularisation to prevent overfitting. SGD, a common gradient-based algorithm, will be evaluated alongside two evolutionary algorithms inspired by bird flocking behaviour: SL-PSO SS-PSO. NSGA-II, a multi-objective evolutionary algorithm, is evaluated to balance prediction accuracy and regularisation. The effectiveness of each method will be assessed based on prediction accuracy and regularisation measures. The findings will provide valuable insights into the trade-offs between prediction accuracy and regularization, as well as the suitability of single-objective and multi-objective optimization approaches for neural network training.

A. Literature review

1) Architectures: Convolutional neural networks (CNNs) are the most common architecture for image classification. These networks are specifically designed to work with spatial data such as images by creating complex feature maps from them through deep learning. This allows the network to be unaffected by translations of input images, making it more robust in its responses. Common architectures that work well on unseen image classification tasks are AlexNet, GoogLeNet, ResNet [1], and VGG [2].

a) AlexNet: AlexNet is one of the earliest CNNs that revolutionised deep learning. It featured RELU activation functions instead of tanh functions which was the standard at the time [?]. It also allowed for scalability in terms of being able to run on multiple GPUs and addressed overfitting issues by using methods such as dropout. AlexNet uses 60 million parameters which contributed to its accuracy , however, due to its age, newer architectures achieve greater parameter efficiency. AlexNet does not have great flexibility, as removing layers greatly affects its performance [3]. This is important as we want to improve running times of our population-based algorithms as much as we can, and this may involve scaling the model down [4]. While AlexNet was important for the development of CNNs, its aging architecture allowed for newer models to surpass it in terms of accuracy and efficiency, which what we aim to utilise in our work.

b) ResNet and GoogLeNet: Newer architectures like ResNet and GoogLeNet improve on AlexNet by being much more efficient for training deep networks [5]. This is done through skipping connections in regard to ResNet to solve the vanishing gradient issue along with dimensionality reduction through 1x1 kernels with GoogLeNet. While ResNet and GoogLeNet both offer these powerful features, our original problem does not have a large dimensionality, as we work with very small images in comparison to what the network was originally designed with [6]. ResNet's computation complexity is also much higher due to its skip connections and effectiveness when training deep networks. This further increases the number of parameters needing to train which puts a strain on our resource bound environment. We ideally want to find a balance between accuracy and performance, which we find with VGG, which offers powerful training mechanisms with a simpler, more flexible architecture [7] as we explain later in section 2.

c) Transformers: Transformers are commonly used for NLP tasks but can be modified for image classification as seen by the Vision Transformer [8] with strong performance on CIFAR-10 [9]. Transformers often have many more parameters than CNNs, requiring significantly more training data to surpass the performance of a CNN, and they are slower to run in resource-bound environments [10]. In Section 2 we critically evaluate the models and decide on the architecture that bests suit our problem.

2) Training algorithms:

a) Gradient-based algorithms: An optimiser is required by the architecture to train the model on the input dataset by optimising the weights of each layer in the network. Gradient based optimisers such as stochastic gradient descent (SGD) are well studied and benefit from large flexibility to be used in many problems without lots of tweaking. Additionally, convergence is fast when the correct hyperparameters are chosen. Despite this, gradient based methods often suffer from vanishing/exploding gradients [11] because of backpropagation and as a result, hinders convergence. Importantly, they are sensitive to the landscape of the problem and can fall into local minima or plateaus along with not working with discontinuous problems. As a result, optimisers such as ADAM have been created to try and circumvent problems such as rugged landscapes.

b) Population-based algorithms: Population based optimisers such as Genetic Algorithms (GA) and Particle Swarms are an alternative method for optimisation. GA's use a set of potential solutions acting as an initial population which then goes under various techniques such as crossover, mutation, and selection to provide a set of offspring that may achieve better results. The diversity of the population allows for greater exploration of the search space, increasing the chances of reaching the global minima. Non-differentiable objective functions and noisy fitness landscapes are much better handled by these than other optimisers [12]. Population based optimisers are much more computationally expensive however due to the quantity of evaluations required, increasing accordingly to the population size. Additionally, they require much more careful tuning of the hyperparameters for the optimiser to converge and therefore is much less flexible when presented new problems [13].

In our work, we look at Particle Swarm Optimisation (PSO), an optimisation technique, first introduced in [14]. Canonical PSOs use a set of particles as potential solutions. They do not use selection, crossover or mutation operators and instead works on a personal best and global best system, so less hyperparameters need tuning to work. The movement of these particles mimics the behaviour of bird swarms in nature, where individuals learn from the best member of the group. This learning process updates the particles speed and positions leading to improved fitness [15].

However, traditional PSOs are not well designed to handle problems of large dimensionality. Particles may become trapped in local minima, preventing them from reaching the global optimum [16]. To address this issue, we explore strategies for escaping local minima such as incorporating GA techniques like particle mutation. These techniques form the foundation of our third proposed algorithm SSPSO.

SL-PSO uses a 3-stage process for optimisation involving evaluation, swarm sorting and behaviour learning. In particular, the behaviour learning aspect consists of worse particles learning from better ones and are more influenced the lower they are in the rankings. It is much more efficient with problems of high dimensionality and as a result, can converge faster at the cost of its complexity.

B. *Our contribution*

In our project, the following contributions make significant contributions to the field of neural network optimisation:

- We address the problem of SL-PSO's getting stuck in a local minimum by introducing a new genetic algorithm SSPSO that combines the strengths of PSO and genetic algorithms (GA). SSPSO includes PSO's leader-following approach and GA's particle mutation. The algorithm applies mutation dynamically within the main loop (between behaviour learning and the generation of swarm P(t+1)) to particles once a rapidly decreasing standard deviation indicates that particles are converging early with the aim to enhance exploration capabilities leading to improved performance over classical PSO optimisers.
- We compare SGD and classical PSO methods to our own proposed algorithm by evaluating the accuracy and loss of the neural network on test images. We find that SGD outperforms evolutionary based algorithms and remains the leader in neural network optimisation methods.
- We compare single-objective and multi-objective methods.

II. Architecture

A. *Description*

In our work, we use a Convolutional Neural Network based on a simplified version VGG with a several 3x3 convolutional layers stacked as our throughput.

We utilise the VGG architecture as it a more parameter efficient model in comparison to architectures like ResNet which allows the model to run faster in resource-bound environments [?]. Additionally, while other architectures may be more powerful in terms of deep learning, they require more training data to be fully taken advantage of. The VGG architecture provides a strong balance between flexibility and accuracy whilst also being more efficient to train.

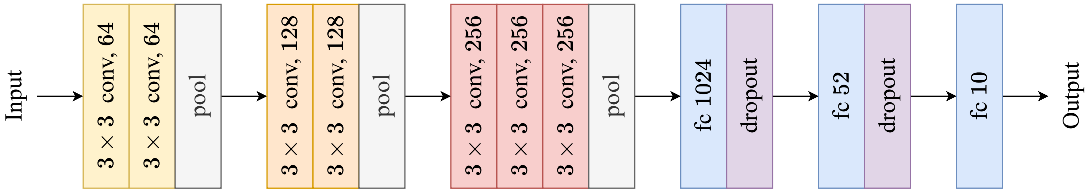

Fig. 1. Model architecture, which is a modified version of VGG.

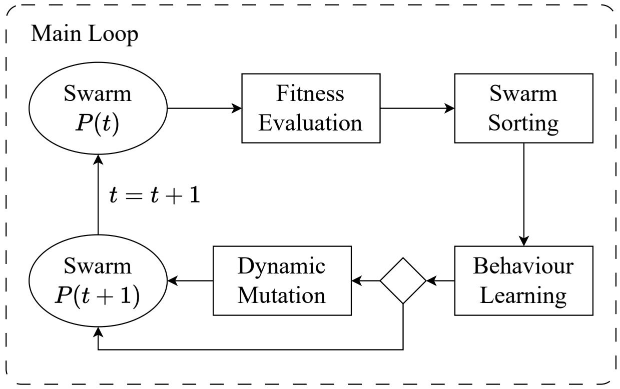

Fig. 2. Example of a figure caption.

B. Justification

As previously stated, population-based optimisers provide significant computational overhead to run. We utilise a less complex VGG architecture to run and compare all three optimisers in a realistic amount of time. By using a simpler architecture and only optimising the final layer, we reduce the number of resources needed to process each forward pass in our Genetic Algorithm as this happens several times per generation, which adds up throughout the optimisation process. Additionally, the VGG architecture provides a strong foundation for the model and the optimisation process in terms of accuracy to performance overhead [17]. To avoid overcomplicating the problem, we avoid using transformers.

III. Training algorithm

A. Description

Our chosen algorithm Social Snake Particle Swarm Optimisation (SS-PSO) extends SL-PSO. SS-PSO adds a fourth step of dynamic mutation based on the standard deviation of the population σ_{pop} at the end of the generation cycle, as shown in Figure 1. Like how behaviour learning makes the worst particle learning the most from all the other particles, dynamic mutation mutates the worst particles the most.

By calculating σ_{pop}, we identify when the particles have converged into a single area, indicating the presence of a local minimum.

B. Justification

We chose to base our algorithm off PSO due to the significant advantages it has over GAs. These include:

- Being simpler to implement than a canonical GA due to a smaller number of operators needing to be implemented. The more operators you have, the more tuning of hyperparameters for your problem are needed and consequently makes the optimiser less flexible for different architectures as a result. This makes the process more tedious in the case that a switching of network architecture is needed.
- Additionally, PSOs do not require creating or deleting existing individuals in the population and instead just requires a simple update to the particles velocity and direction, using less resources as a result.
- Importantly, PSOs are often faster at converging towards the global optimum than GAs as seen by the results of [?] where they compared the results of the two optimisers.

C. Euclidean distance

We calculate the Euclidean distance

$$dist(p, q) = |q - p| \tag{1}$$

of each particle's fitness $fitness(\boldsymbol{pop}_i)$ compared to the fitness $fitness(\boldsymbol{pop}_0)$ of fittest particle $\boldsymbol{pop}_0$ at generation t and give particles that are closer to $\boldsymbol{pop}_0$ a greater chance of mutating than others, encouraging the particles to escape the local minimum, leaving the previous best particle behind.

Euclidean distance was used similarly in [18] to measure the distribution of particle positions around the global best particle. Inertia and acceleration parameters were adaptively updated depending on a particle's estimated evolutionary state.

Based on the dynamic influence of inertial weights on particles can pull particles out of local minima or conversely focus the search around one area to help the optimiser converge to an optimal solution quicker and enhances the performance of classical PSO [19].

D. Mutation operator

We do not mutate the current best particle to act as a checkpoint for the best fitness so far in the case that the other particles have much worse fitness after mutation or if the area was previously not a local minimum and can then move back towards that particle later if needed.

Adaptive and non-adaptive mutation size methods were proposed by [20] and [21] with PSO encouraging leaders to explore more and jump out of a local minimum to lead the other particles to better positions however the methods still led to premature convergence. We took this mutation initiative and applied a probability of mutation to all particles to put more

Algorithm 1: SS-PSO's addition to SL-PSO main loop

```
input : $\sigma_{pop}$, standard deviation of population's fitness
        values
        $\sigma_{thr} = 1.0$, threshold for $\sigma_{pop}$
        $\boldsymbol{pop}_0$, fittest particle in population
if $\sigma_{pop} < \sigma_{thr}$ then
    MutationTick $\leftarrow$ MutationTick $+1$
    else
        MutationTick $\leftarrow 0$
    end
end
if $MutationTick \geq 3$ then
    for $i = 1 : m$ do
        Calculate
          $d_i = dist(fitness(\boldsymbol{pop}_i), fitness(\boldsymbol{pop}_0))$
        Add to a data structure $\boldsymbol{d}$ containing distances
          for the whole population.
    end
    Rescale the distances in $\boldsymbol{d} = \{d_i, \ldots, d_m\}$ such
      that $\forall d \in \boldsymbol{d} : 0 \leq d \leq 1$ and the smaller the value,
      the closer it is to 1.
    for $i = 1 : m$ do
        Mutate the particle $\boldsymbol{pop}_i$ with DEAP's
          mutGauss($\mu = 0$, $\sigma = 0.5, d_i$).
    end
end
```

encouragement on the 'jumping out' evolutionary state. We believe giving a mutation probability to more than one particle increases the likelihood the swarm will explore elsewhere.

The pseudo-code for the Social Snake part of the PSO is shown in Algorithm 1.

Escaping local minima with SS-PSO is not guaranteed but the chance of converging towards the global minimum instead of local minima, as seen with SL-PSO, greatly increases. However, SS-PSO requires more computational resources to run compared to SL-PSO due to needing to mutate most particles. For SS-PSO work at its best, an appropriate value of standard deviation to start mutating at must be chosen along with the σ_m and μ_m values of the mutation operator itself. In our work, we utilise DEAP's Gaussian mutation function for mutating the population.

We choose mutation over other methods as this is an explored topic in papers such as [22] and [23] that both focus on implementing the genetic algorithm operator as their method for escaping local minima. The mutation in both papers randomise part of the particle according to a given probability, essentially repositioning the particle to be outside of the local minimum. The way they determine when to start the mutation of particles is different to our implementation. To improve the performance of the algorithm, we use a simpler determination by just using the standard deviation to make an education guess if the particles have reached a minimum. Additionally, mutation allows for greater flexibility when implementing different architectures in comparison to other approaches [24] to perform a similar thing as there are less hyperparameters to be tuned when doing so.

We use a probability to mutate instead of mutating all the particles by a certain amount for two reasons. The most important is that we do not want the position of a particle to drastically change, instead we only want to assist it in getting out of a minimum by shifting some of the decision variables slightly. As we do not know which decision variables in a particle are good during the algorithm, we want to give a chance that the particle will not mutate which will not affect the good decision variables, improving performance as we do not need to mutate as many particles and additionally reducing any bias that may occur while doing so.

TABLE I
SUMMARY OF PARAMETERS AND RESULTS

Optimiser	SGD	SLPSO [25]	SSPSO
Swarm size	-	185	153
Social influence	-	•	•
Learning probability	-	•	•
Dynamic Mutation	No	No	Yes
Validation Loss	•	•	•
Testing Loss	•	•	•
Validation Accuracy	•	•	•
Testing Accuracy	•	•	•

IV. RESULTS

A. Setup

The first step involves evaluating the fitness of each particle in our search space. To do this, we generate our population with the dimensionality of n=op+b, where b is the number of bias weights, o is the number of classes and p is the number of parameters for each output. Each particle is a different version of the final layer. To evaluate a single particle, we split it into weights and biases and place it into the final layer of the model. We then perform a forward pass of the network with a selected batch size of training images and calculate the cross-entropy loss for that particle to act as our fitness function. After doing this for the population, we then sort the swarm by fitness in ascending order and then perform the same particle update as SL-PSO using a velocity equation.

A baseline model was created using Stochastic Gradient Descent. To compare the efficacy of population-based methods with the typical gradient based method for the classification of images in the CIFAR-10 dataset, fine tuning was performed on the final fully connected layer. The initial population size for the population-based optimisation algorithms is set to n. For our network, b=10, o=10, p=84, therefore n=850.

All experiments were conducted on a computer running Ubuntu 22.04 LTS with a 12-core Intel Core i7-8700 CPU 3.20GHz with 32GB of RAM and an Nvidia Quadro P4000 with 8GB of VRAM.

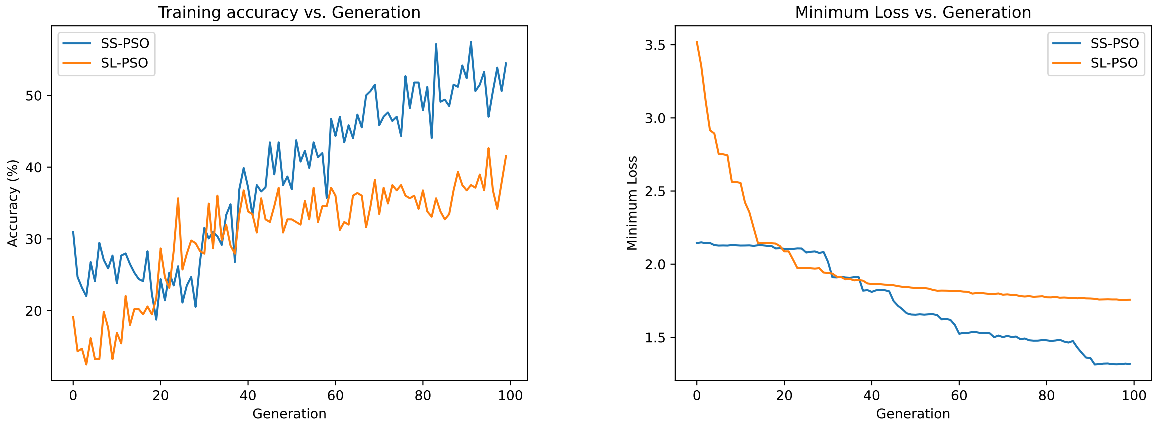

Fig. 3. Placeholder for the accuracy and loss plots

B. *Interpretation*

Firstly, we observe the training performance of SGD, acting as our benchmark. We can see that within 10 epochs, the optimiser reaches a great score of 85

In comparison, both the population-based optimisers behave very differently. When searching in the space of between +/- 0.1, we can see a much slower rate of convergence in loss along with slower increases in accuracy over time. SL-PSO, by generation 100, reaches an accuracy of only 41.5% on the training set, dramatically lower than SGD with 38.5% on the unseen test set. Additionally, we also see that the accuracy and loss both plateau early on and does not improve further. Our custom algorithm, SS-PSO, shows an expected improvement from SL-PSO where it reaches an accuracy of 54% on the training set, with a 51.5% accuracy on the test set. This is a substantial increase of 10% on the unseen data. Observing the average loss of the two algorithms we can see SS-PSO to appear unstable in comparison showing spikes throughout, this is due to the mutation of the particles, briefly causing this jump but ends up finding a new loss leader later on. Evidence of this can be seen in the minimum loss trends of the two where small dips in loss can be seen for SSPSO when mutation kicks in. SSPSO also does not appear to converge early on and continues to increase in accuracy throughout the 100 generations, potentially showing that it may increase in accuracy if we allow more than 100 generations.

The final result of each algorithm shows that SGD as a whole is better for most scenarios due to its flexibility along with faster and greater performance relative to its convergence time and accuracy compared to population-based algorithms. The important catch is that gradient based algorithms do not work when the objective is not a continuous function and is therefore not differentiable, which causes backpropagation to not work. Additionally, when using more complicated networks, the issue of gradient explosion and vanishing may occur which hinders performance. In these scenarios, SL-PSO can solve the problem at the cost of convergence speed and importantly, accuracy [12]. SS-PSO takes this one step further and gives a good boost to SL-PSO by further decreasing the loss whenever it can. This decrease in loss may be more applicable to different landscapes compared to others as seen in our charts. SS-PSO shows a good proof of concept in escaping local minima when presented with a rough landscape and a discontinuous objective function. SS-PSO can also run for a larger number of generations before reaching a peak in comparison to SL-PSO.

Previous papers have explored the idea of particle swarm optimisation and attempting to apply several methods to potentially increase the probability of getting the particles out of a minimum and continue the search towards the global minimum. Papers have previously attempted to dynamically change the weight inertia of the update equation for each particle [27], we take inspiration from this by changing the probability of direct mutation based on the Euclidean distance of each particle to the best. Papers also delve into adding the mutation aspect into a canonical PSO while we attempt to do such a thing on social learning based PSOs which has not been seen before. This gives us the benefits of efficiently working with large dimensionality problems along with solving premature convergence issues in the same optimiser without losing a great amount of performance.

While Social Learning by itself is sufficient to solve the problem and achieve a high accuracy, after many generations, the particles often fall into a local minimum which then struggles to get out of it with the standard velocity and directional updates. As we believe that the particles may not have explored the entire search space, we promote exploration by increasing the probability accordingly to get out of the local minima and continue exploring more promising directions. Additionally, we also promote exploitation in the sense that if the particles have reached a potential solution, we try to mutate them in the hopes of finding a potentially better solution later.

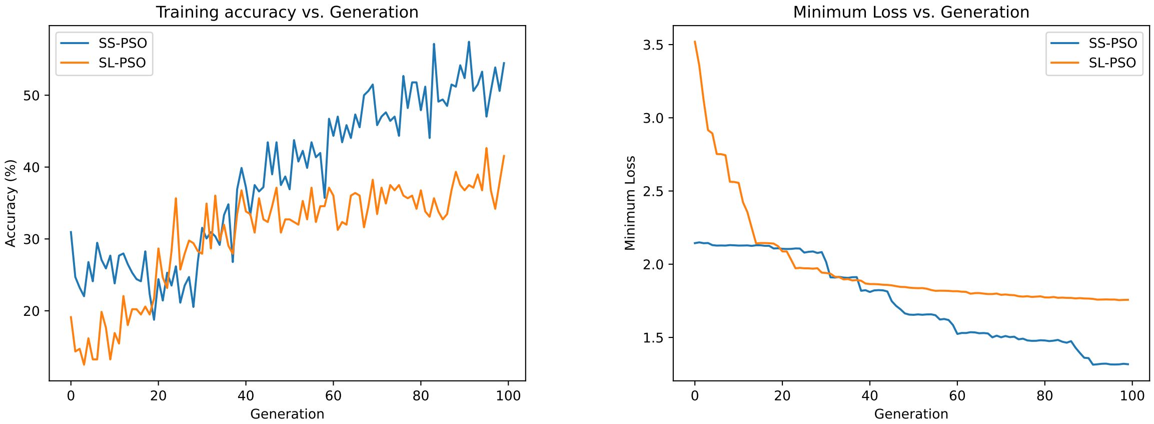

Fig. 4. Placeholder for the accuracy and loss plots

V. Bi-objective problem

Regularisation is introduced into a model to address the problem of overfitting which happens when the model doesn't generalise well to unseen data and has learned the training data too well. Dropout or normalisation layers can be added to the model directly in an attempt to make the model less complex. Instead of doing this we can turn regularisation into a bi-objective function and represent the optimisation problem as a bi-objective problem, where the objectives are to maximise accuracy and minimise model complexity. We use a known optimisation algorithm Fast-Non-Dominated Sorting Algorithm (NSGA-II) to find a set of solutions (Pareto front), in figure 2 that balances between accuracy and a chosen regulariser, sum of weights squared, L2. (Equation 1). The set provides us with solutions that represent a range of trade-offs between the conflicting objectives, accuracy and complexity.

$$\lambda \sum_{i=1}^{k} w_i^2 \tag{2}$$

The crowding distance of the solutions within each front are calculated so that the algorithm can sort leading with the best solution. Dominance rank as well as crowding distance are considered to sort the solutions, the best solutions have larger crowding distances, to maximise the diversity, and lower dominance ranks. Then the algorithm can select using tournament selection the best solutions from the pareto front for the next population.

The set up of our bi-objective optimisation experiment is detailed in table x. We use a population size and iteration size of 100. The dimensions of the individuals are 530 and we define a crossover probability 0.9. We use 1 and -1 as upper and lower bounds an Tournament selection based on dominance followed by crowding distance to find the optimal solutions in the pareto front.

Our primary objective is to maximise accuracy in the network. It is clear to see from the results from NSGA II in figure x that by adding a second objective function it does not give as high accuracy as the single objective optimiser SGD. Furthermore, the bi-objective optimiser converges at a much slower rate, similar to the population-based algorithms, compared to the single objective optimiser. This is because the bi-objective task is trying to balance a set of conflicting solutions, and resultingly does not return the accuracy based optimum solution. Therefore, since our main objective is to maximise the accuracy, we can conclude that the single objective optimiser is better suited to our problem and a more optimal set of weights, improving the accuracy, will be obtained using this method. Papers looking at similar problems also back up our claim that a single optimisation is better suited for this problem. Overfitting is important but it can be incorporated into the network more efficiently during training using methods such as dropout or image augmentation. In [27] the authors find single-optimisation methods outperform multi-objective methods in terms of accuracy and convergence speed. Research in [28] also back up our claims that regularisation can be effectively applied to the network during training whilst still achieving high generalisation performance.

TABLE II
Summary of parameters for NSGA-II

Hyperparameters	Values
Iterations	100
Dimensions	530
Population size	100
Encoding	Real coded
Upper bound	1
Lower bound	-1
Crossover Probability	0.9

Add a short paragraph to comment in what scenarios NSGA-II may work well or talk about when focusing on the second objective might be useful (unless you did that already). Talk about why the second objective may not be as necessary for this scenario. Add a short paragraph to comment in what scenarios NSGA-II may work well or talk about when focusing on the second objective might be useful (unless you did that already). Talk about why the second objective may not be as necessary for this scenario.

VI. Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

References

[1] 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.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[4] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2020.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[8] A. Khan, Z. Rauf, A. Sohail, A. R. Khan, H. Asif, A. Asif, and U. Farooq, "A survey of the vision transformers and their CNN-transformer based variants," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2917–2970, Dec. 2023.

[9] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," University of Toronto, Tech. Rep., 2009.

[10] L. Deininger, B. Stimpel, A. Yuce, S. Abbasi-Sureshjani, S. Schönenberger, P. Ocampo, K. Korski, and F. Gaire, "A comparative study between vision transformers and CNNs in digital pathology," 2022.

[11] Y. Bohra, "The Challenge of Vanishing/Exploding Gradients in Deep Neural Networks," https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/, Jun. 2021.

[12] D. Xu, Y. Li, X. Tang, Y. Pang, and Y. Liao, "Adaptive particle swarm optimization with mutation," in *Proceedings of the 30th Chinese Control Conference*, 2011, pp. 2044–2049.

[13] L. Bliek, A. Guijt, R. Karlsson, S. Verwer, and M. de Weerdt, "Benchmarking surrogate-based optimisation algorithms on expensive black-box functions," *Applied Soft Computing*, vol. 147, p. 110744, 2023.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[15] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[16] M. Li, W. Du, and F. Nian, "An Adaptive Particle Swarm Optimization Algorithm Based on Directed Weighted Complex Network," *Mathematical Problems in Engineering*, vol. 2014, p. 434972, Apr. 2014.

[17] A.-C. Mihai and D.-T. Iancu, "Optimizing a convolutional neural network using particle swarm optimization," in *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2022, pp. 1–7.

[18] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[19] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[20] J. Tang and X. Zhao, "Particle swarm optimization with adaptive mutation," in *2009 WASE International Conference on Information Engineering*, vol. 2, 2009, pp. 234–237.

[21] J. Chen, Z. Ren, and X. Fan, "Particle swarm optimization with adaptive mutation and its application research in tuning of PID parameters," in *2006 1st International Symposium on Systems and Control in Aerospace and Astronautics*, 2006, pp. 5 pp.–994.

[22] Z. Assarzadeh and A. R. Naghsh-Nilchi, "Chaotic particle swarm optimization with mutation for classification." *J Med Signals Sens*, vol. 5, no. 1, pp. 12–20, 2015 Jan-Mar.

[23] N. Li, Y.-Q. Qin, D.-B. Sun, and T. Zou, "Particle swarm optimization with mutation operator," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, vol. 4, 2004, pp. 2251–2256 vol.4.

[24] V. Steffen, "Particle swarm optimization with a simplex strategy to avoid getting stuck on local optimum," *AI, Computer Science and Robotics Technology*, Oct. 2022.

[25] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences*, vol. 291, pp. 43–60, 2015.

[26] L. Gu, Y. Liu, and J. Zhen, "Optimization of PSO algorithm based on adaptive inertia weight and escape strategy," *Journal of Physics: Conference Series*, vol. 1486, no. 3, p. 032033, 2020.

[27] T. G. Tan, J. Teo, and K. O. Chin, "Single- versus Multiobjective Optimization for Evolution of Neural Controllers in Ms. Pac-Man," *International Journal of Computer Games Technology*, vol. 2013, p. 170914, Feb. 2013.

[28] Y. Tian and Y. Zhang, "A comprehensive survey on regularization strategies in machine learning," *Information Fusion*, vol. 80, pp. 146–166, 2022.

Appendix

A. Contributions

Each member contributed equally to all project components.