Open-Source Neural Architecture Search with Ensemble and Pre-trained Networks

Séamus Lankford

***Abstract*—The training and optimization of neural networks, using pre-trained, super learner and ensemble approaches is explored. Neural networks, and in particular Convolutional Neural Networks (CNNs), are often optimized using default parameters. Neural Architecture Search (NAS) enables multiple architectures to be evaluated prior to selection of the optimal architecture. Our contribution is to develop, and make available to the community, a system that integrates open source tools for the neural architecture search (OpenNAS) of image classification models. OpenNAS takes any dataset of grayscale, or RGB images, and generates the optimal CNN architecture. Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and pre-trained models serve as base learners for ensembles. Meta learner algorithms are subsequently applied to these base learners and the ensemble performance on image classification problems is evaluated. Our results show that a stacked generalization ensemble of heterogeneous models is the most effective approach to image classification within OpenNAS.**

***Index Terms*—AutoML, transfer learning, pre-trained models, ensemble, stacking, super learner, PSO, ACO, CNN.**

I. Introduction

Open-source AutoML [1] solutions such as Auto-WEKA [2] and TPOT [3] focus on creating simpler neural architectures. Libraries capable of generating more complex CNN architectures are also available [4]. In addition to open-source options, many large corporations have developed powerful online platforms to enable the generation of neural architectures automatically. Chief among these solutions is Google's Cloud AutoML and Microsoft Azure's AutoML. However, the alternative of using commercial platforms is expensive leaving users with few practical or viable options.

The development of an open-source NAS tool, OpenNAS[1] [5] seeks to address these shortcomings by integrating multiple open-source NAS approaches. With OpenNAS, CNN architectures for grayscale and RGB image datasets are found through the Swarm Intelligence (SI) heuristics of Particle Swarm optimization (PSO) [6] and Ant Colony optimization (ACO) [7]. Pre-trained models using VGG16, VGG19 [8], ResNet50 [9] and MobileNet [10] architectures were fine-tuned and used as feature extractors. Finally, models derived using SI and pre-trained approaches were combined into network ensembles and evaluated.

Manuscript received January 8, 2021; revised March 13, 2021. This work was supported by the ADAPT Centre, which is funded under the SFI Research Centres Programme (Grant 13/RC/2016) and is co-funded by the European Regional Development Fund.

S. Lankford is with the Adapt Centre, Dublin City University, Ireland (e-mail: seamus.lankford@adaptcentre.ie).

[1] https://github.com/seamusl/OpenNAS-v1

II. Background

A. Convolutional Neural Networks

Initially proposed by LeCun [11], CNNs are feed-forward Deep Neural Networks (DNNs) used for image recognition. In this study, SI and ensemble approaches are used to find better combinations of convolutional, pooling and fully connected layers for CNN architectures.

B. Neural Architecture Search

The process of automatically finding and tuning DNNs is referred to as Neural Architecture Search (NAS). Systems implementing NAS typically consist of a search space, a search algorithm and an evaluation strategy. The architectures to be evaluated are set out in the search space, the search algorithm determines how the search space is to be explored and the evaluation strategy determines the best architectures on unseen data. Brute force training and evaluation of all possible model combinations is a crude approach to NAS whereas an improvement is to use SI heuristics. Ensembles, combining multiple models, is an alternative which frequently generates better results.

C. Transfer Learning

Transfer learning is used widely in the deep learning domains of computer vision and natural language processing [12], [13]. The approach involves taking a model developed for one task and applying it as the starting point for a new model which carries out different tasks.

Training networks on large datasets, such as ImageNet, can take days of GPU time. Through the use of transfer learning, features learnt during this process and the underlying model architecture, can be rapidly transferred to a new model domain. Fortunately, many large corporations and research institutions, have made such pre-trained models publicly available.

As part of this research, several pre-trained models are set as the starting point for new models which are fine-tuned and evaluated. These networks included shallow networks such as 16 layer and 19 layer VGG networks [8], a more complex 50 layer ResNet [9] and a 28 layer MobileNet [10]. Original papers for each network type were studied to find the classification error rates on benchmark datasets.

D. Swarm Intelligence

Swarm Intelligence (SI) is an important category of heuristics within the domain of Evolutionary Computing. While many SI algorithms exist, the most prominent are

DOI: 10.7763/IJMO.2021.V11.774

Particle Swarm Optimization (PSO) [14] and Ant Colony Optimization (ACO) [15]. Open-source libraries can facilitate SI implementation.

Using a PSO algorithm, an open-source python library for CNN optimization, openCNN, was developed by Fernandes et al [16]. An alternative ACO based approach, known as DeepSwarm, was developed by Byla and Pang [17]. Both libraries have been demonstrated to offer competitive performance in the classification of CIFAR-10 [18] and Fashion_Mnist data [19].

E. Ensemble Techniques

Cheng Ju *et al*. [20] explored the available options when designing an ensemble for image classification. A detailed analysis was conducted which encompassed the following ensemble techniques: unweighted average, majority voting, Bayes optimal classifier, stacked generalization and a super learner: a cross-validation based stacking method. In their study, the super learner proved the most accurate across all methods.

The super learner approach is an extension of stacking in that it creates an ensemble based on cross-validation. A weighted combination of many candidate learners, developed using different algorithms, are combined to build the super learner [21].

The effects of an ensemble of DNN acoustic models in automatic speech recognition is investigated by Geoffrey Hinton at al [22]. The results clearly show the value of the ensemble approach. Using the same architecture and training methods as the baseline, 10 separate models were trained. Sufficient diversity was introduced through randomly initializing models with different initial parameter values. Such a simple approach allowed averaged predictions of the ensemble to significantly outperform individual models.

III. Proposed Approach

The number of hidden layers, the number of neurons per layer, the type of activation function and the choice of optimizer are among the parameters which need to be optimized as part of a neural architecture search. NAS implementation can be achieved through a variety of approaches including transfer learning using pre-trained networks, network morphism or swarm intelligence. Furthermore, the performance of NAS derived networks can often be enhanced through the use of ensembles.

A. Pre-trained Networks

The internal architecture of VGG16 is illustrated in Fig. 1. As with all CNNs, the architecture is subdivided into a series of blocks which are separated by pooling layers. These blocks may be composed of either convolutional layers or fully connected layers.

Each convolutional layer has set of kernels (i.e. filters) with learnable parameters. The filter size, in both VGG16 and VGG19, is set to 3x3 pixels whereas the number of filters used varies between different blocks. With VGG architectures, the number of filters increases from 64 to 512 as an image progresses through the layers. The filter size and number of filters used are shown in Fig. 1. The effectiveness of pre-trained VGG models, both as feature extractors and fine-tuned models, in generating optimal architectures is explored as part of the approach taken in this study.

Fig. 1. Architecture of VGG16.

In the context of this study, ResNet50 was also used which is a 50 layer ResNet implementation. ResNet shares many of the same characteristics of the VGG networks i.e. blocks of convolutional layers followed by a fully connected layer and SoftMax activation. However, Resnet differs from other architectures in that it uses a principle known as skip connections which reduces the problem of vanishing gradients associated with deeper networks.

MobileNet, a family of computer vision neural networks designed by Google, was also evaluated. Its shallow architecture and fast performance allows for its use in mobile devices. The structure is broadly similar to the architectures of VGG and ResNet with convolutional layers feeding into a fully connected layer that uses SoftMax classification.

Overall these networks can be viewed as performing two clear functions: feature extraction carried out by convolutional layers and classification which is implemented by the fully connected layers.

Convolutional layers are used for feature extraction since they concentrate on smaller regions of the image using multiple small filters (e.g. 3x3 in the case of VGG). This eliminates the need for feature engineering or extraction, such as PCA, which is needed with other forms of artificial neural networks.

With all approaches, the image is classified into various classes using a fully connected (FC) neural network, following feature extraction of the earlier convolutional layers. The last layer in all architectures is the SoftMax layer which converts the output of the previous layer into a probability distribution which can be used for classification. In the context of DNNs, there are two principal modes of transfer learning namely feature extraction and fine-tuning. These common features, identified above, enabled two types of transfer learning to be incorporated in the approach taken in the development of OpenNAS.

1) Feature extraction on pre-trained networks

Transfer learning was performed on the CIFAR-10 and Fashion_Mnist datasets using filters, learned by state-of-the-art networks. These pre-trained networks were initially developed through training on large datasets such as ImageNet. In this manner, transfer learning enabled the pre-trained networks to classify images it was not trained on. With feature extraction, pre-trained networks are treated as feature extractors. Input images propagate through the network and stop at a pre-specified layer. Outputs of this layer are then treated as the features which is illustrated in Fig. 3.

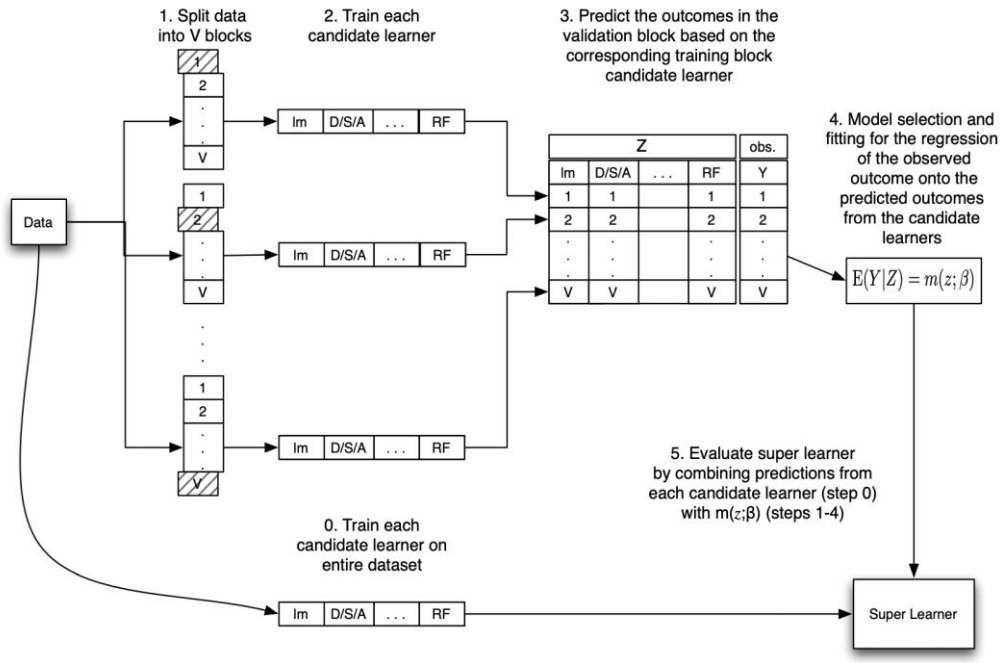

Fig. 2. Super learner approach [21].

Fig. 3. Feature extraction.

Features, which are the output of the max-pooling layer, were flattened into a feature vector. Given a dataset of N images, the process of feature extraction was repeated for all images in the dataset, resulting in a total of N feature vectors. These features, were then trained on scikit-learn machine learning models.

2) Fine tuning of pre-trained networks

With the implementation of fine tuning, hybrid model architectures were created by removing the fully connected layers from the top of the model.

In the OpenNAS design, two blocks were added each of which had a fully connected layer, a batch normalisation layer and a dropout layer. The new hybrid structure was then trained. The inner layers of the model were unfrozen allowing both the convolutional and fully connected layers to be trained for a specific number of epochs. The resulting fine-tuned model architecture is illustrated in Fig. 4.

Fig. 4. Fine-tuning.

B. Ensembles

Ensembles were developed using stacked outputs from base learners. Subsequently, meta learners generated new models by using the stacked ensemble outputs to learn from the base learners. Meta learners using several different algorithms were evaluated. These algorithms include K Nearest Neighbor (KNN) [23], Support Vector Clustering (SVC) [24], Random Forest [25], Logistic Regression [26] and Multi-Layer Perceptron (MLP) [27]. Combinations of homogeneous or heterogeneous base learners were included in creating the network ensembles.

The approach taken in this paper is to focus on stacking ensembles, scikit-learn ensembles and super learner ensembles. With stacking, the accuracy of predictions was improved by combining multiple weaker base learner models. Outputs of N weak learners were combined to form the feature set for a meta learner. Subsequently, the meta learner learns from the prediction outputs of each base learner.

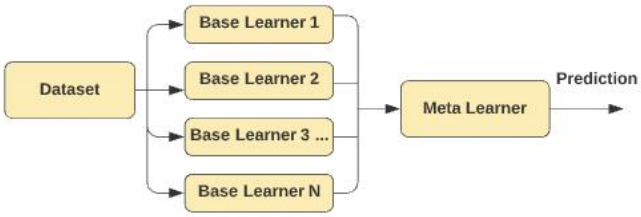

Fig. 5. Stacking approach.

The single level stacking model is further developed with a multi stacked ensemble. With a multi stacked approach, the meta learner is replaced by another set of base learners increasing the model complexity.

The super learner approach is an extension of stacking to k-fold cross-validation whereby all models use the same k-fold splits of the data. The meta-model is fit on the out-of-fold predictions from each model. The steps involved in the super learner approach are outlined in Fig. 2 from Hubbard's original paper [21].

With OpenNAS, other options for neural architecture search are also available. The system allows AutoKeras and auto-model approaches to be used in the search process. Swarm intelligence algorithms may also be selected for optimization. The swarm optimization techniques currently used are Particle Swarm Optimization and Ant Colony Optimization.

2) Stacking with neural networks

As outlined in the system design, ensemble outputs are used to create a stacked training dataset for a meta learner. The meta learner is trained by firstly preparing the training dataset and then using the prepared dataset to fit a meta-learner model. In this manner, features of the meta learner dataset are created using predictions from the base learners.

The stacking ensemble approach adopted by OpenNAS enables both heterogeneous and homogeneous ensembles of base learners models to be evaluated. To develop meta learners, the meta-algorithms chosen as the secondary machine learning classifier included Random Forest, Logistic Regression, KNN, MLP and SVC classifiers. With this implementation, there are two principle modes of operation. The first mode involves the creation of baser learners. These learners are then used to create ensemble outputs to train meta-learners. The second mode of operation simply loads previously built base learners to create the ensemble for the meta-learners.

3) Stacking with scikit-learn

With scikit-learn, ensemble stacking is achieved using the Stacking Classifier library. For the purposes of this study, two types of ensembles were implemented: a one layer stacking ensemble and a multi stacked ensemble consisting of two layers. With the one layer model, illustrated in Fig. 6, two MLP classifiers with different learning rates were used as the base models. The outputs from these learners feed into a Random Forest which is used as the meta learner. A single Stacking Classifier is required.

Fig. 6. Stacking with a single layer.

As illustrated in Fig. 7, the multi stacked implementation consists of two layers of estimators. Layers of estimators are joined using separate Stacking Classifiers. The first layer consisted of a Random Forest, a KNN and 2 MLP classifiers (again with different learning rates). The outputs, i.e. predictions from layer 1 are passed to a layer consisting of a Decision Tree and a Random Forest. Layer 2 outputs are then combined with an SVC classifier to make the final prediction.

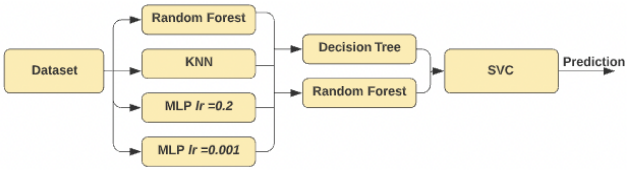

Fig. 7. Multi stacking.

4) Stacking with a super learner

Using the python ML-Ensemble [28] library, a super learner was created. The configuration of base learners used algorithms from Logistic Regression, SVC, KNN, Bagging, Random Forest and Extra Trees. The approach is illustrated in Fig. 8 and the meta model was implemented using a Random Forest algorithm.

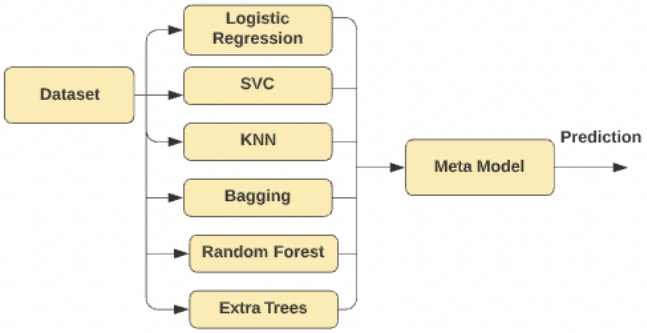

Fig. 8. Stacking with a super learner.

IV. Design

A high level system architecture overview is presented in Fig. 9. The system is organized into the following modules: OpenNAS, pre-processor, trainer, ensemble and loader. Transfer learning, as either a feature extractor or to fine-tune the pre-trained networks, is incorporated in the pre-train function.

Metaheuristics of Particle Swarm Optimization and Ant Colony Optimization are used to search for the optimal neural architecture as part of the SI design. Particle swarms were created using a psoCNN library [16] and ant colonies were implemented using the DeepSwarm library [17]. Existing AutoML tools, such as AutoKeras [4], were also integrated into the OpenNAS system.

With the ensemble module, there are options to build custom stacked ensembles using either homogeneous or heterogeneous base learners. In addition, there are options to create ensembles using either scikit-learn stacking or a super learner. Base learner outputs are subsequently passed to a suite of meta learner algorithms.

The system outputs include the generation of optimal neural architecture models and their associated architecture diagrams.

V. Empirical Evaluation

A. Experimental Setup

Two datasets were chosen for the experimental design, namely CIFAR-10 [18] and Fashion_Mnist [19]. A primary research objective is the development of a Neural Architecture Search tool which chooses the optimal architecture for generic datasets of either grayscale (one channel) or colour (three channel) images. The CIFAR-10 dataset meets this requirement in that it is a challenging dataset of colour images. The Fashion_Mnist dataset is also suitable since it is a well-tested and well understood dataset of black and white images.

For reference, the state of the art (SOA) accuracy achieved on CIFAR-10 is 98.5% whereas with Fashion_Mnist, the SOA accuracy is 94.6% [29].

Full models were developed using a lab of machines each of which has an AMD Ryzen 7 2700X processor, 16 GB memory, a 256 SSD and an NVIDIA GeForce GTX 1080 Ti.

1) CIFAR-10

CIFAR-10 is a dataset of 60,000 32x32 colour images in 10 classes. There are 6,000 images per class creating a

well-balanced dataset. Furthermore, the dataset is divided into five training batches and one test batch, each with 10,000 images. Therefore, there are 50,000 training images and 10,000 test images. The test batch contains exactly 1,000 randomly-selected images from each class. Training batches contain the remaining images in random order and contain exactly 5,000 images from each of 10 classes. CIFAR-10 includes the following image categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

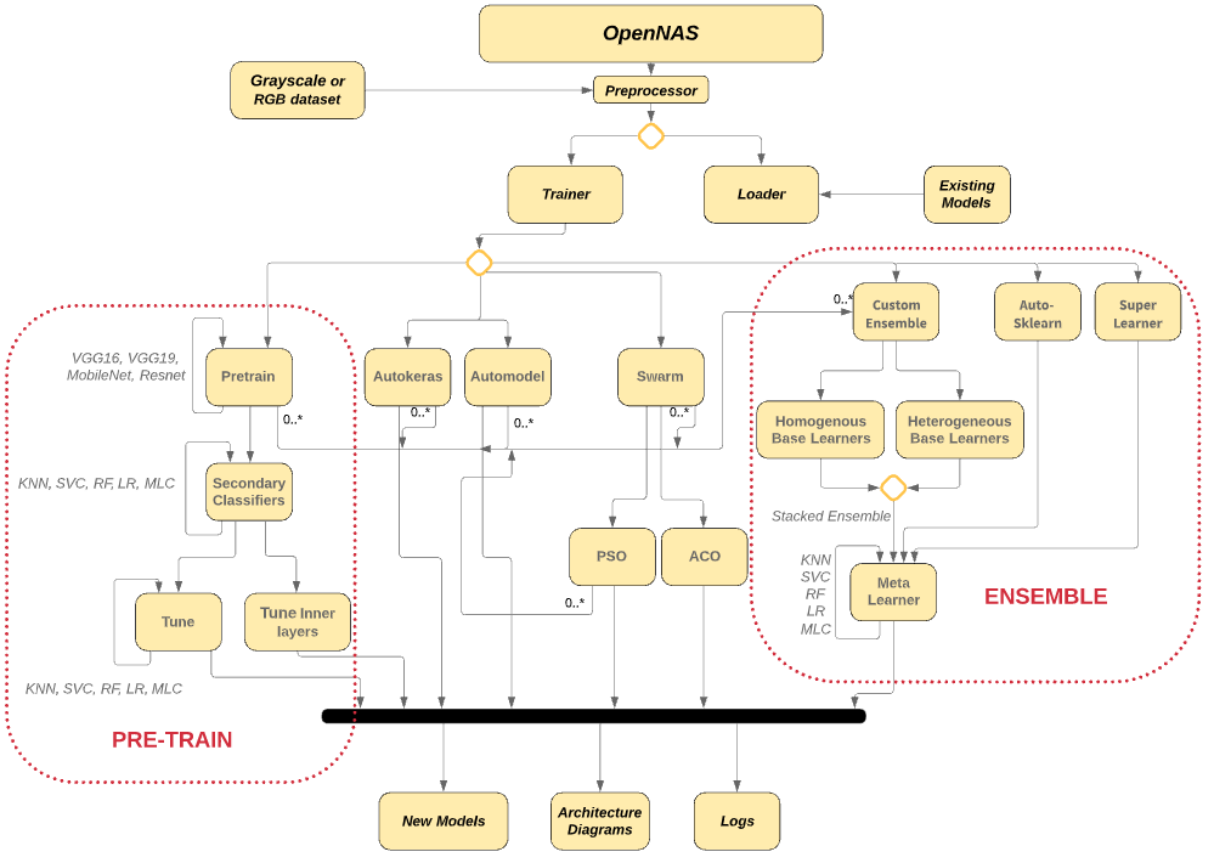

Fig. 9. OpenNAS system design.

2) Fashion_Mnist

Fashion_MNIST is a dataset of grayscale images consisting of a training set with 60,000 examples and a test set of 10,000 examples. Each sample is a 28x28 grayscale image, associated with a label from 10 classes. Fashion item images are labelled according to the following classes: T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag and Ankle boot.

B. Pre-trained Models

Using fine-tuning of pre-trained networks, models were developed with 5 and 10 epochs of training over five independent training runs. The pre-trained networks of VGG16, VGG19, MobileNet and ResNet50 were evaluated. Models were developed using both CIFAR-10 and Fashion_Mnist datasets. A summary of the test results, and their evaluation is presented in the tables and discussion below.

1) Models trained on CIFAR-10 dataset

TABLE I: PRE-TRAINED TEST RESULTS ON CIFAR-10 (5 EPOCHS)

Model	Acc (Mean)	Acc (Max)	Acc (StDev)	Run (s)	Layers (Base)
VGG16	0.888	0.891	0.003	2533	16
VGG19	0.885	0.889	0.003	2931	19
MobileNet	0.813	0.825	0.007	2270	28
ResNet50	0.815	0.823	0.006	3331	50

The key findings of the experimental results for pre-trained models over 5 epochs, illustrated in Table I, indicate the most accurate pre-trained model for the CIFAR-10 dataset used VGG16 as the base model (88.8%). In terms of model accuracy there is little difference between VGG16 (Acc: 88.8%) and VGG19 (Acc: 88.5%). The VGG16 and VGG19 models both perform well.

The consistency of results between VGG16 and VGG19 is not surprising given that the models are the same apart from an extra layer in each of the last two convolutional blocks in VGG19. MobileNet (Acc: 81.3%) and ResNet50 (Acc: 81.5%) lagged significantly in terms of accuracy.

TABLE II: PRE-TRAINED TEST RESULTS ON CIFAR-10 (10 EPOCHS)

Model	Acc (Mean)	Acc (Max)	Acc (StDev)	Run (s)	Layers (Base)
VGG16	0.883	0.892	0.006	5024	16
VGG19	0.890	0.893	0.003	5837	19
MobileNet	0.814	0.821	0.006	4470	28
ResNet50	0.807	0.812	0.007	6583	50

Model training times over 10 epochs, outlined in Table II, are approximately double those of models trained for 5 epochs, as one would expect. Using a greater number of epochs, with its associated longer training times, does not lead to better model accuracy. In fact, accuracy for all models trained over 10 epochs was nearly identical to that achieved over 5 epochs of training. Pre-trained models were

effectively converging at lower levels of training.

Similar to the 5 epoch trained models, the standard deviations on accuracies were low indicating that consistent results can be achieved without significant outliers. The findings indicate that training over 5 epochs creates better performing pre-trained hybrids. It was therefore decided to maintain a cycle of 5 epochs of training for the pre-trained hybrid models.

Training of all pre-trained hybrid models led to overfitting. All pre-trained models show overfitting at early stages. The rapid model convergence can be attributed, in some part, to the use of an image generator.

2) Models Trained on Fashion_Mnist Dataset

TABLE III: PRE-TRAINED TEST RESULTS ON FASHION_MNIST

Model	Acc (Mean)	Acc (Max)	Acc (StDev)	Run (s)	Layers (Base)
VGG16	0.932	0.936	0.003	3033	16
VGG19	0.933	0.936	0.001	3533	19
MobileNet	0.910	0.913	0.004	2706	28
ResNet50	0.913	0.918	0.005	3917	50

From Table III, an analysis of models trained using Fashion_Mnist data clearly shows that higher accuracies can be achieved using a simpler dataset. Mean model accuracy varied between 91.0% and 93.3% for Fashion_Mnist relative to a range of 80.7% to 89.0% for CIFAR-10.

The findings of test runs carried out on the one channel dataset of Fashion_Mnist are consistent with what was observed with CIFAR-10. There are 2 clear groups namely the higher performing VGG16 and VGG19 set compared with the lower performance of MobileNet and ResNet50. The difference at just 2% is much less marked than is the case for CIFAR-10 models where the mean accuracy differential is 8%. It can concluded that the shallower models of VGG16/VGG19 again perform better than the deeper models of MobileNet and ResNet50.

The average run time for classifying Fashion_Mnist data was notably faster than CIFAR-10 (approximately 40% faster) across all model types. Again, this is line with expectations given that CIFAR-10 is a more challenging dataset. As expected, MobileNet, designed as a light weight model, was the fastest of all model types for both Fashion_Mnist and CIFAR-10.

Training the hybrid models on Fashion_Mnist showed similar characteristics to CIFAR-10 training. A high degree of overfitting occurred which again illustrates rapid accuracy convergence for pre-trained models on both triple channel and single channel datasets.

3) Summary of pre-trained evaluation

Several key observations can be made when assessing pre-trained model performance on CIFAR-10 and Fashion_Mnist. It can be seen, from Tables I-III, that deeper models are not necessarily more accurate than shallower networks. In all cases, it was observed that pre-trained hybrid models converge rapidly. This is not surprising given that such models have well developed weights from extensive prior training on large datasets such as ImageNet.

The relative performance of pre-trained models with regard to validation accuracy is also highlighted in Table I and Table II. Of the four models evaluated, ResNet50 and MobileNet perform worse on both Fashion_Mnist and CIFAR-10 relative to the VGG architectures. Given that MobileNet is primarily designed for lightweight mobile applications, this finding is not surprising.

C. Stacking with Neural Networks

Using CIFAR-10 and Fashion_Mnist datasets, stacking ensembles were evaluated using Random Forest, KNN, MLPC, SVC and Logistic Regression as meta learners. Both homogeneous and heterogeneous stacking ensembles were created.

The homogeneous ensembles used in this study simply consisted of two members. ACO ensembles consisted of two members whose architecture was derived from an ACO search whereas the PSO ensembles were developed using a PSO heuristic.

In addition, the performance of heterogeneous ensembles was also explored. The ensemble, Hetero-4 comprised of four models using two VGG16 and two VGG19 models. The Swarm ensemble was also a four model ensemble consisting of two ACO trained models and two PSO trained models.

1) Ensemble performance on CIFAR-10

The relative performance of all meta learners in classifying CIFAR-10 data is clearly illustrated in Fig. 10. Two clear groups are identified in the relative performance of all meta learners in classifying CIFAR-10 data. The higher performing group of the Random Forest and KNN classifiers stand out in comparison to the poorer performing group consisting of the MLPC, SVC and Logistic Regression algorithms. For lower performing ensemble groups, the differential is substantial. In the case of the VGG16 ensemble, there is a difference of 3.7% in accuracy achieved between using a Random Forest and an SVC approach. The accuracies achieved by higher performing meta learners across all ensemble types are summarized in Table IV and in Table V.

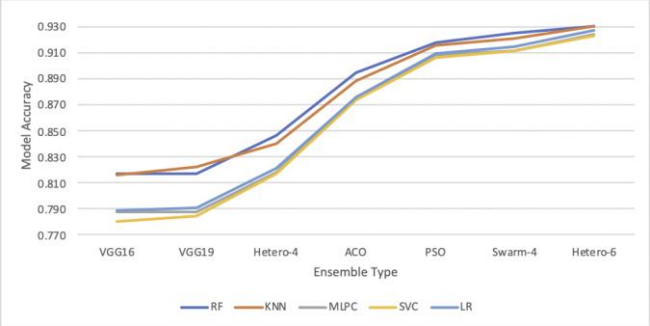

Fig. 10. Relative mean performance of ensembles on CIFAR-10.

Ensembles consisting of weaker members performed worse than ensembles with higher performing members. This is evident from Table IV where the Hetero-4 ensemble achieves 84.7% using Random Forest whereas the Swarm ensemble came in at 92.5%.

With this study, the impact of the number and diversity of models, on overall ensemble accuracy can be seen. Increasing the number of models within an ensemble often increases ensemble accuracy. The Hetero-6 ensemble performed significantly better (93.1%) compared with its Hetero-4 counterpart (84.7%) using a Random Forest meta learner. Heterogeneous ensembles, containing diverse models, were seen to offer better performance compared to their homogeneous counterparts.

TABLE IV: ENSEMBLE MEAN PERFORMANCE ON CIFAR-10

Model	RF	KNN	Best Member	Delta	Run Time (s)
Hetero-6	0.931	0.930	0.900	3.07%	341
Swarm	0.925	0.921	0.900	2.46%	254
PSO	0.918	0.916	0.900	1.80%	198
ACO	0.895	0.889	0.848	4.67%	76
Hetero-4	0.847	0.841	0.755	9.22%	170
VGG19	0.818	0.822	0.755	6.31%	86
VGG16	0.817	0.816	0.743	7.43%	76

2) Ensemble performance on fashion_mnist

The relative performance of meta learners in classifying Fashion_Mnist data, using different types of ensembles, was investigated. The findings, illustrated in Fig. 11, are consistent with previous observations. Similar to the CIFAR-10 evaluations, there were two distinct groups of meta learners namely the higher performing set of Random Forest and KNN compared with the weaker performance of MLPC, SVC and LR. For all ensemble types, Random Forest was again the strongest performer of all meta learners.

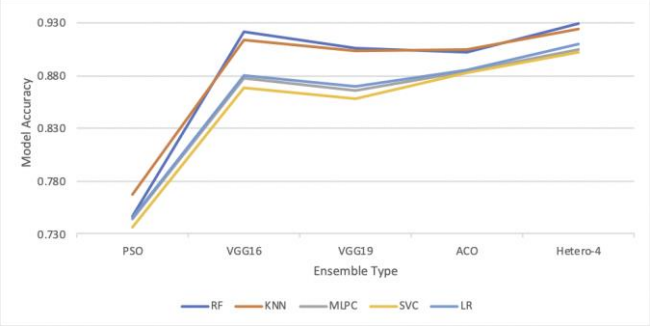

Fig. 11. Relative mean performance of ensembles on Fashion_Mnist.

Consistent with a previous observation, it can be seen that the ensemble set with the highest number of members offers the greatest performance. The Hetero-4 ensemble, with 4 members, achieves an accuracy of 93% compared with a slightly lower accuracy of 92.2% on the VGG16 ensemble of two members.

TABLE V: ENSEMBLE MEAN PERFORMANCE ON FASHION_MNIST

Model	RF	KNN	Best Member	Delta	Run Time (s)
Hetero-4	0.930	0.924	0.831	9.89%	156
VGG16	0.922	0.913	0.807	11.43%	75
VGG19	0.906	0.903	0.831	7.52%	83
ACO	0.902	0.904	0.867	3.44%	33
PSO	0.747	0.768	0.686	6.09%	93

D. Scikit-Learn Stacking and Super Learner

For the purposes of this study, the effectiveness of scikit-learn in classifying CIFAR-10 and Fashion_Mnist data was evaluated. Scikit-learn stacking was compared with a super learner approach, which is a stacking ensemble variation incorporating cross fold validation.

1) Scikit-learn and super learner on CIFAR-10

On first inspection of CIFAR-10 classification in Table VI, the accuracy results for both the super learner (49%) and scikit-learn (52%) appear poor. The performance of these approaches is governed by the algorithms chosen for the base learners and meta learners. Several variations of base learner and meta algorithms were tested. Variations included increasing the number, and diversity, of base learners. A multi stacked approach, with 2 layers, was also implemented. The accuracy of all OpenNAS approaches showed little deviation and stayed within a range of 49% to 53%. Experiments conducted, as part of the original Auto-Sklearn paper indicate a baseline accuracy of 51.7% on CIFAR-10 demonstrating a consistency with the results observed as part of this study [30], [31].

TABLE VI: SCIKIT-LEARN STACKING AND SUPER LEARNER PERFOMANCE ON CIFAR-10

	Accuracy (Mean)	Runtime (s)
1 Layer	0.524	8852
2 Layers	0.520	11910
Super Learner	0.490	5507

2) Scikit-learn and super learner on fashion_mnist

By comparison with the findings for CIFAR-10, the accuracies obtained on Fashion_Mnist when using either the scikit-learn or the super learner approach are much improved. As previously noted, Fashion_Mnist is a less challenging dataset given that it contains grayscale images.

The accuracies achieved, and their associated run times, are illustrated in Table VII. In comparing approaches, the super learner approach offered better performance in both its accuracy (88.7%) and run time of 2144 seconds.

TABLE VII: SCIKIT-LEARN AND SUPER LEARNER PERFORMANCE ON FASHION_MNIST

	Accuracy (Mean)	Runtime (s)
Super Learner	0.887	2144
2 Layers	0.877	3366
1 Layer	0.869	2418

VI. DISCUSSION

For the CIFAR-10 dataset, the highest performing model is a heterogeneous ensemble of six base models feeding into a Random Forest meta learner, which in itself is an ensemble. Effectively this creates an ensemble of ensembles to ensure an accuracy of 93.1%. The base models consisted of two ACO derived models, two PSO derived models, two pre-trained models (i.e. one VGG16 and one VGG19 model). The accuracy achieved by the ensemble is significantly higher (3.1%) when compared with the best performing models in previous OpenNAS studies [5]. In comparison to approaches, which rely solely on SI heuristics [17], the difference is even greater (4.41%).

The pre-trained networks of MobileNet and RestNet50 delivered the poorest performance with CIFAR-10. The other pre-trained networks, using VGG architectures, performed very well on the same dataset. However the accuracy of the VGG ensembles was still 4% lower than the highest ranking ensemble, Hetero-6.

Many of the characteristics exhibited with CIFAR-10 were also seen in Fashion_Mnist classification. The lowest performing models were again the pre-train set of Resnet50 and MobileNet. The VGG16 and VGG19 models both performed well on Fashion_Mnist. In relation to the best performing ensemble, it is worth noting the highest performing base learner had an accuracy of 83.1% but still managed to deliver overall accuracy of 93%. A marginally

higher ensemble accuracy should be possible if better performing base models were included. It was shown that a mean accuracy of 93.3% on Fashion_Mnist could be achieved using a VGG19 model.

Scikit-learn stacking and super learner approaches performed poorly on CIFAR-10. Clearly, they are not suited to the classification of complex triple channel image datasets, which demand a convolutional neural network approach to achieve accuracies greater than 90%. Run times associated with various ensemble types are illustrated in Tables I-IV. The difference in run time between stacking ensembles and that of the scikit-learn or super learner approaches is very significant. In fact, in classifying CIFAR-10, the run time of the slowest stacking ensemble (Hetero-6) is 15 times faster than the quickest of the super learner and scikit-learn approaches. The use of pre-built base models enabled such fast performance from stacking ensembles whereas the super learner and scikit learn approaches required new models to be built.

VII. Conclusion

With OpenNAS, a heterogeneous ensemble achieved the highest accuracy in classifying CIFAR-10 data. The accuracy achieved with OpenNAS ensembles is competitive with the current state of the art.

Meta learner algorithms have a significant impact in determining stacking ensemble accuracies. The Random Forest classifier is consistently the best meta learner, irrespective of the underlying ensemble.

The super learner and scikit-learn stacking approaches are fundamentally designed for simpler neural networks using classifier algorithms from the scikit-learn suite. However, they have also been shown to perform well on convolutional neural networks which classify less complex grayscale image datasets such as Fashion_Mnist.

While Keras offers a powerful framework for neural net development, the strengths of the sci-kit learn library should not be overlooked. In particular, in the absence of pre-built base learners, it was shown how an scikit-learn or a super learner approach can be used to quickly develop high performing ensembles for simpler datasets. However, creating a stacking ensemble of pre-built models is significantly faster than building models from scratch using either scikit-learn or a super learner.

It has been found that custom, heterogeneous stacked ensembles of pre-built SI and pre-trained models deliver superior performance for colour image datasets, both in accuracy and run time.

VIII. Future Work

Given the previously discussed modular design, future work could be carried out within each subcomponent. This work could be broadly classified into either further development work or the evaluation of different system configurations.

Development work would focus mainly on more rapid convergence of models during system training, while reducing the amount of overfitting.

Greater use of specific techniques to reduce overfitting should be employed. Many of these techniques are well established and routinely used in deep learning. Hinton et al [32], outline how data augmentation and dropout can be applied to good effect in dealing with overfitting.

While dropout layers were added to the blocks attached to the pre-trained networks investigated in this study, the effect of different dropout rates was not examined. A standard dropout rate of 0.5 was applied. A configurable parameter could easily be created which would enable testing the effectiveness of various dropout rates.

Large weights could be penalised by adding a cost to the network's loss function through the use of L1 and L2 regularisation. Similar to the dropout rate, a configurable parameter could be used for the hybrid and pre-train modes. Future work could also consider the reduction in pre-train network capacity through the removal of specific layers or the reduction in the number of elements of some of the hidden layers.

With regard to system settings, the number of configurations evaluated for the SI components was quite limited due to the long run times associated with model development. As outlined in the design, there are many parameters associated with training models using PSO. The effect of modifying population size and the number of iterations was investigated. However, it would be interesting to study the effects of modifying the parameters for CNN architecture, CNN training and the probability settings.

Exploration of the effects of modifying parameters associated with Ant Colony Optimization would also be worthwhile. The study primarily focused on changing both the number of ants and the number of training epochs. However, increased accuracy could be achieved, and insights gained, from modifying default pheromone settings and providing a greater set of CNN layer parameters to choose from in the configuration files.

Conflict of Interest

The author declares no conflict of interest.

Author Contribution

As the sole author, Séamus Lankford, conducted the whole of this research and wrote the paper.

Acknowledgment

The author wishes to acknowledge the support received by Dr Diarmuid Grimes of the Munster Technological University. Furthermore, the author would also like to thank the School of Computing at Dublin City University (ADAPT centre) for their support.

References

[1] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*, Springer Nature, 2019.

[2] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 826–830, 2017.

[3] R. S. Olson and J. H. Moore, "Tpot: A tree-based pipeline optimization tool for automating," *Automated Machine Learning: Methods, Systems, Challenges*, p. 151, 2019.

[4] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1946–1956.
[5] S. Lankford and D. Grimes, "Neural architecture search using particle swarm and ant colony optimization," in *Proc. 28th AIAI Irish Conference on Artificial Intelligence and Cognitive Science*, 2020.
[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.
[7] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[10] H.-Y. Chen and C.-Y. Su, "An enhanced hybrid mobilenet," in *Proc. 2018 9th International Conference on Awareness Science and Technology (iCAST)*. IEEE, 2018, pp. 308–312.
[11] L. C. Yann, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. the IEEE 86*, no. 11, 1998, pp. 2278-2324.
[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
[13] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1633–1685, 2009.
[14] B. A. Garro and R. A. Vazquez, "Designing artificial neural networks using particle swarm optimization algorithms," *Computational Intelligence and Neuroscience*, 2015.
[15] M. Mavrovouniotis and S. Yang, "Training neural networks with ant colony optimization algorithms for pattern classification," *Soft Computing*, vol. 19, no. 6, pp. 1511–1522, 2015.
[16] F. E. F. Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm and Evolutionary Computation*, vol. 49, pp. 62–74, 2019.
[17] E. Byla and W. Pang, "Deepswarm: Optimising convolutional neural networks using swarm intelligence," in *UK Workshop on Computational Intelligence*, Springer, 2019, pp. 119–130.
[18] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," vol. 55, 2014.
[19] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.
[20] C. Ju, A. Bibaut, and M. van der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *Journal of Applied Statistics*, vol. 45, no. 15, pp. 2800–2818, 2018.
[21] M. J. Van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Statistical Applications in Genetics and Molecular Biology*, vol. 6, no. 1, 2007.
[22] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.
[23] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.
[24] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
[25] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[26] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.
[27] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para*, Cornell Aeronautical Laboratory, 1957.
[28] S. Flennerhag, "mlens documentation," 2017.
[29] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," 2019.
[30] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: The next generation," 2020.
[31] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Auto-sklearn: efficient and robust automated machine learning," *Automated Machine Learning* Cham, 2019, pp. 113–134.6.
[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp.84–90, 2017.

Séamus Lankford is a PhD candidate with the Adapt Centre in Dublin City University. His PhD concentrates on neural machine translation of low resource languages with a specific focus on the Irish language. At the Munster Technological University (MTU), Séamus lectures in the area of Physical Computing and supervises undergraduate projects in machine learning. He is the placement coordinator and industry point of contact for software development students. Having graduated from University College Cork (UCC) with a BE (hons) in electrical engineering, he worked both as an engineer and as a software developer with the ESB, the European Space Agency and Motorola prior to joining MTU. His postgraduate qualifications include an MBA (UCC) and an MSc in artificial intelligence (CIT). His MSc focused on the NAS aspect of AutoML and he developed an open-source neural architecture search tool (OpenNAS) for CNN image classification. His research interests are in the areas of AutoML, machine learning, natural language processing and machine translation.