# Optimising Deep Learning by Hyper-Heuristic Approach for Classifying Good Quality Images

Muneeb ul Hassan[1], Nasser R. Sabar[2][0000−0002−0276−4704], and Andy Song[1]

[1] RMIT University, Melbourne, VIC 3000, Australia
muneeb_hassan@outlook.com,andy.song@rmit.edu.au
[2] La Trobe University, Melbourne, VIC 3083, Australia
n.sabar@latrobe.edu.au

**Abstract.** Deep Convolutional Neural Network (CNN), which is one of the prominent deep learning methods, has shown a remarkable success in a variety of computer vision tasks, especially image classification. However, tuning CNN hyper-parameters requires expert knowledge and a large amount of manual effort of trial and error. In this work, we present the use of CNN on classifying good quality images versus bad quality images without understanding the image content. The well known datasets were used for performance evaluation. More importantly we propose a hyper-heuristic approach for tuning CNN hyper-parameters. The proposed hyper-heuristic encompasses of a high level strategy and various low level heuristics. The high level strategy utilises search performance to determine how to apply low level heuristics to automatically find an appropriate set of CNN hyper-parameters. Our experiments show the effectiveness of this hyper-heuristic approach which can achieve high accuracy even when the training size is significantly reduced and conventional CNNs can no longer perform well. In short the proposed hyper-heuristic approach does enhance CNN deep learning.

**Keywords:** Hyper-Heuristics · Deep Learning · CNN · Optimisation.

## 1 Introduction

Deep learning is a fast growing area in Artificial Intelligence as it has achieved remarkable success in many fields apart from the well publicised Go player - AlphaGo [1]. These fields include real time object detection [2], image classification [3] and video classification [4]. It also performed well in speech recognition [5] and natural language processing [6]. Major deep learning methods are Convolutional Neural Network, Deep Belief Network and Recurrent Neural Network. One of the problems of these deep learning methods is the configuration of the learning process because these learning algorithms are sensitive to parameters and a good performance is often the result of a good parameter combination. However finding a good combination is not a trivial task. For example the parameters in Convolutional Neural Network typically involve batch size, drop out rate, learning rate and training duration. They all can significantly impact the

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

learning performance of deep learning on a particular task. In this study we will address this issue by introducing a hyper-heuristic approach to automatically tune these parameters. The particular problem in this study is image classification. We would like to train a deep network classifier to differentiate good quality images versus bad ones regardless the image content. The problem itself is novel.

Image Classification has been studied of many decades and is one of the key areas in computer vision. The task of image classification is to differentiate between images according to their categories. Image classification usually has a set of targets for example handwritten digits in images[7], human faces appeared on photos [8], various human behaviours captured in video image frames [9] and target objects like cars and books. However, in many real world scenarios, image quality, which is independent of image content, is also of significant importance. It is highly desirable that good photos can be separated from bad photos automatically. Bad images then could be improved or rejected from an image collection so less resources would be consumed. An extension on this is to even automatically select aesthetic images. The aim of this study is the first step, utilising deep learning to differentiate good images from images of obvious poor quality such as blurred images and noisy images. In particular the research goal of this study is to answer the following questions:

1. How to formulate deep learning to differentiate between images of good quality and images of bad quality without understanding the image content?
2. To what extend the training samples can be reduced while still maintaining good accuracy in classifying good vs bad images?
3. How to automatically tune the deep learning parameters to achieve good classification results?

Hence our investigation is also organised in three components. The first part is try to determine a suitable convolutional network structure as a classifier for good and bad images. Secondly, we study the impact of the training size on the classification performance. Thirdly, a hyper-heuristic approach is introduced to evolve the appropriate parameter combinations.

In Section 2 the image datasets are introduced. Section 3 describes the deep learning methodology while Section 4 describes the hyper-heuristic methodology. Section 5 shows the experiments with results. The conclusion is presented in Section 6.

## 2 Image Data Sets

In this study the well know image classification benchmark, the MNIST dataset is used to represent the good images [7]. MNIST is a standardized image collection which consists of handwritten digits from 0 to 9. Each digit is a 28x28 pixel gray scale image. MNIST comes with a training set which consists of 60000 such images of digits and a test set which contains 10000 similar images.

A variation of MNIST dataset which is called noisy MNIST or n-MNIST, is used to represent the bad images [10]. There are three subsets of n-MNIST:

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

1. MNIST with motion blur
2. MNIST with additive white gaussian noise(awgn)
3. MNIST with AWGN and reduced contrast.

These datasets are the exact replicas of original MNIST but with additional noise. Each image in n-MNIST is also 28x28 gray scale. There are 60000 training examples and 10000 test examples. The labels in training and test data-sets are hard encoded, e.g. each label is a $1 \times 10$ vector.

**Fig. 1.** Example of images from MNIST Dataset[11]

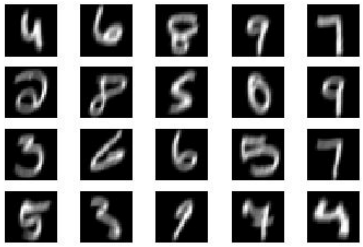

**Fig. 2.** Example of images from Motion Blur Dataset.

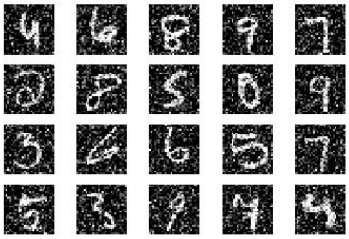

**Fig. 3.** Example of images from AWGN Dataset

**Fig. 4.** Example of images from Additive White AWGN Dataset.

The MNIST with motion blur filter is created by imitating a motion of came by 5 pixels with an angle of 15 degrees which makes the filter a vector for horizontal and vertical motions. The MNIST with AWGN is created by introducing additive white Gaussian noise with signal to noise ratio of 9.5. The MNIST

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

with reduced contrast and AWGN is created by introducing contrast range with AWGN with signal to noise ratio of 12 [10].

## 3 Deep Learning Methodology

In this study, we use the well-know convolutional neural network (CNN) through Keras and Tensorflow. Keras is a high-level neural network API which is built on top of Tensorflow. With keras, we can define models with different standalone configurable modules which then can be combined to form a neural network model. Tensorflow is a deep learning library developed by Google [12]. Tensorflow is a directed graph which consists of nodes and it also maintain and update the state of the node. Every node has zero or more input and zero or more outputs. Value flow among the node to node and values are arbitrary long arrays called tensors. An example of a tensor graph is shown in figure 5. That is a simple equation of cost computed as a function of rectified Linear Unit(ReLu) in which the matrix of weights $W$ and input $x$ are multiplied then adding a bias $b$.

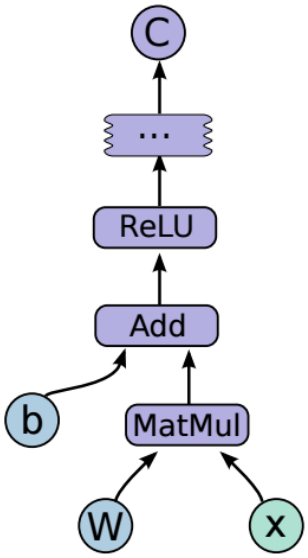


**Fig. 5.** A Tensorflow Computation Graph[13]

For our image classification tasks, we use two 2D convolution layers (convolution2D), with a 2D max pooling layer (MaxPooling) placed after the second convolution layer. The output of MaxPooling is flattened to a one dimensional vector which will be passed through a fully connected dense layer. The dense

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

layer and a drop out layer are introduced after the MaxPooling2D layer to produce better generalization. For all the layers the Rectified Linear Unit (ReLu) activation is used. The output of dense layer uses the softmax activation for probabilistic classification.

The hyper-parameters of the CNN learning are listed below. The optimization target in this study is to find a good combination of these hyper-parameters and ultimately lead to a better accuracy:

1. The batch size - the number of training examples used in one iteration.
2. The number of epochs - representing the number of iteration over the entire data set.
3. The number of neurons in the fully connected layer.
4. Drop out probability
5. The learning rate
6. The Rho factor
7. The epsilon factor

During the learning, we split the training data into a training set and a validation set. Validation data consist of 20% of the original training set and the training set uses the rest. During training, the validation loss is monitored. When it stops decreasing or starts increasing then the training will terminate to avoid over-fitting.

Once the learning is terminated, the trained network will be applied on the test image set to obtain test accuracy. The accuracy in this study is simply classification accuracy which is calculated as

$$Accuracy = \frac{\sum True\ Positive + \sum True\ Negative}{Total\ number\ of\ Images} \tag{1}$$

There are other ways to evaluate the model, for example ROC, F-measure and MSE. Only classification accuracy measure described above is used for simplicity reason. Also our image datasets are quite balanced and true and false cases are equally important. Hence training and test accuracies are sufficient to guide the learning and to indicate the performance of learned models.

Our second aim is to see how training size would impact the learning. It is obvious that the computational cost will be less if the training set is small. However a data set, which is too small, would not be representative enough to enable good learning. Therefore it is important to find the right balance between good performance vs computational cost, especially in real world applications. In this study we try to find minimum size for training which can still lead to reasonable test performance. Logarithmic scale is used here, in the order of $2^n$, $2^{n-1}$, $2^{n-2}$, until $2^3$ and $2^2$.

Note the size reduction only applies on the training data. The test set, which contains 10000 MNIST images (good) and 10000 n-MNIST images (bad), is consistently used in all experiments. Only test accuracy is used to report the learning performance unless specified otherwise.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

## 4 Hyper-heuristic Parameter Optimisation

hyper-heuristics have been proposed for selecting and generating heuristics to solve a particular problem [14]. It has been successful in many different fields [14], [15], [16], [17], [18], [19], [20]. The aim of hyper-heuristic is to find and assemble good optimisation heuristics. Different operations or techniques can be introduced as heuristics so the overall optimisation could be more effective and more efficient.

Hyper-heuristic often begins from randomly generated initial solution and then iteratively improve the solution. A traditional selection hyper-heuristic approach has two key components: low level heuristics and high level strategy. The low level heuristics operate on the solution space. The quality of solution is being evaluated by the objective function from the domain. Whereas high level strategy operate on the heuristic space. It will form heuristics to improve the result and secondly it will also determine whether to accept or reject the generated solution by the acceptance criterion. The components of this framework are briefly described below:

### 4.1 High Level Strategy

The high level strategy uses the past search performance of low level heuristics to decide which heuristic should be applied at each decision point. It selects one from a pool of heuristics in the low level. This work uses the Multi-Armed Bandit(MAB) as an on-line heuristic selector [21], [22]. MAB is based on the record of past performance, e.g. the performance in previous iterations. The record stores an empirical reward and confidence level. The former is the average rewards achieved by that heuristic. The confidence level is the number of times that the heuristic has been selected. The higher values of these two scores indicate better quality of the heuristic [23]. MAB goes through all heuristics one by one and selects the one which returns the maximum value when applied Equation (2).

$$\arg \max_{i=LLH_1 \ldots LLH_n} \left( q_{i(t)} + c\sqrt{\frac{2log \sum_{i=LLH_1}^{LLH_n} n_{i(t)}}{n_{i(t)}}} \right) \tag{2}$$

where $LLH_n$ is the total number of heuristics in the low level, $n_{i(t)}$ is number of times that $i^{th}$ heuristic has been applied up to time $t$ and $q_{i(t)}$ is the empirical reward of the $i^{th}$ heuristics up to time $t$ which is calculated as follows: $q_{i(t)}$=$q_{i(t)}$ + $\Delta$, where $\Delta$ is the difference between the quality of the old and new solutions.

### 4.2 Acceptance Criterion

Acceptance criterion is in the high level and is independent of the domain. Monte Carlo acceptance criterion is used in this study [17]. A solution that improve the

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

objective function will be accepted if the following condition is met [23].

$$R < exp(\Delta f) = exp(f_t - f_{t-1}) \tag{3}$$

where $R$ is the random number between [0,1] and $\Delta f$ is the difference between performance at $(t-1)$th and $(t)$th iterations.

### 4.3 Low level heuristics

In this work, 18 heuristics are included in the low level. Every heuristic has different characteristics hence may lead to different search behaviours. We use the following six heuristics to form the set of low level heuristics. Each heuristic is used in several ways to change one, two, three, real values parameters only, integer parameters only or all parameters.

**Parametrised Gaussian Mutation**

$$X_i = X_i + N(0, \sigma^2) \tag{4}$$

where $\sigma^2$ is 0.5 times the standard deviation [23].

There another three operators which are the same as above but with different $\sigma$ values ranged from 0.2, 0.3 and 0.4 of the standard deviation.

**Differential Mutation**

$$X_i = X_i + F \times (X_{1i} - X_{2i}) \forall i = 1...n \tag{5}$$

where $X_i$ is the decision variable for a given solution and $X_1 i$ is the best solution and $F$ is the scaling factor[23].

**Arithmetic Crossover**

$$X_i = \lambda \times X_i + (1 - \lambda) \times X_{1i}, \forall i = 1...N \tag{6}$$

where $\lambda$ is random number with range 0 to 1. $X_i$ is the current solution and $X_1 i$ is the current best solution [23].

### 4.4 Initial Solution

This in our study is a set of CNN parameters that need to be tuned. These parameters are represented as an array. Each parameter initially is randomly generated. The random function is as follows:

$$x_p = l_p + Rand_p(0, 1) \times (u_p - l_p), p = 1...p \tag{7}$$

where $p$ is the total number of parameters to be tuned. $Rand_p$ returns a random number within 0 and 1. $l_p$ and $u_p$ are lower bound and upper bound respectively for that parameter[23].

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

## 5 Experiments and Results

The first set of experiments are for image classifications. There are two most commonly used optimisers that were studied, namely Adam and Adadelta. In [24], it was mentioned that Adam and Adadelta provide the best convergence during the learning process. Table 1 show the classification performance on noisy MNIST sets with these two optimisers. The learning rate was set as 0.2 for all experiments. This preliminary experiments show that Adadelta can achieve better accuracy in comparison with Adam.

**Table 1.** Experiment with training on MNIST and testing on n-MNIST

| Datasets | Optimiser | Learning Rate | Train Accuracies | Test Accuracies | Epochs |
|---|---|---|---|---|---|
| mnist-m-b | Adam | 0.2 | 0.9828 | 0.9631 | 4 |
| mnist-m-b | Adadelta | 0.2 | 0.9732 | 0.9660 | 4 |
| mnist-awgn | Adam | 0.2 | 0.9810 | 0.7023 | 4 |
| mnist-awgn | Adadelta | 0.2 | 0.9737 | 0.7897 | 4 |
| mnist-rc-awgn | Adam | 0.2 | 0.9814 | 0.5287 | 4 |
| mnist-rc-awgn | Adadelta | 0.2 | 0.9740 | 0.6676 | 4 |

After a range of preliminary experiments, we settled on the settings include the optimisation algorithm, learning rate, drop out rate and number of neurons in the dense layer to start our experiment on classifying the noisy-MNIST and MNIST images. The images for training data are more than 60,000. We decrease the data size by half starting from $2^{16} = 65540$ images to see the impact on test accuracy. For each size we repeat the experiment 30 times. The results are shown in Table 2 including the average training accuracies and test accuracies of the 30 runs. The epochs are all set as 10 to be consistent.

As we can see from Table 2, the classification performance between training on 65540 images and 512 images are not much different, meaning 512 is sufficient for training image classifiers to recognise good quality images. The drop in performance between 512 and 64 images is not major as well. The set of 32 images starts showing significant performance loss indicating more training images are required. When the training size is as small as 4, the test accuracy becomes 50% which is pretty much random guessing for this binary classification task.

The above experiments confirm that the size of training dataset does impact on training. In the next set of experiments the hyper-heuristic approach presented in Section 4 is added in the learning process to tune the network parameters. The results are shown in Table 3 which listed the average test accuracies of 30 runs on training set of size 512 to that of size 4. Sizes above 512 are not included as the results from these sets would be all similar and close to 100%. For comparison purposes, the test results of training without the hyper-heuristic approach from Table 2 are repeated in the middle column of Table 3 .

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

**Table 2.** Experiment with different training sizes

| Training Size | Training Accuracies | Test Accuracies | Epochs |
|---|---|---|---|
| 65540 | 0.9999 | 1.0 | 10 |
| 32770 | 1.0 | 1.0 | 10 |
| 16384 | 0.9999 | 1.0 | 10 |
| 8192 | 0.9998 | 1.0 | 10 |
| 4096 | 0.9979 | 0.9998 | 10 |
| 2048 | 0.9866 | 0.9917 | 10 |
| 1024 | 0.9639 | 0.9922 | 10 |
| 512 | 0.9043 | 0.9910 | 10 |
| 256 | 0.7500 | 0.9891 | 10 |
| 128 | 0.6078 | 0.9898 | 10 |
| 64 | 0.7031 | 0.9824 | 10 |
| 32 | 0.7600 | 0.7905 | 10 |
| 16 | 0.5625 | 0.6959 | 10 |
| 8 | 0.5205 | 0.6469 | 10 |
| 4 | 0.5000 | 0.5022 | 10 |

**Table 3.** Experiment with different training size using hyper-heuristic approach

| Size of Training Data | Test Accuracies (No HH) | Test Accuracies (with HH) |
|---|---|---|
| 512 | 0.9910 | 0.9990 |
| 256 | 0.9891 | 0.9990 |
| 128 | 0.9898 | 0.9988 |
| 64 | 0.9824 | 0.9911 |
| 32 | 0.7905 | 0.9165 |
| 16 | 0.6959 | 0.9068 |
| 8 | 0.6469 | 0.6872 |
| 4 | 0.5022 | 0.5211 |

From test accuracies listed in Table 3 we can see the big improvement introduced by the hyper-heuristics approach on sizes 32 and 16. For larger size there are still performance increases but there are not much room for improvement. For smaller size like size 4, the sample is too few to be learnable hence the parameter tuning could not be much of help. This result indicates that with the hyper-heuristic tuning approach, it is possible to reduce the required training size. For applications of which training examples are few or expensive to obtain, our parameter optimisation could be very helpful.

To investigate the computational cost of the parameter optimisation, we also measured the running time of the above experiments which were all conducted on a machine with Intel core i3 with processor 1.90GHz, 4.00 GB RAM and 64-bit Windows 10. The results are presented in Figure 6 which shows the average time in seconds of 30 runs of learning on sizes 4, 8 up to 128, with and without the hyper-heuristic parameter optimisation. As can be seen on the fig-

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

ure, the optimisation process does take extra time. However the time increase is acceptable, maximum of a double time in the case of 128 training images. In comparison, exact methods for combinatorial optimisation are too expensive to be practical.

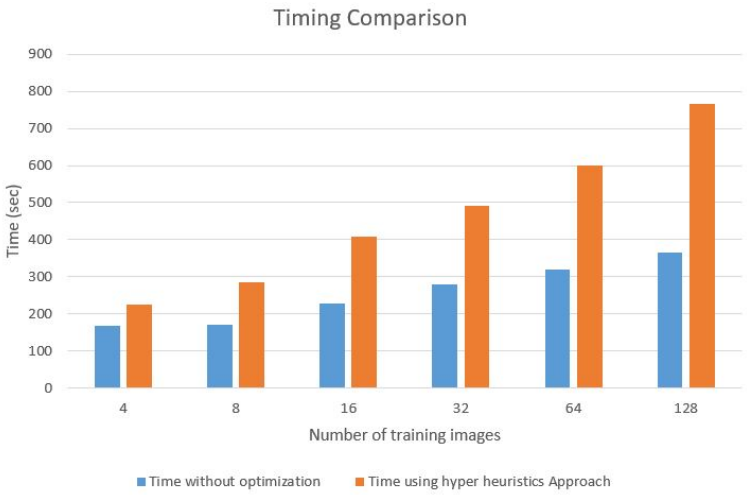


**Fig. 6.** Comparison on Running Time with and without Hyper-Heuristic Optimization

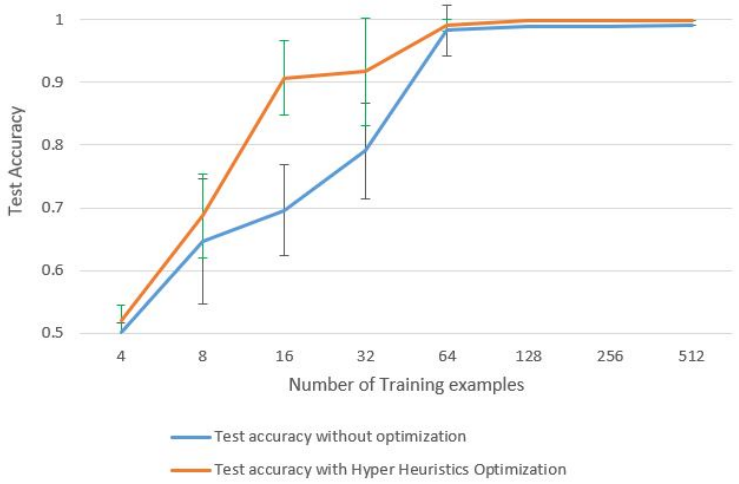


**Fig. 7.** Test accuracies with and without Hyper-heuristic Parameter Optimisation

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

From the above experiments we can see the hyper-heuristic approach can greatly improve learning without requiring too much extra computational resources. To further illustrate the differences realised by our hyper-heuristic approach, the test performance on different sizes are plotted as in Figure 7. The lines represent the average of 30 runs, while the bars on size 4 to size 64 are the standard deviation of these 30 runs of using and without using hyper-heuristic optimisation. As can be seen in this figure, the gap at size 16 and at size 32 are significant. To verify the significance, T-tests are conducted on test accuracies on size 16 which resulted a p-value of 0.000002, and on size 32 which resulted a p-value of 0.000025. These p-values are way below the null hypothesis threshold 0.05, showing the differences that hyper-heuristic optimisation made on test performance are indeed significant.

## 6 Conclusions

In this work, we utilised deep learning to classify images of good quality versus images of poor quality without understanding or examining the image content. Based on our investigation using MNIST and n-MNIST benchmark, we can conclude that deep learning with convolutional neural networks can handle this type of image classification tasks and can achieve high performance with sufficient amount of training images. Our study also confirms that the learning performance is affected by training size. Learning image quality classifiers does not need large amount of samples. However the learning would still suffer if the training set is too small.

Another important part of this study is introducing hyper-heuristic approach based parameter optimisation to automatic configure the learning. Through our experiments it is clear that this optimisation method can improve the learning especially when the training size is not sufficient but not too few. Furthermore, the additional computational cost introduced by our hyper-heuristic method is not too expensive. That makes this method attractive especially in real world applications where training samples might be expensive or difficult to obtain.

## References

1. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
2. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
3. Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

4. Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.
5. Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.
6. Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.
7. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
8. Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.
9. Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.
10. Saikat Basu, Manohar Karki, Sangram Ganguly, Robert DiBiano, Supratik Mukhopadhyay, Shreekant Gayaka, Rajgopal Kannan, and Ramakrishna Nemani. Learning sparse feature representations using probabilistic quadtrees and deep belief nets. *Neural Processing Letters*, pages 1–13, 2015.
11. Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern recognition*, 36(10):2271–2285, 2003.
12. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
13. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
14. Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of metaheuristics*, pages 449–468. Springer, 2010.
15. Nasser R Sabar and Graham Kendall. Population based monte carlo tree search hyper-heuristic for combinatorial optimization problems. *Information Sciences*, 314:225–239, 2015.
16. Nasser R Sabar, Xiuzhen Jenny Zhang, and Andy Song. A math-hyper-heuristic approach for large-scale vehicle routing problems with time windows. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 830–837. IEEE, 2015.
17. Nasser R Sabar and Masri Ayob. Examination timetabling using scatter search hyper-heuristic. In *Data Mining and Optimization, 2009. DMO'09. 2nd Conference on*, pages 127–131. IEEE, 2009.
18. Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. Grammatical evolution hyper-heuristic for combinatorial optimization problems. *strategies*, 3:4, 2012.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41

19. Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, 19(3):309–325, 2015.
20. Salwani Abdullah, Nasser R Sabar, Mohd Zakree Ahmad Nazri, Hamza Turabieh, and Barry McCollum. A constructive hyper-heuristics for rough set attribute reduction. In *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*, pages 1032–1035. IEEE, 2010.
21. Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. A dynamic multi-armed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, 45(2):217–228, 2015.
22. Nasser R Sabar, Jemal Abawajy, and John Yearwood. Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems. *IEEE Transactions on Evolutionary Computation*, 21(2):315–327, 2017.
23. Nasser R Sabar, Ayad Mashaan Turky, and Andy Song. Optimising deep belief networks by hyper-heuristic approach. In *CEC 2017-IEEE Congress on Evolutionary Computation*, 2017.
24. Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_41