Optimizing Deep Learning Parameters By Hyper heuristics Evolution

A minor thesis submitted in partial fulfillment of the requirements for the degree of Masters of Computer Science

Muneeb ul Hassan
School of Computer Science and Information Technology
Royal Melbourne Institute of Technology
Melbourne, Victoria, Australia

June 2, 2017

Declaration

This thesis contains work that has not been submitted previously, in whole or in part, for any other academic award and is solely my original research, except where acknowledged.

This work has been carried out since Feb 2017, under the supervision of Andy Song.

Muneeb ul Hassan
School of Computer Science and Information Technology
Royal Melbourne Institute of Technology
June 2, 2017

Acknowledgements

I would like to take this opportunity to convey my special thanks to my supervisor Dr Andy Song for his incredible support during the semester. His support is beyond an ordinary supervision. Thanks to Nasser. R Sabar from Queensland university of Technology for helping me in my thesis and for commenting on my work and his great suggestions, to my parents for their continual support throughout my life for giving me endless support, motivation and courage to overcome the hardships.

Contents

List of Figures

List of Tables

Abstract

Deep Convolutional Neural Networks (CNN) has been successful is classification especially image recognition and video processing tasks. However, the performance of a convolutional neural networks is often highly dependent on the setting of the Hyper-Parameters. In this work, we first formulate the deep learning model to classify the good quality images and bad quality images without understanding the content of the image. The well known dataset were used for evaluating the performance of the convolutional network. The Hyper-Heuristics approach is used to tune the parameter of the convolutional network. Our experimental result show that this hyper-heuristic approach can achieve high accuracy and it is suitable to tune the hyper-Parameters. In addition, the result illustrate that this approach is optimal and efficient and obtain better accuracy with all training data sizes.

Note: *This Thesis has been submitted to SEAL17 Conference as a Conference paper.*

Chapter 1

Introduction

Deep learning is growing very fast and its one of the fast growing area in artificial intelligence. It has been used in many fields extensively including real time object detection[6], image recognition[7] and video classification[8].It also attain good result in understanding speeches and natural language processing e.g teaching machines to read[9], Generating sequence[10], speech recognition[11] etc. It gain popularity in recent years when AlphoGo beat the human champion in a game of Go[12]. Deep learning usually implemented as Convolutional Neural Network, Deep Belief Network, Recurrent Neural Network etc. One of the problems of deep neural networks is configuration of its parameters and it is sensitive to parameter tuning. The parameters including batch size, optimizer, drop out rate, epochs, learning rate etc. significantly impact the performance of our learning algorithm. In this study we first investigate the deep learning model for image classification and we would like to introduce a method to automatically tune the parameters to improve performance.

Image Classification has been studied of many decades and its one of the significant area in computer vision.The task of image classification is to differentiate between images according to their categories. Image classification usually have a set of targets e.g recognizing digits in images[2], recognizing human faces[13], recognizing human action[14] and recognizing target objects in images like car vs no car, book vs no book etc. However, in real application classification of good vs bad images have been very important which can automatically find out which image need improvement and which image needs to be rejected from image collection. This would also help us to automatically generate aesthetic evaluation of images. The goal is to utilizing deep learning to differentiate not just the set of target but evaluating the quality of images i.e good quality image vs bad quality image.

1.1 Research Questions

1. How to formulate deep learning model to differentiate between good quality images and bad quality images without understanding the content of the image?

2. How to minimize the training size and still achieve good accuracy in classifying good vs bad images?

3. How can we use automated hyper-Parameter tuning method to achieve optimal classification results?

In order to answer the above questions we first investigate the previous strategies and talk about how can we achieve the good accuracy with the given number of standard dataset and once we achieve that, we reduce the dataset gradually and see the performance of the network with the same parameters. We will try to find the hyper-Parameters through optimization techniques which will increase the model accuracy with the low number of training size.

1.2 Research Contribution

This research made the contribution in the field of artificial intelligence and deep learning for computer vision. In this work, we tried to make a model and classify images and will try to increase the accuracy.Secondly, we try to reduce the training size and get the good accuracy. We are using optimization technique to search the hyper parameters.

1.3 Organization

The rest of the thesis is organized as follows. Chapter 2 describe the background of the material that is used in the thesis e. g we will give the overview of convolutional network and its different layers, keras and tensorflow etc. In Chapter 3, we will talk about the methodology, how we approach this problem and then chapter 4 will give our experiments results as well as analysis of our result. In Chapter 5, we concluded our thesis and discuss some ideas about future work.

Chapter 2

Background

Artificial neural networks are becoming increasingly popular in the machine learning field due to their successful use in many different problem domains. This project focus on using neural networks to try to solve image classification problem. These problems are a type of supervised learning, where each image has an associated label to indicate a category that it belongs to. The images are fed into the neural networks as input to train the network and then check the accuracy on the images which are not fed into the networks i.e some new images.

The term 'good' refers to the images which are clean from any noise and they are not blurred whereas the term 'bad' refers to the images which are blurred or images with some noise. The task for selecting good images from a large database of pictures is hard and time consuming. This problem is very interesting because of photographic and images communities are become more common and to select images from their database which are not relatively free from any noise is hard. Lets take an example of Instagram, one of the largest social networks of photo community, which hosted a million of images and if we want to separate the good and bad photos manually, its pretty hard and we want to attempt this task using computational techniques. We are using deep learning techniques to classify the images. The convolutional neural networks model is one of the best method to classify images. We try to determine the degree of accuracy when classifying the images in such scenario and made an attempt to increase the accuracy. To increase the accuracy, we need to modify the networks parameters such as the learning rate and drop out rate, optimizer, number of epochs, batch size and number of neurons in the dense layer.

Neural networks are biologically inspired programming model which can learn from observational data. Deep learning is the most successful and powerful technique for learning. Neural network provides state of the art solution of many problems in video processing and image recognition.

Before neural networks, hand crafted computer programs were written for image classification but neural networks made it easy without explicitly define the image description. The idea is to take the large number of images, known as training data and build a model which can learn from those training data and we test the accuracy on test data.

In neural networks, the first layer is always the input layer and it the leftmost layer in the network architecture. The rightmost layer is the output layer. The middle layer is called the

Hidden layer because the neurons in this layer are neither input nor outputs. The networks may have number of hidden layers range from one to infinity, the higher the number of hidden layers, the complex the architecture is.

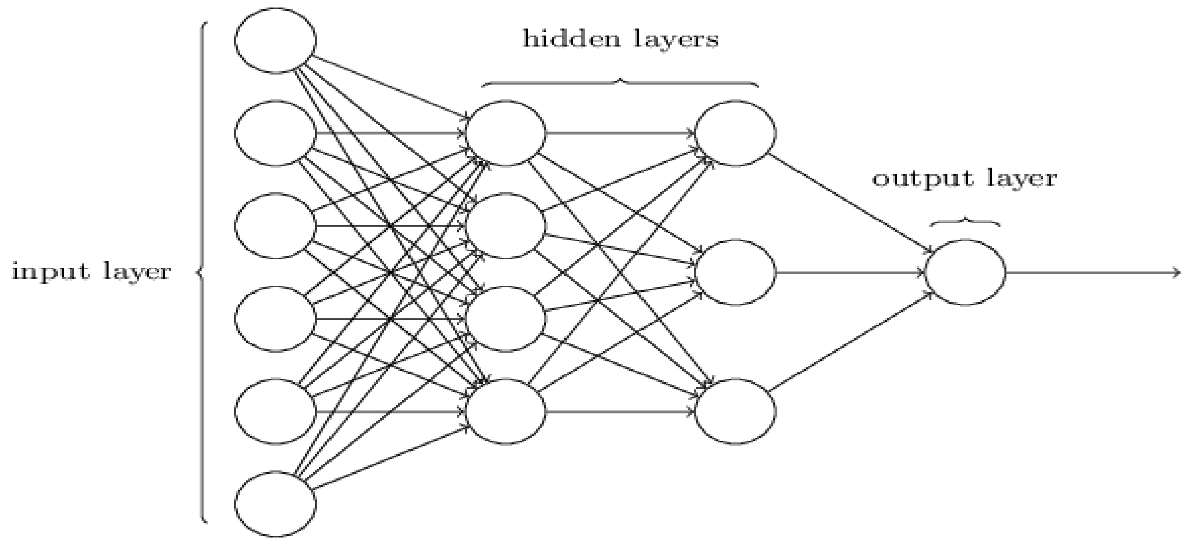

Figure 2.1: Example of neural network[1]

Convolution Neural Network is a variant of multilayer perceptron. The Classical CNN is LenNet which shown in figure 2.2. Current approaches for image classification make essential use of convolutional neural network. It was first used by lecun et al.[2]. The major breakthrough occur in 2012 when Alex et al. introduced AlexNet[15]. After that many subsequent studies have applied deep learning for image recognition in which convolutional neural network gives some astonishing results. For example, MNIST digit recognition task approaches near human(<0.25%)[16].

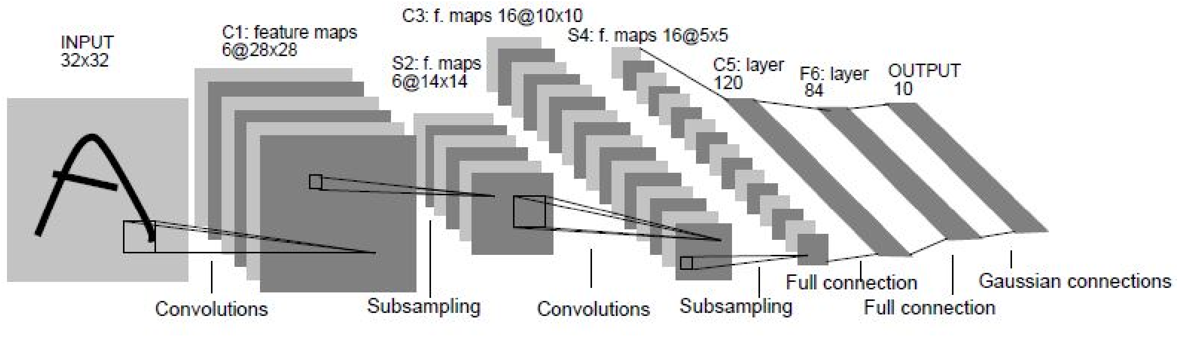

Figure 2.2: Architecture of LeNet By Lecun et al[2]

Convolutional network is far better for classification task as compared to other approaches and it is evident from number of results which came in the past few years. After AlexNet[15], Zeiler et al introduce ZF-net which achieved state of the art result with the error rate of 11.5% on ILSVRC 2013 dataset[17]. Google introduces the inception module in deep convolutional network and achieve remarkable result with error rate of 6.7% on dataset of ILSVRC 2014[18].

It gives the idea that the CNN layers do not need to be stacked sequentially. Convolutional neural networks are currently the most powerful tool for solving computer vision problem. So that's why CNN is the most significant approach to solve aesthetic judgment task.

In convolutional neural network, finding the right hyper-parameter is quite hard and it require expertise and can be time consuming. Grid search, Bayesian Search and manual Search have been used widely for hyper-parameter optimization but these technique are not optimal. Hyper-Heuristics Approach is one of the approach which is used to on image reconstruction benchmark. This approach is task independent.

2.1 Convolutional Neural Network

Neural network is a large collection of simple neural units. These neurons are connected with each other and with different layers. One neuron take one input and the number of inputs defined the number of neurons in the first layer. There are many types of neural network e.g Convolutional neural network, recurrent neural network, Modular neural network etc.

Convolutional Neural networks are the type of supervised learning algorithms. Convolutional neural networks are made up of different neurons with learnable biases and weights. It is designed to process the data that comes in the form of multiple layers[19]. Each layer consist of different number of neurons and each neuron receives several inputs, weighted sum over them and pass it to activation function and deliver it to the output. Convolutional networks transform the volume of activation function to another activation function through differentiable function.

2.1.1 Convolutional Layer

It is one of the core building block that does most of the heavy computation. The parameters of convolutional layer consists of learnable filter or kernels. Each learnable filter convoluted with width and height during the forward pass computing the 2D activation map by dot product the input and the entries of the filter and whole filter is slide over the width and height of the input volume. The weight vector which generates features map reduce the complexity of model.

2.1.2 Non-linearity Layer

In this layer, various activation functions have been applied on neurons which introduce the non linearity which is desirable for multi-layer perceptron. The activation functions applied are tanh, Sigmoid and ReLU. ReLu is most favourable because it is efficient and it train several time faster than other fucntions[20].

2.1.3 Pooling Layer:

A pooling function replaces the output of net at a certain location with the summary or statistics of the nearby outputs[21]. Pooling layer take each feature map from the convolutional layer and prepare a feature map using some function. There are many functions to be used in pooling layer For example, average Pool, max pool, L2 norm pooling etc. In max-pooling, pooling unit outputs the maximum activation of input region and in average poling, it outputs the average of input region.

2.1.4 Fully Connected Layer

In fully connected layer, each neuron in the previous layer connected to the every other neuron in the current layer and generate the global semantic information. Number of connected layers depends upon the architecture.

2.1.5 Regularization

Regularization is a technique which we uses to prevent from overfitting. Overfitting is a major problem in neural networks which occurs when the model learn too much and it does not generalize very well on training data. The best way to detect overfitting is to keep track of validation accuracy as the network train. If the validation accuracy is not improving, we should stop the training. The most common regularization technique is weight decay or L2 regularization. In L2 regularization we add an extra term to the cost function called regularization term.

$$C = -\frac{1}{n}\sum_{xj}\Big[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\Big] + \frac{\lambda}{2n}\sum_{w} w^2 \tag{2.1}$$

The first part of the equation is cost function and the second term is regularization term. The effect of this term is to make the network learn from small weights and minimize the cost function.

2.1.6 Dropout

Dropout is another technique to prevent overfitting. The idea is to drop the number of nuerons along with their connections randomly during training. This technique prevent from learning too much[22]. This technique reduces the overfitting and give major improvement in the accuracy as compared to other regularization technique.

2.1.7 Application Of Convolutional Neural Network

The application of convoloutional neural network are from broad range. In computer vision, CNN is used for face recognition, scene labelling, image classification, action recognition and human pose estimation. Face recognition constitutes a series of related problem e.g identifying all the faces in the picture, focusing on each face despite bad lighting and identifying unique features. In scene labelling, each object is labelled with the category it belong. Image classification is also constitute of series of problem. It identify the content of the image and labelled it with correct category. Human pose estimation is long standing problem in computer vision. This problem was solved with traditional hand crafted features but deep learning gave the better result in classifying human pose.

Convolutional Neural network provide the major breakthrough in the field of computer vision but it is also did good with natural language processing. Speech is raw and to recognize it is one of the important task in Natural Language Processing. Handling noise is one of the main task in speech recognition and CNN did it successfully. Another important application of NLP is text classification. [23] performs classification and extraction task using convolution neural network and obtain good results.

CNN is better approach than other deep learning methods in application like computer vision and natural language processing. CNN gives better result as compare to other methods because of weight sharing and local connectivity.

2.2 Keras and Tensorflow

Keras is an open source high-level neural network API which is built on top of tensorflow, and this was used to create and train neural networks in our experiment. Keras provide modularity and extensibility. In keras, we define models with different standalone configurable modules which can combine to form a neural network model. Keras is a high-level API which provides the frontend. At the backend of keras, tensorflow was used.

Tensorflow is an open source software library developed by Google.[24]. It is a flexible data flow-based programming paradigm. It creates a graph which describes how the data is flow between operations that are to be performed. It is an interface to implement machine learning algorithms[3]. Tensorflow is a directed graph which consist of nodes and it also maintain and update the state of the node. Every node have zero or more input and zero or more outputs. Values flow among the node to node and these values are arbitrary long arrays called tensors. Tensorflow supports wide range of operations including element wise mathematical operation, array operations, matrix operation, neural net building operations etc.

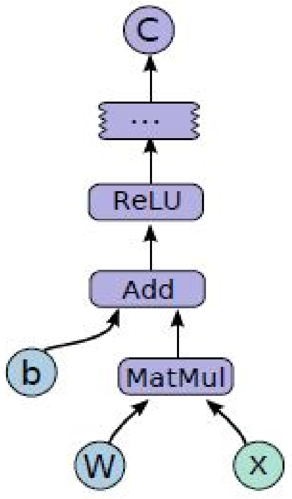

Figure 2.3: Computational graph created by Tensorflow[3]

The example of computational graph shown in figure 2.3, its a simple equation of rectified Linear Unit(ReLu) in which the matrix of weight and input(x) are multiplied with the addition of bias. Tensorflow create graph like this for each calculation and execute it in sessions.

2.3 DataSet

2.3.1 MNIST

In our study we used well known MNIST dataset[2] to classify good and bad images. The MNIST dataset is a standardized dataset that is often used for bench-marking. The MNIST dataset consists of images containing handwritten digits from 0 to 9, with each image being a 28x28 gray scale image. The training set consist of 60000 of these images and the test set contains 10000 images.

2.3.2 Noisy-MNIST(n-MNIST)

We also used another version of the MNIST dataset which is called noisy MNIST or n-MNIST[25]. There are three version of noisy MNIST which are

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 these dataset are also 28x28 gray scale image with 60000 training examples and 10000 test examples. The labels in training and test datasets are one hot encoded i.e each label is 1x10 vector.

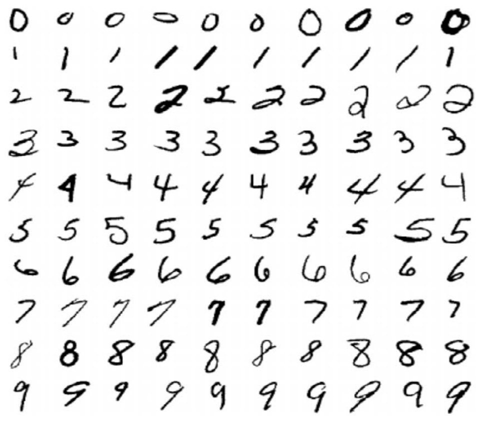

Figure 2.4: Example of images from MNIST Dataset[4]

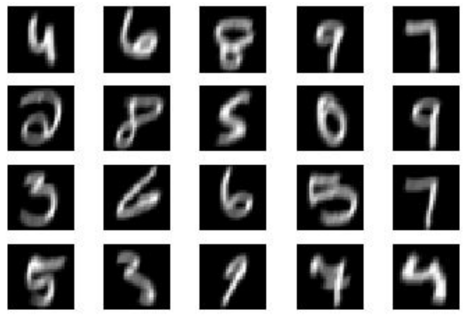

Figure 2.5: Example of images from Motion Blur Dataset.

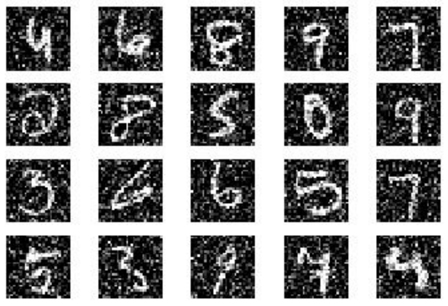

Figure 2.6: Example of images from AWGN Dataset

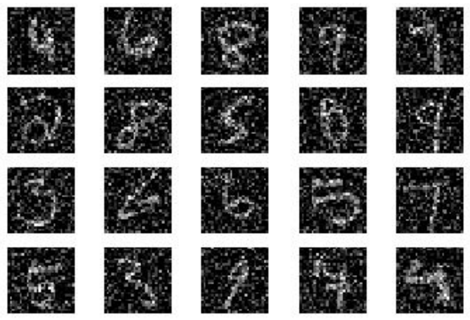

Figure 2.7: Example of images from Additive White AWGN Dataset.

The MNIST with motion blur filter is created by imitating a motion of camera 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 with reduced contrast and AWGN is created by introducing contrast range with AWGN with signal to noise ratio of 12 [25].

The MNIst and n-MNIST dataset contain images with 1 color channel. When selecting the data for our training set, validation set and test set, we are not given any information about how the sampling of these images was done. We select a random order to construct our training data, validation data and test data. We took 20% for validation data from our training set and the test data remain constant through out the experiments.

2.3.3 Imagene Dataset

Imagene[26] is an evolutionary art generating program which creates series of images with shapes and patterns using genetic programming approach. These images have no semantic content but have aesthetic appeal[27]. Each image have 500 x 500 dimensions. Both grayscale and coloured images were provided but it was decided to use grayscale images with what was previously used with MNIST. Half of the images have been modified by inserting random noise square into the image at random location. An example images have been shown at figure 2.8 and 2.9. The images with no noise inserted is one class and and images with noise is another class.

Due to large image size (500 x 500), it is very difficult to process with current system configuration as compared to MNIST data set(28 x28). The larger image size resulted in memory issue, so to avoid this condition, we downsize all the images to 40 x 40.

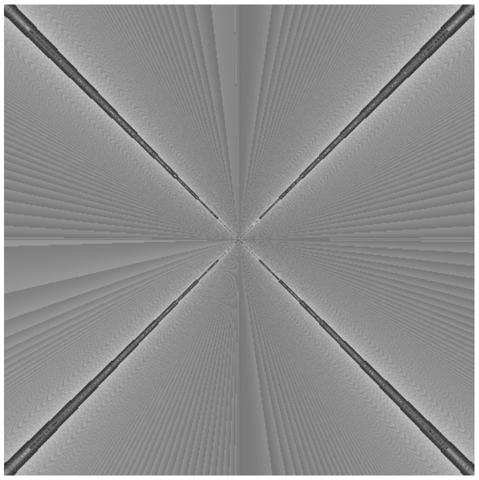

Figure 2.8: Image treated as Good

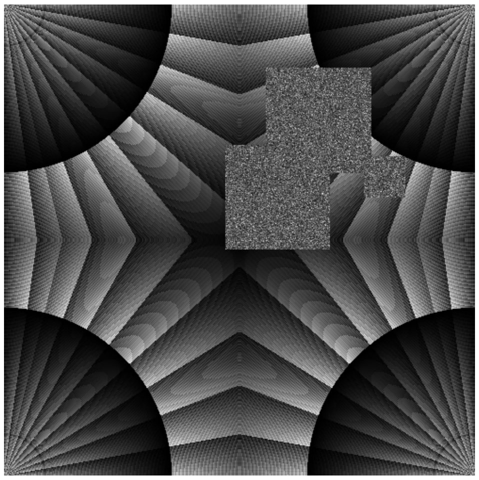

Figure 2.9: Image treated as Bad

The dataset consist of 95 images from each class giving 190 images in total. The 20% of these used for testing out of 191 which are 39 images. The total of 151 is used for training the model with the 20% used for validation.

2.4 Hyper-Heuristics Parameter Optimisation

The art of hyper heuristics optimisation is to find the heuristics methods to find the computationally hard search problems which combine machine learning techniques to select, combine, generate often simpler heuristics. The hyper heuristics is a method for selecting and generating heuristics to solve search problems[28]. Hyper-heuristics has been successful in many different fields[28][29][30][31][32].

Hyper-Heuristics begins from the initial solution which will generated randomly and then improve the solution iteratively until the best solution is met. Hyper-heuristic divided into two components

1. Low Level Heuristics
2. High level Heuristics

2.4.1 Low Level Heuristics and High Level Heuristics

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 heuristics operate on the heuristic space. It will form the heuristics to improve the result and secondly it will also determine whether to accept or reject the generated solution. This is called the Acceptance Criterion.

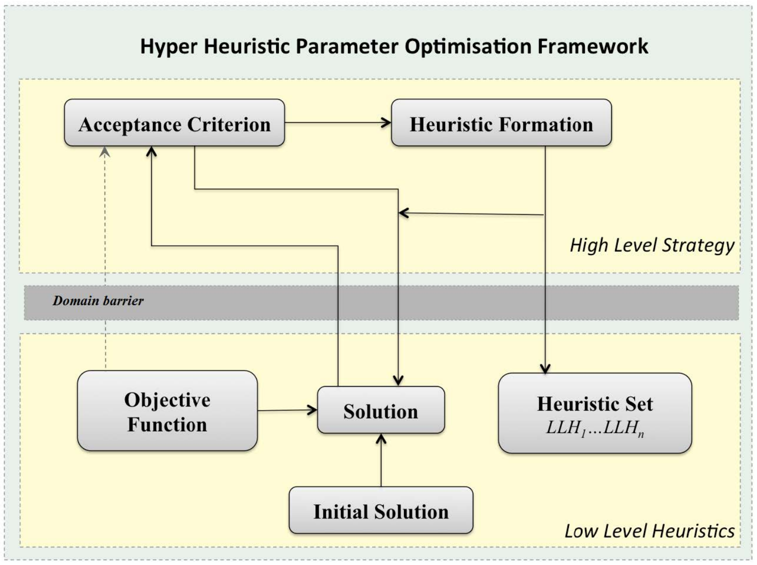

Figure 2.10: Framework for HyperHeuristics Optimisation[5]

The figure 2.10 showed the main components of Hyper-Heuristics framework. We explained each of the component in the following section.

2.4.2 Heuristic Formation

For heuristic formation, we need to collect heuristic from the pool of heuristics in the low level component. The selection criteria used is Multi-Armed Bandit(MAB). MAB select the heuristics on the basis of past record, it records all the performance of all heuristics. There are two variables which keep record of the heuristics i.e empirical reward and confidence level. The empirical reward portray the average rewards acquired by each heuristic. The confidence level is the number of time the heuristic has been applied. The empirical reward and confidence is higher the better[5].

2.4.3 Acceptance Criterion

Acceptance criterion belongs to the high level of framework and it is independent of task domain. Acceptance Criterion decides whether to accept the generated solution which is generated by applied heuristics. Monte Carlo acceptance criterion have been used to accept the solution[31]. Monte Carlo acceptance criterion accepts the solution that improve the objective function compared to the last solution that was accepted. The worse objective function can also be accepted if the following condition is met[5].

$$R < exp(\Delta f) = exp(f_t - f_{t-1}) \tag{2.2}$$

Where R is the random number between [0,1] and Δf is the difference between objective value at iteration t and $t-1$.

2.4.4 Heuristic Set

There are several heuristics used to generate new solutions. Each heuristic subsumed various characteristics in search and different search behaviours.

Parameterized Gaussian Mutation

$$X_i = X_i + N(0, \sigma 2) \tag{2.3}$$

where $\sigma 2 = 0.5$ is the standard deviation[5]. The other heuristics are same but with different σ value range from 0.2 to 0.4.

Differential Mutation

$$X_i = X_i + F \times (X_{1i} - X_{2i}) \forall i = 1...n \tag{2.4}$$

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[5].

Arithmetic Crossover

$$X_i = \lambda \times X_i + (1 - \lambda) \times X_{1i}, \forall i = 1...N \tag{2.5}$$

Where λ is random number with range 0 to 1. X_i is the current solution and $X_1 i$ is the current best solution[5].

2.4.5 Initial Solution

Initial solution is a set of CNN parameter that need to be tuned represented as 1D array. Each parameter is randomnly generated. The random function is as follows:

$$x_p = l_p + Rand_p(0,1) \times (u_p - l_p), p = 1...p \tag{2.6}$$

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[5].

2.4.6 Objective Evaluation

The objective evaluation function is used to measure the quality of solution. For simplicity We used test accuracy.

2.5 Aesthetic Evaluation

In this section we give the overview of some methods that have been adopted to solve the image aesthetic problem. Biologically speaking, a reasonable solution to this problem may lead to better understanding of the human vision[33]. This task is basically performed by human and human selects the images on their judgment.

Currently many researchers did this task by using a computational approach and machine learning techniques. Previous study has shown that the features that are regarded as the most important features in image aesthetic are colour[33], texture, composition, saturation and hue[34]. The main task in aesthetic selection of images is the selection of features and on what basis features are selected. Datta et. al. takes a data mining approach to image evaluation[34]. From a set of photographs, features are selected based on machine re-presentable heuristics that can be applied to photographs e.g. textual smoothness, golden ratio and low depth of field[34]. They establish a significant correlation between various visual properties of images and their aesthetics rating. They used 15 visual features which include saturation, hue, exposure of light, colourfulness, rule of thirds, familiarity measure, wavelet based texture, aspect ratio, region composition etc.[33]. The limitation of this method is that they just used the heuristic features and some features relevant to photographic quality. They may have ignored some important features which are significant for the classification of images like converging lines and light source classification.

In [35],they found the distinguished factors that make the photo high quality, such as simplicity, realism and basic photographic techniques. Simplicity is defined as when the subject is separate from background. They separate these two things by using focus, color contrast and lighting contrast. Realism photographic technique are integral part in assessing the photo. It include blur, color palette, composition etc. The model used hand crafted method, so this may not be so effective.

In[36], they proposed the features in terms of color presentation and spatial composition, which includes color palette, layout/edge composition and global texture features and they assist the image on that features. In[37],Marchesotti et al. proposed the use of generic image descriptor. In [38][39], they manually grouped the images of same category(e.g plant, animal, building etc.) and build an aesthetic model. In [38], they specifically work on the global/ regional feature of the image.

An attempt was made to solve this problem through machine learning. The approach was to construct a binary classier using features by using a variety of classification algorithms[33]. The algorithms used to classify images are Sequential Minimal Optimization(SMO), Random Forest, J48 and OneR. After that the most frequently selected features are the one that are highly associated with aesthetic value. As they mentioned in their paper that 55 features are not capturing many of the criteria used in making the judgment.

Many studies after that applied machine learning to photographic images to classify images on the basis of their aesthetic value [40][34][41][42]. Another work used restricted boltzman machine to find out the features in the images and classify high value and low value aesthetic images[27].

In [43], an attempt was made to solve this problem by creating query dependent aesthetic model for each image. They created five layers in which two are covolutional layers and three are fully connected layers The task is to classify images into two classes i.e good and bad. They used 19000 images to train their network which is downloaded from DPChallenge.com, which is a social photo sharing website. In this method, for every given query, a model is built to describe its unique aesthetic attributes. The key problem with this technique is the noise in the query image i.e visually and semantically similar to query image. They also defined the universal aesthetic model which stated as for the given images provided its aesthetic quality and classify it as good and bad image. In universal aesthetic model, it is assumed that all the images share the common aesthetic model.

Chapter 3

Methodology

This research is try to answer the three questions. Our investigation is also planned in three components. The first part is try to determine the good convolutional network structure and network to address the question one. Secondly, we studied the impact of the training size on the classification performance. Third, we will study the automated optimization mechanism to find the parameters that will increase the test accuracy. The series of steps we performed are listed below:

1. Formulating two datasets, one representing good image i.e with no noise, the other represent bad image i.e images with noise.
2. Train Convolutional neural network for classification task.
3. Compare the model accuracy of different dataset sizes.
4. Take average accuracy of different datasizes over 30 runs.
5. Trying to improve the accuracy of network provided small data size.
6. Analyze different combination of hyper-Parameters manually which increases the accuracy.
7. Analyze the combination of hyper-parameters using hyper heuristics framework.

We collected two different dataset, one is MNIST[2] and second is noisy-MNIST(n-MNIST)[25]. The n-MNIST is the noisy version of MNIST dataset. The n-MNIST dataset have been divided into three different dataset which are MNIST with motion blur, MNIST with Additive White Gaussian Noise and MNIST with Additive White Gaussian Noise with Reduced Contrast. All the datasets containing 60,000 training examples with 10,000 test examples. We divided our dataset into two categories i.e good images which are MNIST dataset and bad images which are n-MNIST data. We give the new label to our dataset and ignore the previous labels which are indicating the content of the images. First we randomly collect 60,000 images from n-MNIST and we have 60,000 images in MNIST. We mix the both dataset MNIST and n-MNIST and randomize it and select around 60,000 images for training. We fix the test size which consist of 20,000 images. Test size remains same throughout the experiment.

Our network consist of two convolution2D layers, with the following MaxPooling2D layer after the second convolution. Afterwards, the output of MaxPooling is flattened to 1D and pass it through the fully connected dense layer. We introduce drop out layer after the MaxPooling2D layer and after the dense layer for better generalization. ReLu(Rectified Linear Unit) activation is used for all layers. The output of dense layer use the softmax activation for probabilistic classification. The selected hyper-Parameters are listed below. Later, we used optimization technique to find the suitable combination of hyper-Parameter which gives the best accuracy.

1. The batch size represents the training examples being used simultaneously during one iteration.
2. The number of Epochs represents the number of iteration over the entire data set.
3. The number of neurons in the fully connected layer.
4. Drop out probability
5. Learning Rate of an optimization algorithm
6. The rho factor
7. The epsilon factor

During the training, we split our training data into training and validation set. Validation data consist of 20% of the training set. During training, we keep an eye on validation loss,as it stops decreasing or start increasing we stop the training because of the overfitting of the data. We used training and test accuracy to calculate how fit the model is. Training accuracy is calculated as

$$Accuracy = \frac{\sum TruePositive + \sum TrueNegative}{Totalnumber of Images} \tag{3.1}$$

There are other ways to determine the model fit e.g ROC, F-measure, MSE etc. but we are using accuracy for simplicity reasons because the data is quite balanced and we have equal emphasizes on true and false. Train and test accuracy will be a good indicator of model performance. The second research question is how the size impact the training and test accuracy. It is well understood that when we have less training set, then the computational requirement is less. So, it is more economical to train and also less training set usually introduce the problem that the data is not representative enough. Therefore, the training accuracy might not be good. So, it is a good balance between good performance vs computational cost. Therefore, we try to find minimum less training examples which will give us reasonable test performance. Therefore, we use logarithmic scale to reduce our dataset like 2^2, 2^3, 2^4 so on and so forth. We use this only for reduce our training data. We maintain the test set same through out the experiment because it is going to be a true assessment of our performance. Therefor, in our study all our experiments in terms of test are consistent. We are using only test accuracy to report the performance because over training happens which will leads to high training accuracy but low test accuracy. We emphasize that classification performance has been our primary interest in classification. Our motivation for training and analyzing models on different datasets sizes was, if there is some hyper-Parameters which effects significantly on the model.

The range of every hyper-parameter are very important for hyper heuristics algorithm to search in the specified space. In [44], they defined the the search space for different hyper parameters. The range of the hyper-parameters used in our experiment are listed in table 3.1.

Hyper-parameter	Range
Batch size	$[2^4 - 2^8]$
Dropout rate after Pooling Layer	$[0 - 0.8]$
Number of Neurons[1]	$[1 - 100]$
Epochs[1]	$[1 - 10]$
Learning Rate	$[10^{-1.5} - 10^{0.5}]$
rho	$[0.8 - 0.999]$
Epsilon	$[10^{-9} - 10^{-3}]$

Table 3.1: Range of Hyper-Parameters

Some of the hyper-parameter range are not defined in the [44]. We select the hyper-parameters which are not define in [44] by conducting the extensive experiments with different combinations and then come up with these ranges.

[1]Parameter Range not defined in [44]

Chapter 4

Experiments

In this section, we discuss the range of experiments we did and describe the results of these experiments. The first experiment we conducted is on n-MNIST dataset. We ran the same experiment on three different datasets and try to find the maximum accuracy.

1. mnist with motion blur referred as mnist-m-b
2. mnist with additive white gaussian noise(AWGN) referred as mnist-awgn
3. mnist with reduced contrast and AWGN referred as mnist-rc-awgn

The number of images for training are 48,000, validation data are 12,000 and 10,000 images for test data. The experiment which we conducted to classify MNIST digits from 0 to 9 have same architecture through out the experiment.

Datasets	Optimizer	Parameters	Train Accuracies	Test Accuracies	Epochs
mnist-m-b	adadelta	lr-0.2	0.9730	0.9739	4
mnist-rc-awgn	adadelta	lr-0.2	0.9519	0.9272	4
mnist-awgn	adadelta	lr-0.2	0.9709	0.9554	4

Table 4.1: Experiment on n-MNIST Data

We did this experiment select the best optimization algorithm for our architecture and fix it for out future use. We experiment with two mostly used optimizer Adam and Adadelta. In [45], they mentioned the Adam and Adadelta provide the most suitable convergence. We experiment with these two optimisers.

Datasets	aptimizer	Parameters	Train Accuracies	Test Accuracies	Epochs
mnist-m-b	adam	lr-0.2	0.9828	0.9631	4
mnist-m-b	adadelta	lr-0.2	0.9732	0.9660	4
mnist-awgn	adam	lr-0.2	0.9810	0.7023	4
mnist-awgn	adadelta	lr-0.2	0.9737	0.7897	4
mnist-rc-awgn	adam	lr-0.2	0.9814	0.5287	4
mnist-rc-awgn	adadelta	lr-0.2	0.9740	0.66.76	4

Table 4.2: Experiment with training on MNIST and testing on n-MNIST

It is evident that the "adadelta" gives the better accuracy as compare to "adam". We did another experiment in which we train the model on noisy mnist and check the accuracy on original mnist data. The purpose of this experiment is to see, if we get the good accuracy after tarining with noisy MNIST

Datasets	Optimiser	Parameters	Train Accuracies	Test Accuracies	Epochs
mnist-m-b	Adadelta	lr-0.2	0.9736	0.9714	4
mnist-awgn	Adadelta	lr-0.2	0.9699	0.9706	4
mnist-rc-awgn	Adadelta	lr-0.2	0.9519	0.9601	4

Table 4.3: Experiment with training on n-MNIST and testing on MNIST

We are trying different combinations of noisy MNIST data like training on one noisy MNIST data and testing on other. The adadelta is used as an optimiser with 4 epochs.

Training Dataset	Testing Dataset	Parameters	Train Accuracies	Test Accuracies
mnist-m-b	mnist-awgn	lr-0.2	0.9733	0.4139
mnist-m-b	mnist-rc-awgn	lr-0.2	0.9738	0.2319
mnist-awgn	mnist-m-b	lr-0.2	0.9703	0.9539
mnist-awgn	mnist-rc-awgn	lr-0.2	0.9704	0.9153
mnist-rc-awgn	mnist-m-b	lr-0.2	0.9500	0.9327
mnist-rc-awgn	mnist-awgn	lr-0.2	0.9513	0.9527

Table 4.4: Experiment with n-MNIST with different combinations

After range of experiment, we decided our optimization algorithm, learning rate, drop out rate and number of neurons in the dense layer to start our experiment to classify the noisy-MNIST and MNIST data. We tried out with all three dataset. The total size of dataset is 120,000 in which 96,000 are for training, 24,000 is for validation and 20,000 images is for test data.

4.1 Classification Experiment on MNIST And n-MNIST

Datasets	Train Accuracies	Test Accuracies	Epochs
mnist-awgn	0.9999	1.0	10
mnist-m-b	0.9994	0.9998	10
mnist-rc-awgn	1.0	1.0	10

Table 4.5: Classify good and bad MNIST images

The model loss is show in the figure 1

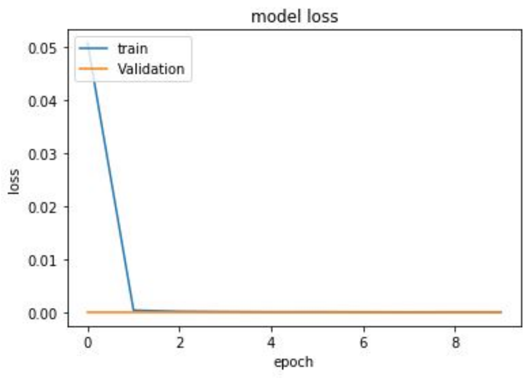

Figure 4.1: Training loss with MNIST-awgn dataset

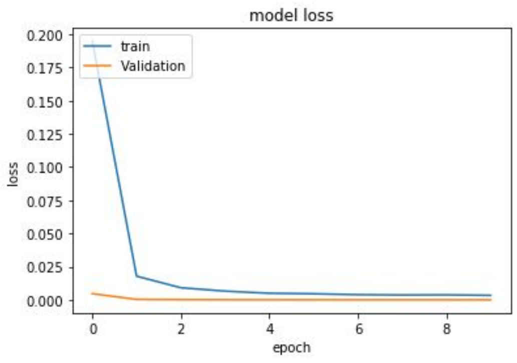

Figure 4.2: Training loss with Motion Blur Dataset.

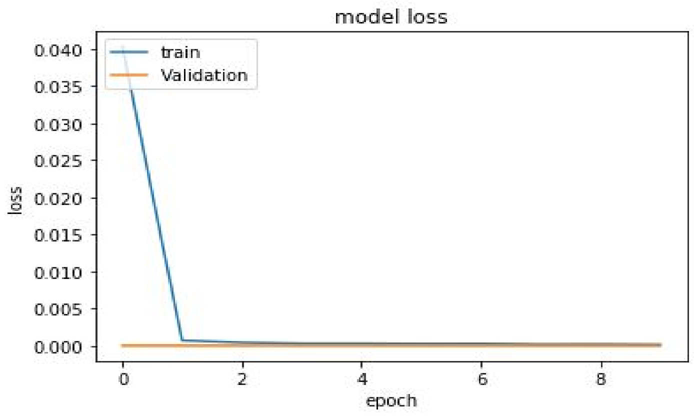

Figure 4.3: Training loss with AWGN Dataset

The images for training data are more than 60,000 and it more than enough to train the model. We decrease the data size exponentially and determine the accuracy and it is evident

Dataset Size(training samples)	Train 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	0.75	0.9891	10
128	0.6078	0.9898	10
64	0.7031	0.9824	10
32	0.76	0.7905	10
16	0.5625	0.6959	10
8	0.5205	0.6469	10
4	0.500	0.5022	10

Table 4.6: Experiment with different tranining samples

that we get a good accuracy when we have 1024 data size. We ran experiment 30 times with each data size to get the average.

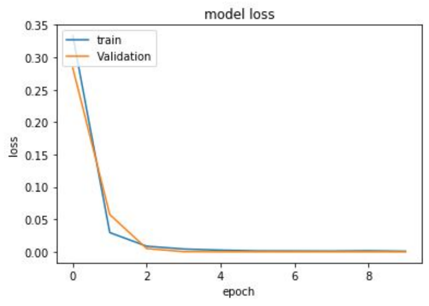

Figure 4.4: Model loss with 16384 training samples

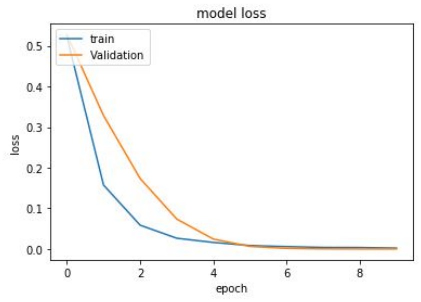

Figure 4.5: Model loss with 8192 training samples.

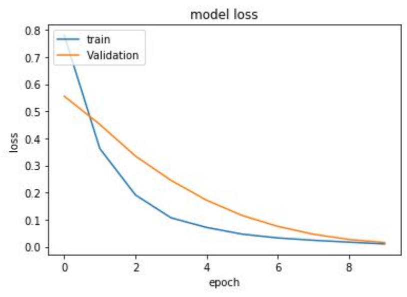

Figure 4.6: Model loss with 4096 training samples

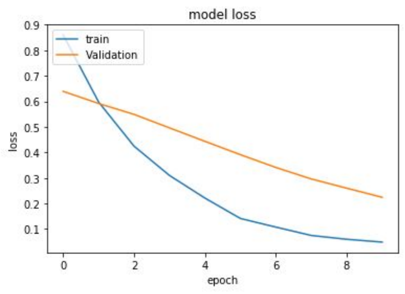

Figure 4.7: Model loss with 2048 training samples.

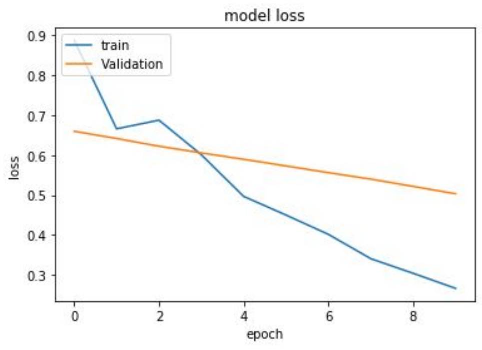

Figure 4.8: Model loss with 1024 training samples

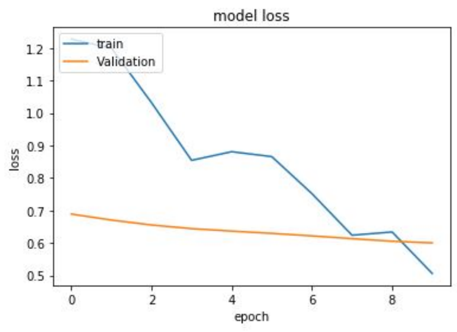

Figure 4.9: Model loss with 512 training samples.

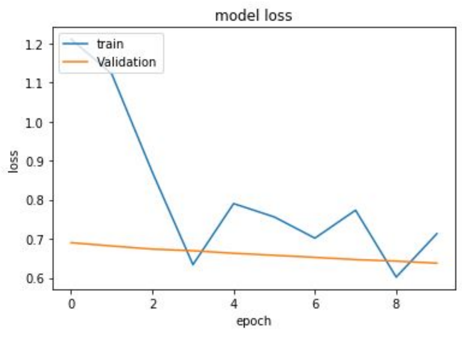

Figure 4.10: Model loss with 256 training samples

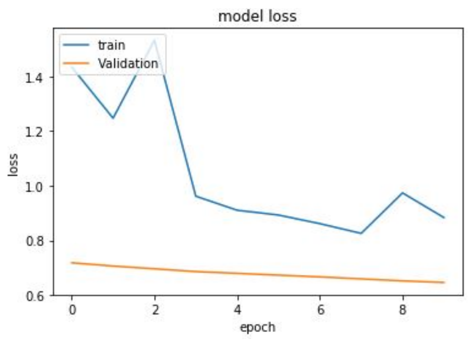

Figure 4.11: Model loss with 128 training samples.

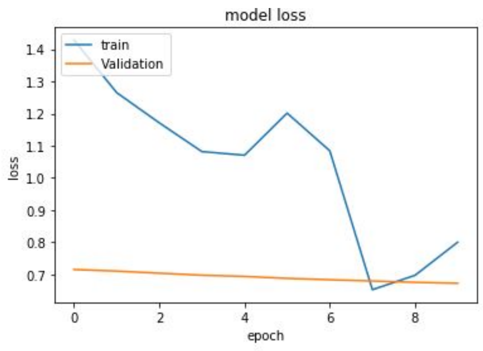

Figure 4.12: Model loss with 64 training samples

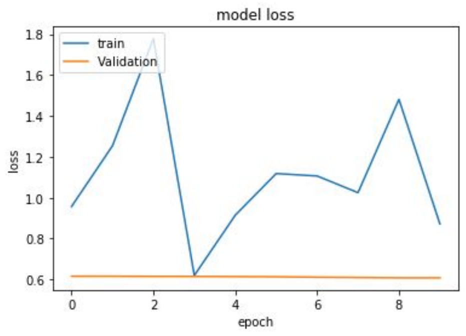

Figure 4.13: Model loss with 32 training samples.

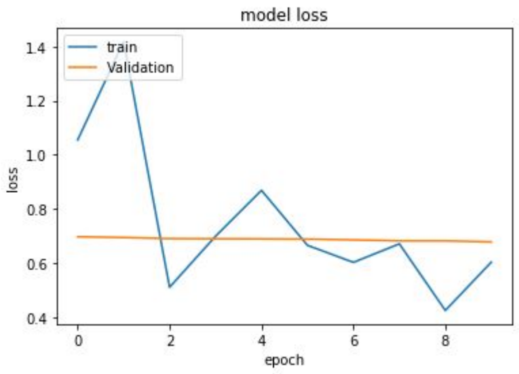

Figure 4.14: Model loss with 16 training samples

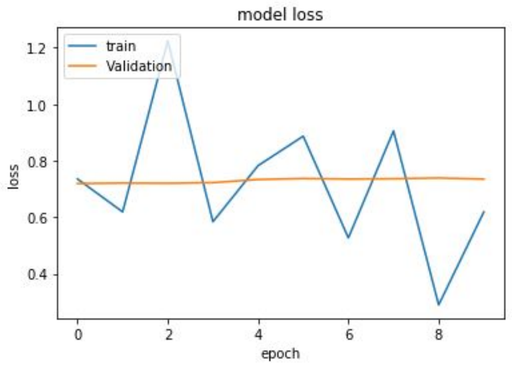

Figure 4.15: Model loss with 8 training samples.

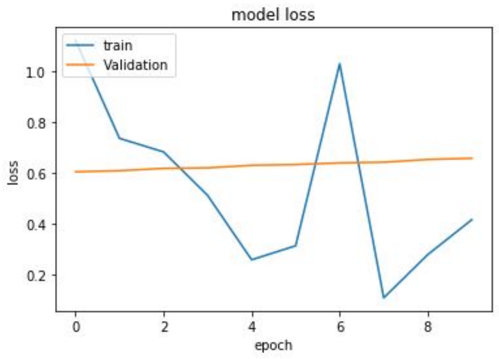

Figure 4.16: Model loss with 4 training samples

From our training loss, we can see that the model will learn until we have 16 to 32 images for training. The validation accuracy is slightly decreasing with the number of epochs but with 8 images, the validation accuracy is increasing. We are trying to increase the accuracy with training size 16 samples. Our test data is consistent through out the experiment. We are testing on 20,000 images of test data. We perform these experiment 30 times to get the average of test and training accuracy and how much our accuracy are deviated. The graph can be shown in figure 4.17.On x-axis, $4(2^1)$,$4(2^2)$,$8(2^3)$ represents represents the data size and with 16 training example we got the accuracy around 65%.

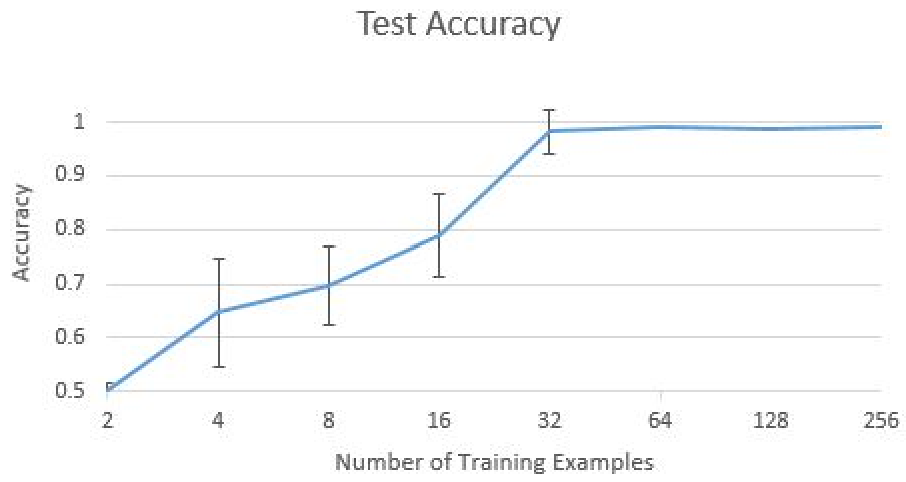

Figure 4.17: Test Accuracy graph

4.2 Experiment for Manual Parameter Optimization

As we mentioned earlier that there are two most widely used strategies for hyper-parameter optimization manual search and grid search. We tried to find the parameters manually. The results are shown in table 4.7. The problem with these strategy is the search space is very small and it takes more time to find the parameter. We conduct the manual tuning which is quite inefficient.

Dropout	Convolutional filter	No. of neurons	Epochs	test accuracy
0.1	100	28	10	0.78625
0.2	100	28	10	0.81515
0.3	100	28	10	0.5537
0.4	100	28	10	0.6509
0.5	100	28	10	0.74175
0.6	100	28	10	0.75695
0.7	100	28	10	0.62185
0.8	100	28	10	0.50555
0.9	100	28	10	0.52003
0.2	100	5	10	0.64385
0.2	100	10	10	0.7758
0.2	100	15	10	0.57075
0.2	100	20	10	0.62305

0.2	100	25	10	0.571
0.2	100	30	10	0.8212
0.2	100	35	10	0.50195
0.2	100	40	10	0.63065
0.2	100	45	10	0.58905
0.2	100	50	10	0.50135
0.2	100	55	10	0.8522
0.2	100	60	10	0.56735
0.2	100	65	10	6424
0.2	100	70	10	0.5068
0.2	100	75	10	0.60305
0.2	100	80	10	0.6347
0.2	100	85	10	0.7071
0.2	100	90	10	0.8655
0.2	100	95	10	0.63755
0.2	5	90	10	0.561
0.2	10	90	10	0.6446
0.2	15	90	10	0.50225
0.2	20	90	10	0.79795
0.2	25	90	10	0.5429
0.2	30	90	10	0.5731
0.2	35	90	10	0.52015
0.2	40	90	10	0.6211
0.2	45	90	10	0.7922
0.2	50	90	10	0.70565
0.2	55	90	10	0.51885
0.2	60	90	10	0.60065
0.2	65	90	10	0.78375
0.2	70	90	10	0.6428
0.2	75	90	10	0.86825
0.2	80	90	10	0.792
0.2	85	90	10	0.50415
0.2	90	90	10	0.5
0.2	95	90	10	0.7164

Table 4.7: Manual Search for Hyper-Parameters

Another strategy for hyper parameter search is GridSearch. The problem with this technique is that, it takes a lot of time. We try to optimize the drop out rate and number of convolutional filters and it takes around 6 hours to complete that. This trick is not efficient. This is available in scikit library.

4.2.1 Experiment using Hyper Heuristics Approach

In hyper heuristics approach, the experiment is conducted with training size from 4 to 512. We run our algorithm thirty times on each data set size and took the average of the thirty best run. We also record the training time with hyper heuristics optimization and with manual search.

Table 4.8: Experiment with different training size using Hyper Heuristic Approach

Test Accuracy		
Dataset Size(training samples)	Without optimization	With Optimization
512	0.9910	0.999
256	0.9891	0.999
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

It is evident from the test accuracy that the hyper heuristics approach is far better as compare to other approach. We also record the time of our experiment. With little bit extra time we get the huge improvement. It is evident from our timing diagram.

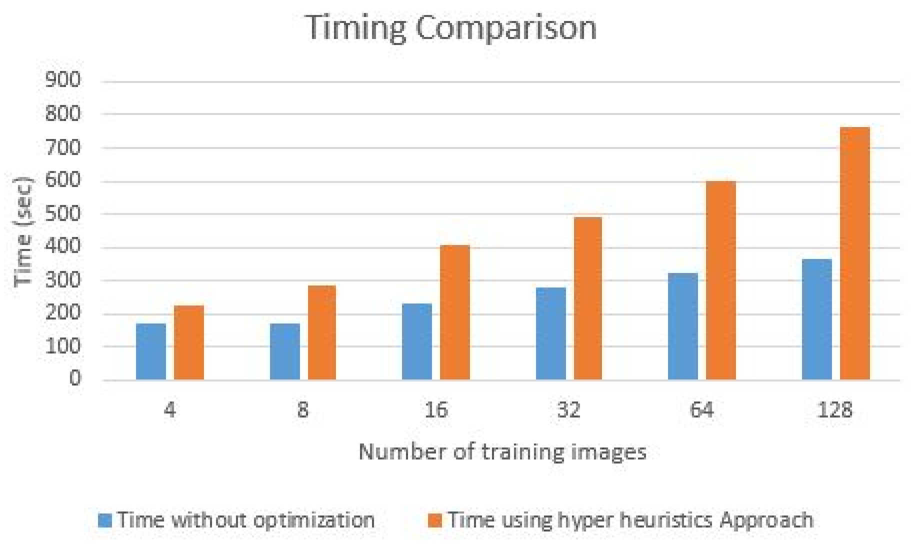

Figure 4.18: Time comparison

The experimental results are listed in table 4.6, 4.7 and 4.8. Each table represents the test accuracy with different training size. Classification performance has been our primary focus in the discovery and analysis of hyper-parameters. It is our goal to better the classification accuracy with less training size. We trained convolutional neural network on only 16 and 32 of these images and find the suitable hyper parameter using hyper heuristics method which result in increasing the test accuracy.

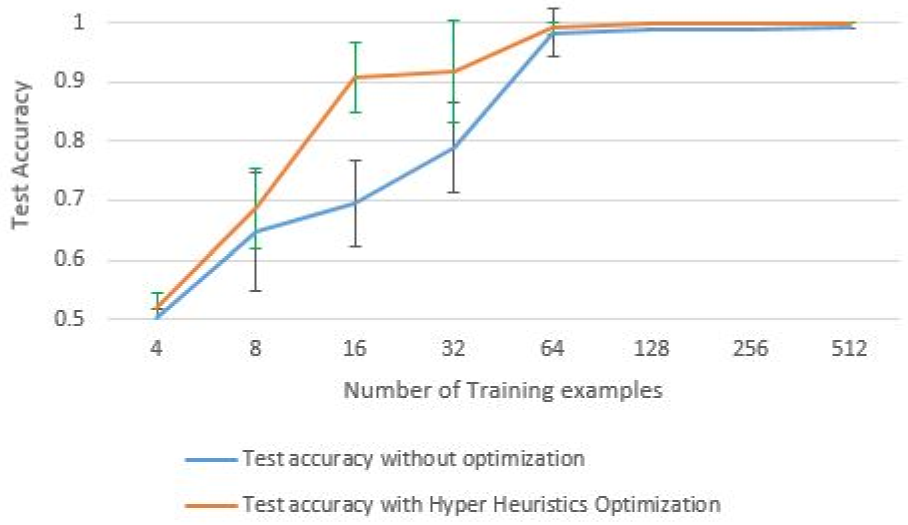

Figure 4.19: Test accuracy with and without optimization

Figure 4.19 shows the test accuracy of our classification task with different training examples. There is significant difference in the accuracy with training examples 16 and 32.

To further verify the effectiveness of our hypothesis, we further conducted a student's t test on the test accuracy of training size 16 and 32 with and without hyper heuristics optimization. The p-value suggest that the result is significant with the confidence level of 95%. This again prove the significance of the result.

4.3 Classification With Imagene Dataset

We start training by fixing the parameter same as the last experimental setup. We note that the model didn't generalize well. As shown in 4.9, with ten epochs we didn't get the suffiecient accuray. We increase the epoch from 10 to 100.

Epochs	Train Accuracy	Test Accuracy
10	0.8099	58.12
100	0.9504	58.97

Table 4.9: Imagene data Experimental result

It is noted from the training accuracy with 100 epochs that our model fit too much because the training loss is decreasing and validation loss is not decreasing after some epochs shown in 4.21.

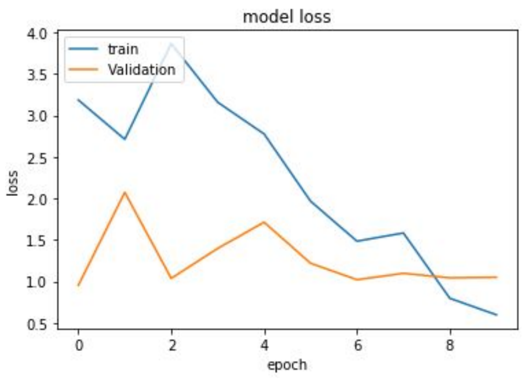

Figure 4.20: Training loss 10 epochs

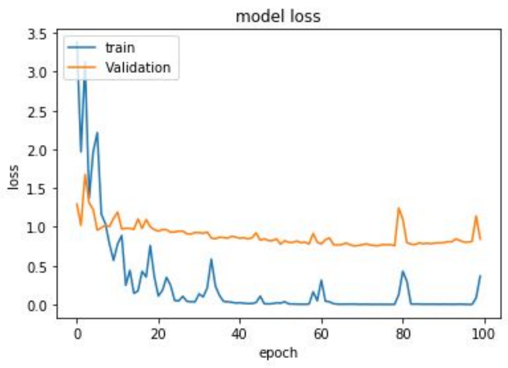

Figure 4.21: Training loss 100 epochs

4.3.1 Experiment using Hyper Heuristics Approach

We perform some experiment using hyper heuristics algorithm to find the best parameter and get the best test accuracy.

Drop out	Learning Rate	Neurons	Epochs	Test Accuracy
0.020	0.291	50	5	0.6321

Table 4.10: Hyper Heuristic with Imagene Dataset

Using Hyper heuristic approach, the test accuracy have been increase a little. Hyper heuristcs method successfully find some parameter that increase the test accuracy. The parameter range used is same as before. This is one of the issue, this algorithm didn't find the best accuracy because those parameters range are strictly defined for MNIST dataset.

Due to time constraint, we didn't able to perform large number of experiments on Imagene dataset. Significant issues were encountered when we start working with Imagene dataset. The main problem was overfitting where test accuracy were poor despite model achieve around 98%. As we downsample the images from 200x200 to 40x40, this could be an issue but this is just a hypothesis and we cannot firmly conclude that. The architecture used in this experiment is good for MNIST dataset and more deep architecture may increase the performance.

Chapter 5

Conclusions and Future Work

In this work, we investigated the hyper-heuristics based parameter optimization method to improve image classification of Deep Neural networks. The method have been tested on well known MNIST dataset. The conclusion to first question is that it is possible to differentiate between good quality images and bad quality images without understanding the content. The convolutional architecture with right number of parameter and suitable data set achieve the good accuracy and generalize well. For our second question, we conclude that with the decrease of datasize, the train and test accuracy also decreasing. To increase the accuracy, it is required to tune parameter and find out the range of parameter which result in good accuracy. Finding the parameter manually is hard task and it is inefficient. The conclusion to our third question is that the Hyper-heuristics optimisation approach is valid which gives good result even with the smallest dataset. Furthermore, the comparison result with MNIST without optimisation and MNIST with optimisation were very encourage as hyper heuristic method increase the accuracy upto 90% with 16 and 32 training size. Future work could explore many Deep learning advances. The most obvious one is to do the training and doing the same procedure on CIFAR-10. The more extension to this is to test the same hypothesis on other benchmark dataset.

Appendix A

Testbed Configuration

Processor : Intel(R) Core(TM) i3-3227U CPU @ 1.90GHz

RAM: 4.00 GB (3.88 GB usable)

Operating System : Microsoft Windows 10 Education

System Type: 64-bit Operating System, x64-based processor

Python version: 3.5

Tensorflow Version: 1.0.0

Keras Version: 2.0.0

Anaconda Version: 4.3.17

References

[1] Michael A. Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015. `http://www.neuralnetworksanddeeplearnin.com`.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[17] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[20] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[22] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[23] Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 39–48, 2015.

[24] 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.

[25] 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.

[26] Qinying Xu, Daryl D'Souza, and Vic Ciesielski. Evolving images for entertainment. In *Proceedings of the 4th Australasian conference on Interactive entertainment*, page 26. RMIT University, 2007.

[27] Allan Campbell, Vic Ciesielksi, and A Kai Qin. Feature discovery by deep learning for aesthetic analysis of evolved abstract images. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 27–38. Springer, 2015.

[28] 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.

[29] Nasser R Sabar and Graham Kendall. Population based monte carlo tree search hyper-heuristic for combinatorial optimization problems. *Information Sciences*, 314:225–239, 2015.

[30] 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.

[31] 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.

[32] Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. Grammatical evolution hyper-heuristic for combinatorial optimization problems. *strategies*, 3:4, 2012.

[33] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Wang. Studying aesthetics in photographic images using a computational approach. *Computer Vision–ECCV 2006*, pages 288–301, 2006.

[34] Vic Ciesielski, Perry Barile, and Karen Trist. Finding image features associated with high aesthetic value by machine learning. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 47–58. Springer, 2013.

[35] Yan Ke, Xiaoou Tang, and Feng Jing. The design of high-level features for photo quality assessment. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 419–426. IEEE, 2006.

[36] Kuo-Yen Lo, Keng-Hao Liu, and Chu-Song Chen. Assessment of photo aesthetics with efficiency. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2186–2189. IEEE, 2012.

[37] Luca Marchesotti, Florent Perronnin, Diane Larlus, and Gabriela Csurka. Assessing the aesthetic quality of photographs using generic image descriptors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1784–1791. IEEE, 2011.

[38] Wei Luo, Xiaogang Wang, and Xiaoou Tang. Content-based photo quality assessment. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2206–2213. IEEE, 2011.

[39] Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2408–2415. IEEE, 2012.

[40] Allan Campbell, Vic Ciesielski, and Karen Trist. A self organizing map based method for understanding features associated with high aesthetic value evolved abstract images. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2274–2281. IEEE, 2014.

[41] Philip Galanter. Computational aesthetic evaluation: Past and future. In *Computers and Creativity*, pages 255–293. Springer, 2012.

[42] Penousal Machado and Amílcar Cardoso. Generation and evaluation of artworks. In *Proc. of the 1st European Workshop on Cognitive Modeling, CM'96*, pages 96–39. Citeseer, 2010.

[43] Xinmei Tian, Zhe Dong, Kuiyuan Yang, and Tao Mei. Query-dependent aesthetic model with deep learning for photo quality assessment. *IEEE Transactions on Multimedia*, 17(11):2035–2048, 2015.

[44] Ilya Loshchilov and Frank Hutter. Cma-es for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

[45] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.