

Identifying Hyperspectral fruit images using Convolutional Neural Network on Google Cloud Platform

Sushmitha Suresh

M.Sc. in Computing
in Big data and
Analytics

2020



lyit

Institiúid
Teicneolaíochta
Leitir Ceanainn

Letterkenny
Institute
of Technology

Computing Department, Letterkenny Institute of Technology, Port Road, Letterkenny, Co. Donegal,
Ireland.

Identifying Hyperspectral fruit images using Convolutional Neural Network on Google Cloud Platform

Author: Sushmitha Suresh

Supervised by: Harran Martin

A thesis submitted in partial fulfilment of the requirements for the
Master of Science in Computing in Big data and analytics

Declaration

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of Master of Science in Computing in **Big data and Analytics**, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution. I understand that it is my responsibility to ensure that I have adhered to LYIT's rules and regulations.

I hereby certify that the material on which I have relied on for the purpose of my assessment is not deemed as personal data under the GDPR Regulations. Personal data is any data from living people that can be identified. Any personal data used for the purpose of my assessment has been pseudonymised and the data set and identifiers are not held by LYIT. Alternatively, personal data has been anonymised in line with the Data Protection Commissioners Guidelines on Anonymisation.

I give consent for my work to be held for the purposes of education assistance to future Computing students at LYIT and it will not be shared outside the Computing Department of LYIT. I understand that my assessment may be shared with any other third party and will be held securely in LYIT in line with the Records Retention Policy.

Signature of Candidate

Date

Acknowledgements

I sincerely thank Prof. Harran Martin for the valuable guidance given from the start until the end to carry out this dissertation

I wish to express my gratitude to Prof. Edwina Sweeney for providing me enough credits to work on the Google Cloud Platform while executing my artefact

I wish to thank all my academic staff Prof. Lennon Ruth, Prof. Sweeney Edwina, Prof. Sweeney Angela, Prof. Connolly James, Prof. Eogan Furrey for their support during the course

I thank my parents and friends for their support and encouragement given to carry out this dissertation

Abstract

Computer vision and Image processing combination is playing the vital role in automation of fruit classification. Hyperspectral imaging is a continuously growing area and has received considerable attention in the last decade. Hyperspectral data provide a wide spectral range, coupled with a high spectral resolution. These characteristics is suitable for detection and classification of surfaces and chemical elements in the observed images. Several factors make hyperspectral fruit recognition a challenging task which includes fruits that arise in fluctuating brightness, obstructed by other object, geometric changes, varied rotated angles, texture reflectance properties etc. Since grading of fruits based on these quality features is difficult to be detected by human inspection, automatic classification using Convolutional Neural Network (CNN) has been proposed. All the existing methods executed CNN model to classify images on local machines and faced challenges in obtaining low classification accuracy, high computation load, and long running time to train and test the model. Thus, in this thesis Convolutional Neural Network using TensorFlow framework is used to classify multiple fruit images shot in different angles and to overcome the limitations observed in the existing methods, the execution of the CNN model is carried on the cloud platform specifically on Google Cloud Platform (GCP).

Acronyms

Acronym	Definition	Page
CNN	Convolutional Neural Network	2
DNN	Deep learning Neural Network	2
GPU	Graphical Processing Unit	3
API	Application Programming Unit	3
ML	Machine Learning	6
EoC	Ensemble of Classifiers	8
DCS	Distributed Control System	9
SVM	Support Vector Machine	10
ISVM	Iterative Support Vector Machine	10
BoF	Bag of Features	11
SURF	Speeded-Up Robust Feature	12
KNN	K-Nearest Neighbor	13
MRF	Markov Random Forests	14
LBP	Linear Binary Pattern	15
CLBP	Complete Linear Binary Pattern	15
PCA	Principal Component Analysis	16
DoG	Difference of Gaussian	17
SIFT	Scale Invariant Feature Extraction	24
mCNN	Multichannel Convolutional Neural Network	25
ReLU	Rectified Linear Unit	35

Table of Contents

Declaration	10
Acknowledgements.....	11
Abstract	12
Acronyms	13
Table of Figures.....	16
1. Introduction.....	1
1.1. Purpose.....	1
1.2. Background	2
1.3. Aim	3
1.4. Research Question	3
1.5. Objectives	3
2. Literature Survey.....	5
2.1. Introduction.....	5
2.2. Existing Classical statistical approaches for Image classification	7
2.2.1. Use of Dynamic classifier Ensemble Algorithm	8
2.2.2. Hyperspectral image classification using Iterative Support Vector Machine (ISVM)	10
2.2.3. Comparison of different Machine learning framework to obtain the best framework for image classification.....	11
2.2.4. Use of Active learning technique based on Markov Random fields for classification of hyperspectral images	14
2.2.5. Use of Multiscale spatial texture features to classify hyperspectral images....	15
2.2.6. Use of Projection-based random forests for feature independent classification of hyperspectral images	16
2.2.7. Image classification using Visual saliency model	17
2.2.8. Classifying images using Manifold learning based non-linearity dimensionality reduction	18
2.3. Use of Convolutional neural network for image classification	20

2.3.1.	Addition of middle output layer for inference to accelerate image classification	20
2.3.2.	Use of Caffe deep learning framework for image classification	23
2.3.3.	Use of Inception-v3 for classifying flowers into different categories.....	24
2.3.4.	Use of Multi-channel convolutional neural network to extract invariant feature for image classification.....	24
2.4.	Conclusion	26
3.	Design and Implementation	28
3.1.	Introduction	28
3.2.	Dataset	28
3.3.	Technology and computing environment	28
3.4.	GCP environment setting for deep learning.....	35
3.5.	Algorithm for classification.....	37
4.	Results	38
4.1.	Introduction	38
4.2.	Observations- Runtime in GCP	38
4.3.	Conclusion	42
5.	CONCLUSION	43
5.1.	RECOMMENDATION	43
Appendix A: References		44
Appendix B: Code Listing		48

Table of Figures

FIGURE 2.1 THE PROPOSED DYNAMIC CLASSIFIER ENSEMBLE ALGORITHM	9
FIGURE 2.2 A FLOWCHART OF ITERATIVE SUPPORT VECTOR MACHINE	11
FIGURE 3.1 GENERAL ARCHITECTURE OF CLOUD COMPUTING SERVICES.....	29
FIGURE 3.2 GPU BASED CLOUD COMPUTING FRAMEWORK	30
FIGURE 3.3 INSTANCE CREATION.....	31
FIGURE 3.4 CREATION OF BOOT SETUP.....	32
FIGURE 3.5 CREATION OF COMPUTE ENGINE.....	33
FIGURE 3.6 DEEP LEARNING VM	34
FIGURE 3.7 GCP ENVIRONMENT SETUP 1	35
FIGURE 3.8 GCP ENVIRONMENT SETUP 2	36
FIGURE 3.9 A TYPICAL CONVOLUTIONAL NEURAL NETWORK	37
FIGURE 4.1 EXECUTION OF CNN PYTHON FILE	38
FIGURE 4.2 CREATED CNN MODEL	39
FIGURE 4.3 SNAPSHOT OF TIME TAKEN TO TRAIN AND TEST CNN MODEL.....	40
FIGURE 4.4 CLASSIFICATION ACCURACY OBTAINED	40
FIGURE 4.5 CLOUD VS ON-PREM.....	41
FIGURE 4.6 PERFORMANCE IN DIFFERENT CLOUD PLATFORM.....	41

1. Introduction

1.1. Purpose

In computer vision domain, one of the machine learning method namely image recognition is used in identifying objects, people, action and so on just like how the human mind functions to perceive objects in real life. In order to achieve such recognition, the computers can exploit computer vision technology along with artificial intelligence software and a camera. With this approach, the computers are taught to understand the visible elements in a picture. By means of counting on huge databases and noticing rising patterns, the computer systems could make experience of images and formulate applicable tags and categories. While for humans and for animal brains it is very easy to identify objects in real life, but in contrast, computers find it difficult with the same chore. When we look at some things like a vehicle or a house or a bird, we commonly don't study it consciously before telling what the object really is. But, for a computer, figuring out something represents a difficult task and the stakes for identifying a solution to that task are excessive. A great way to accomplish it is through making use of meta statistics to unstructured facts. Hiring human experts for manually tagging the libraries of song and films can be daunting. However, it turns into impossible task when it comes to demanding situations inclusive of classifying photographs into special categories like fruit class where it can have comparable hues, form, size and shape. In markets, grading of fruits is based on size, colour surface defect and texture. The sorting of fruit based on these qualities' features are performed manually in few countries like India. This identification process is hard because of different factors, among which might be illumination version, occlusions, in addition to the instances when the fruit well-known shows a comparable visual look to the background (Hussain *et al.* 2018). To triumph over those, a well-generalised model that is invariant and strong to brightness and viewpoint changes and discriminative function representations are required.

One manner to remedy this trouble might be through the utilization of neural networks. A speedy development has been won in artificial neural network area in the recent years. The remarkable accuracy obtained in classification and in operational precision makes it an alternative desire when we manage picture processing issues. Neural community has numerous characters like supervised and unsupervised learning or in feature extraction as it has several layers designed to extract best fundamental features from pictures, which pretty improves the overall performance of community. Due to this, the Convolutional Neural Network (CNN) is making a big fulfilment in massive image technique.

1.2. Background

In the field of computer vision, the role of Convolutional Neural Network is observed to be significant the field of speech recognition, natural language processing and in classifying images (Miikkulainen *et al.* 2019). They have been effective in handling tasks in the areas of processing images and in predictive modelling. As the data is growing continuously, there have been algorithms developed to analyse these data with higher accuracy. One primary benefit of CNN is that it trains the complete system starting from raw pixels of an image until it meets the end system in classifying images into its respective categories, which suppresses the requirement of designing feature extractor manually (Khaing *et al.* 2018)

A powerful GPU is needed to efficiently work with CNN. Now and again, people who have no such computational energy, time and massive-scale training set alas cannot take gain of the effective CNN and therefore executing CNN on cloud platforms will conquer such trouble. Since Google Cloud Platform maintains information secure and makes it to be had speedy, its services are superior for reading large quantity of information, and hence it is utilized in this project to train and test a CNN model. Google Cloud Platform is in use since 2006. Its cognizance is in coping with infrastructure, in configuration of networks, and in arranging servers. Services such as VM instance creation, Machine learning, storage buckets, Networking has been offered by Google Cloud Platform (GCP). With the Machine learning services offered, CNN model can be built and carried out to categorize images via the usage of python programming language on the cloud platform. Considering that, TensorFlow framework offers a couple of stages of abstraction in constructing ML models to choose the

proper one for as according to the user desires. It's used in this project to construct and train CNN models by the usage of the high-level Keras API.

1.3. Aim

This thesis aims to classify hyperspectral images of fruits into their respective categories using Convolutional Neural Network (CNN) under deep learning on Google Cloud Platform, to overcome computational overhead.

1.4. Research Question

The primary question of the research is:

- Do Convolutional Neural Network predict different fruit images even when the images are shot in different angles are fed as an input?

The secondary question associated is:

- Do Google Cloud Platform facilitate Convolutional neural network to classify images in lesser interval of time with higher accuracy?

1.5. Objectives

- To use Google Cloud Platform (GCP) to execute Convolutional Neural Network (CNN) algorithm to classify hyperspectral fruit images
- To use GPU enabled Debian image
- To outline the previous strategies used before the existence of CNN in image classification domain
- To identify the challenges in existing literature regarding classification of images using CNN
- To outline the proposed solution to the challenge identified from literatures having no proper mitigation strategy
- To compare the accuracy percentage obtained from the proposed solution with the existing solutions identified through the literature to conclude the prevailing thesis

1.6. Conclusion

Having discussed the importance of classification of fruits using Convolutional neural network under deep learning, it was outlined the need of GPU for users having no adequate computational platform to build the respective model, in order to efficiently work with CNN. With this concern, the Google Cloud Platform is adopted here in this prevailing thesis upon which the entire image classification implementation is carried out with the objective to reduce the computational overhead. Having said that, to further know the challenges faced in the existing technologies that are already implemented so far in obtaining higher accuracy percentage in identifying images, the next section i.e., chapter 2 summarizes the literature review in the field of image classification domain.

2. Literature Survey

2.1. Introduction

Progression in the field of image classification to obtain higher precision rate has made hyperspectral image classification to be functioning point as of late. In hyperspectral imaging a wide range of colors in an image are determined to know what exactly is imaged rather than just recognizing RGB i.e., Red, Green and Blue hues in every pixel. However, a spectral band is created when a light is striked for each pixel in an image to uncover the details of what is imaged.

Hyperspectral images provide a spectral information to perceive and recognize images that look similar but have unique properties. Thus, hyperspectral imagery gives more precise and detail data extraction than is conceivable (Hamouda *et al.* 2018). Spatial Resolution depicts how much detail in a photographic picture is noticeable to the human eye. The spectral resolution gives fine and point by point data in Hyperspectral image classification and therefore takes into consideration perceiving and classifying materials with contrasts in their reflectance marks. With the progression and of imaging advances and the substantial utilization of images in various fields, according to the 2019 social media statistics it's been stated 40 billion images are uploaded and shared on the social media every day (Dustin W. Stout 2019). The necessity for extraction, preparing and examining in image classification has changed image classification into an interesting issue among researchers with regards to Computer vision, Artificial intelligence, and in Machine Learning domains.

Having said that, several contiguous spectral bands are utilized by hyperspectral images and therefore it is observed it provides rich information when compared to multispectral images. Over the earlier decades, research attempts have been focussed on hyperspectral image classification as discussed in the next chapter by the use of traditional methodologies, for instance, combining spectral and spatial data, utilization of various ensembles, use of active learning, utilization of Markov arbitrary fields, and combination of multiple feature, and so on to classify images into their particular classes. Be that as it may, each one of these methodologies confronted difficulties in the little availability of labelled training samples bringing about the lower level of accuracy (Xia *et al.* 2016). In order to overcome this

challenge, another methodology was executed to extend lack of training samples through data augmentation by Shijie et al (2017) by the utilization of Convolutional Neural Network. A convolutional Neural Network effectively extracts different features from images and hence it is widely used in classifying images. This network divides the input image into smaller squares to learn the features of the image and then preserves the spatial relation between pixels of that image. This technique makes convolutional neural network to classify images faster. It is designed by five layers namely Convolution layer, Rectified Linear Unit (ReLU), Pooling, Sub-sampling and Fully connected layer layer (Zhao *et al.* 2019). In the convolution layer, a feature is taken from the image and its convoluted over the entire image to find the best match; at this stage a stack of filtered images are created from a single image. Mathematics is applied each time the feature is convoluted over the image and therefore each filtered image would have obtained certain values. The next layer i.e., in the pooling layer, the maximum valued filtered image is taken into consideration. The output of this layer nearly looks like the convolution layer. This process of obtaining the maximum pixel value is termed as 'max pooling'. In the ReLU layer, the filtered image possessing negative values are all converted to zero to obtain normalization. In the final layer i.e., the Fully connected layer, the input image enters this layer based on the weights and the average value considered. At this stage, the image is categorized to its suitable class (Lee and Kwon 2017).

CNN is utilized in fields like robotics, signal processing, science and research world. Since, Image processing uses algorithms to enhance images for better recognition it stands one among the popular domains under Computer vision (Naveen Joshi 2018). Today, Machine learning (ML) has demonstrated its achievements in numerous studies particularly in forecasts as these ML techniques are simpler to execute and perform superior to the traditional methodologies. There are two variables in ML namely response and predictor variables; ML learns the connection between them instead of beginning with the data model. During the learning stage ML algorithms observe inputs and responses to discover predominant patterns. Besides, Deep neural systems under the umbrella of Machine learning are the techniques whose principle has been adopted after studying the function of neurons in human brain. This idea gave incredible triumphs starting from 1980s until late 2000s. Since

CNN is configured having height, width and depth as three dimensional parameters and hence it has the capacity to predict even when the images are augmented (Shijie *et al.* 2017).

Another significant challenge is the systems having deep networks have several numbers of parameters. Such systems may have issues as it requires more memory space. With this concern, such systems are not suitable for applications having limited memory space. Henceforth, to further know in depth the difficulties confronted regarding memory space and in identifying the augmented images into its respective classes by the current approaches, the following segment clarifies in detail the literature review concerning the subject discussed up until now.

2.2. Existing Classical statistical approaches for Image classification

A hyperspectral image, otherwise called a hypercube is structured in three dimensions having two spatial dimensions dimension with rows (x) and columns (y) and one spectral dimension having z wavelengths. A hyperspectral image, thus depicted as $I(x,y,z)$ is seen either as a range $I(z)$ or as a spatial image $I(x,y)$ at each individual wavelength z (Fang *et al.* 2018). A graph is plotted between the light absorbance value and the wavelength. By observing the curve obtained from the graph, the grouping of materials in every pixel of the image is carried out. several spectral bands in hyperspectral images give distinctive spectral characteristics of an object present in the same area. Because of the unpredictable circumstance of lighting, pivots of the sensor, diverse atmospheric dissipating conditions, spectra have complex variations (Zhong *et al.* 2018). Therefore, strong and invariant highlights are to be separated for classification. Hyperspectral pictures are data-rich on account of the abundance of spatial and spectral data contained. There has been work done to extract meaningful and significant data to group hyperspectral images utilizing distinctive algorithms discussed in the following section.

2.2.1. Use of Dynamic classifier Ensemble Algorithm

The utilization of dynamic classifier ensemble algorithm for grouping hyperspectral images was proposed by (Su and Du 2015). Since classifiers utilize diverse feature descriptors in an image to increase a proficient classification performance, yet to improve the performance the classifiers are consolidated to accomplish productive yield. For this, either fusion-based or selection-based strategies are used. In this methodology selection-based strategy is utilized rather than fusion-based to combine an ensemble of classifiers to obtain efficient performance (Dhok *et al.* 2019). It is discovered that, in fusion-based, every classifier is connected in parallel and the results are consolidated to accomplish the consensus. Nonetheless, to improve the classification, an Ensemble of Classifier (EoC) is obtained by combining the error of each classifier. Since this process is hard to accomplish, selection-based strategy is adopted wherein the best executing classifier is automatically selected from the Ensemble of Classifiers (EoC) for a given example. In addition, it will, in general, have prevalent execution than the fusion-based strategy as one classifier in selection-based firmly overwhelms the other.

Having said that, the dynamic classifier under selection-based is utilized where it selects the classifier that best suits the present pattern, unlike static classifier, which chooses the best classifier for all the test patterns bringing about increased time for choice. In this approach, the spatial and spectral information is coordinated to improve the classification performance using dynamic classifier ensemble algorithm. However, the spatial and spectral information is utilized both in pre-preparing step and in assignation of labels for a pattern. In general, labelling enables to identify different entities in an image. For instance, although certain fruits belong to different classes, yet they seem to look similar. Categorizing such fruits into different classes would be a tedious task; but using labels it is convenient to quickly categorize them into their respective categories. The primary rationale utilized in allocating labels to a pattern is by using the spatial and spectral information to confirm whether the extent of pixels already named around a pattern surpasses certain limit. On the off chance that it does, the spatial and spectral information will be utilized to decide the name of the pattern otherwise only spectral information will be utilized.

The proposed dynamic classifier ensemble algorithm is structured as shown in figure 2.1

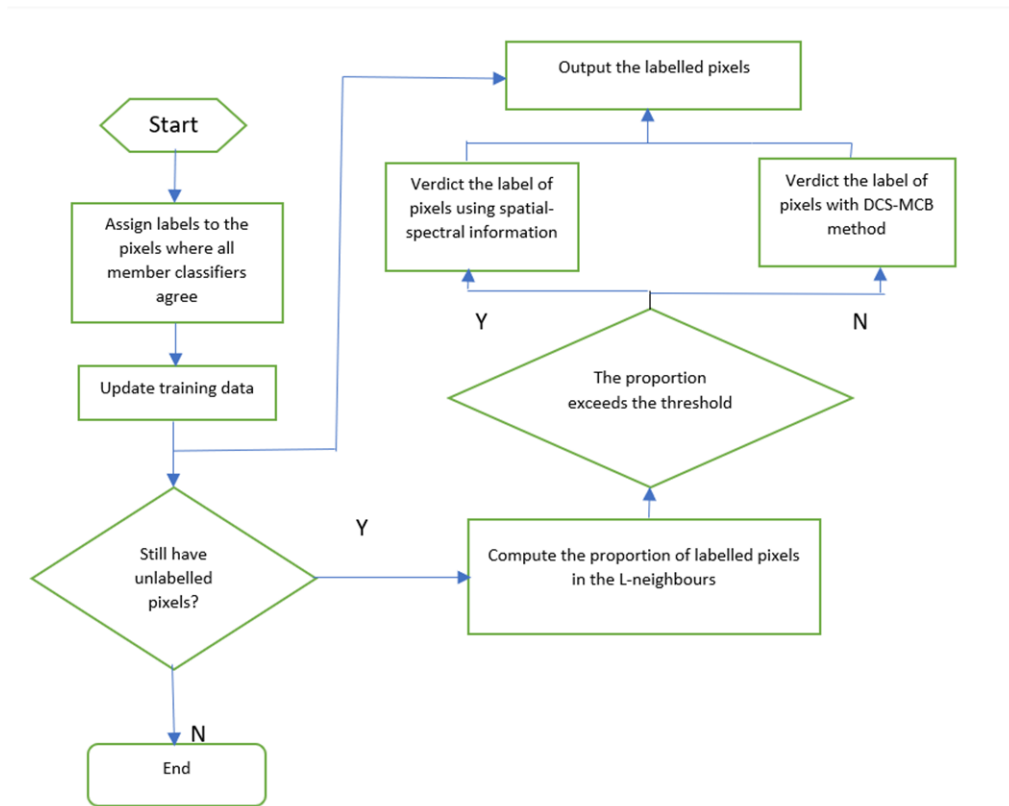


Figure 2.1 The proposed dynamic classifier ensemble algorithm

The algorithm proposed here overcomes major disadvantages found in different Distributed Control Systems (DCS) algorithms. Firstly, it defeats the utilization of just spectral information taken in classifying hyperspectral images instead considers spatial information too; particularly neighbouring pixel information of images, otherwise achieves a bottleneck if no other information is included.

Secondly, rather than utilizing just spectral information for assigning labels to the pattern, both spectral and spatial information is considered to get more noteworthy support comparatively (He *et al.* 2019). Since not all neighborhoods give important spatial and spectral information to recognize a pattern, hence already classified neighbourhood pixels are considered and consequently, pre-processing step is required in this methodology. In this process, the labels are straightforwardly allocated to the pixels agreed by the Ensemble of

Classifiers and these labels would not remain constant after training the data as the errors obtained during the training phase could affect the classification accuracy. Therefore, the pixels are labelled directly while they are involved in training the data. For the classified pixels, the labelled pixels is determined around the unlabelled pixels. On the off chance the number of labelled pixels surpasses the limit (for example, set to 70%) at that point it implies the pixel's spatial information could assign a label for the pixel. To assess the performance of this approach, an experiment was conducted by comparing the proposed strategy with the DCS-LCA (Distributed Control System – Local Class Accuracy) algorithm and with the Support Vector Machine (SVM). At the end of the experiment, it was observed the z value was 1.78 for five percent level of significance showing significant execution comparatively.

2.2.2. Hyperspectral image classification using Iterative Support Vector Machine (ISVM)

As seen in the above section, an attempt was made to improve the classification performance by combining both spatial and spectral information. A similar method is proposed by the author (Zhong *et al.* 2018) with the aim to improve the classification performance using Iterative Support Vector Machine (ISVM) algorithm.

In this methodology, ISVM uses a Principal Component (PC) mainly to reduce many variables into a smaller set of variables to save the memory space during execution. This PC is combined with the input hyperspectral image to form a classification map. A filter named 'Gaussian' is applied on this map in order to retrieve the spatial data from the classified information on the map. The obtained filtered map is fed back to the first step where a hyperspectral image is being processed in the ongoing iteration. This creates a new cube of data to produce the next classification map by combining the input hyperspectral image in the next iteration. As the number of iterations increases, more the number of filtered maps is obtained resulting in better classification. Further, to stop the generation of classification maps a stopping rule is set; the iteration terminates when it exceeds a limited threshold set. This process is represented by the Figure 2.2 as shown below

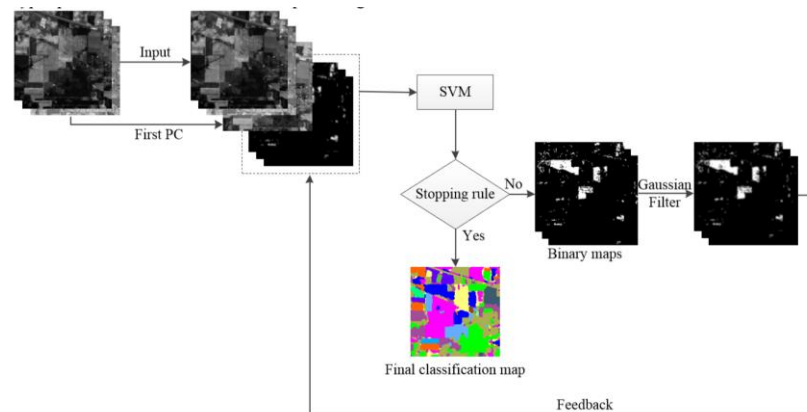


Figure 2.2 A flowchart of Iterative Support Vector Machine

To assess the performance of the proposed approach, an experiment was conducted and it was observed that Iterative Support Vector Machine (ISVM) performed better by accomplishing more than 0.96 exactness rate by and large when contrasted with other edge-preserving filter-based techniques which have demonstrated to have predominance in spatial-spectral hyperspectral classification strategies

2.2.3. Comparison of different Machine learning framework to obtain the best framework for image classification

A Machine learning algorithm is particularly suited for prediction. Since Machine Learning strategies are simpler to actualize as it learns the relationship between the predictor and response variables instead of directly starting with the data model, therefore, performs superior to the traditional approaches. Predictor variable is otherwise called as independent variables are normally related with changes in the response.

Having said that, the author, Loussaief and Abdelkrim (2016) proposed a plan to find the best machine learning framework which can work efficiently in classifying images by training models on different classifier algorithms. Here, the Bag of Features (BoF) system and Speed Up Robust Features (SURF) are utilized to concentrate and distinguish features from images. Bag of Features system initially extracts the features from the images and then learns to fit the features that matches a certain criterion using visual vocabulary (contains a list of images

to portray certain concepts or ideas). Further, it quantizes the features with the help of visual vocabulary to represent images. Whereas, SURF is used to distinguish features from images. To assess the performance, numerous supervised learning algorithms namely SVM, k-closest neighbours, and Boosted Regression trees are explored. The dataset images are encoded using Bag of Features (BoF) vector as shown in the figure 2.3

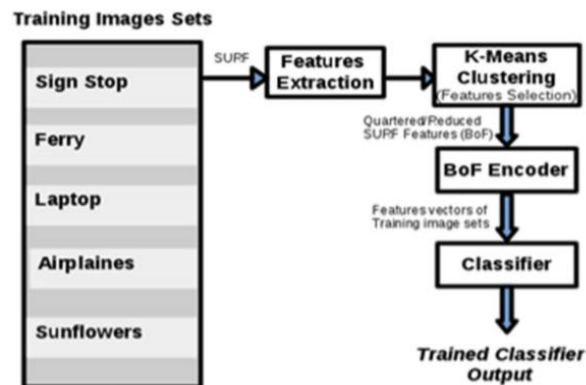


Figure 2.3 Image classification process

Firstly, descriptors from the training images are extracted and to assemble these descriptors and to develop the N visual words, K-means clustering is utilized. A cluster of these descriptors is created using 'Euclidean distance' metric. Each time a descriptor is added into the cluster, the number of inhabitants is created by producing a histogram of counts. Each image is structured with a histogram of vector at the end of the procedure.

To evaluate the performance, the author conducted an experiment wherein SURF extractor and Linear SVM classifier was utilized. Towards the finish of the experiment it was observed the classifier's accuracy was inversely proportional to the number of classifications in the dataset as shown in the figure 2.4

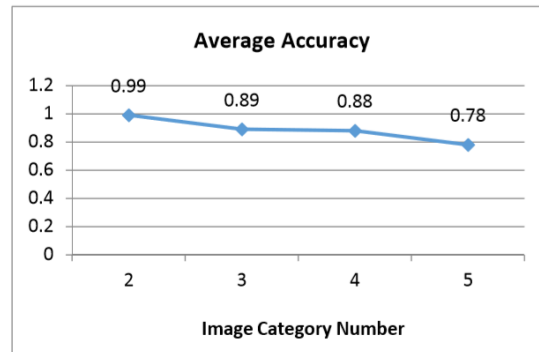


Figure 2.4 Average accuracy of the classifier

Further, by utilizing SURF feature extractor, fluctuating classifier algorithms like Support Vector Machine, K- nearest neighbour algorithm and ensemble classifier algorithm were utilized upon which models were assessed to know the classifier best suitable in image classification. At the end of the training learner evaluation experiment, it was discovered that Support Vector Machine algorithm performed better in image classification when compared with other learning algorithms as shown in the histogram.

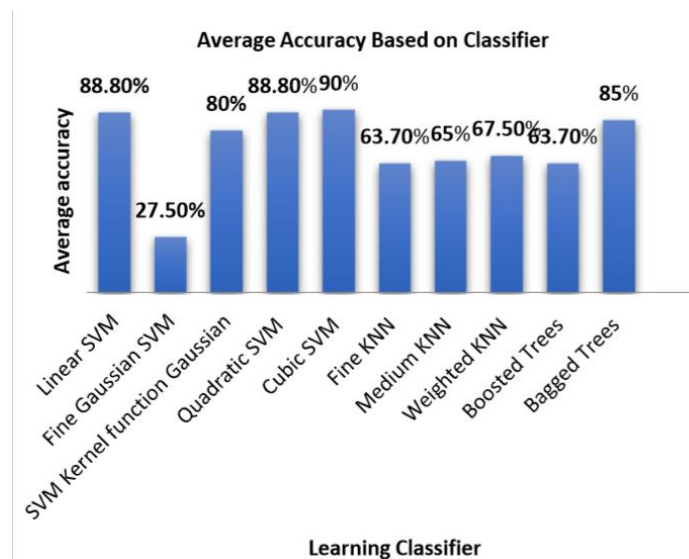


Figure 2.5 Average accuracy based on classifier

Among the different SVM algorithms, the cubic SVM stood first achieving 90% accuracy. Whereas, weighted KNN performed better among its peers achieving 67.50%. Under the ensemble classifiers, Bagged trees performed best with 85% accuracy when compared with the Boosted trees classifier.

Overall, in this methodology, Bag of Features and SURF technique were utilized to extricate features and descriptions from images. By performing relative analyses on Caltech 101 image dataset, it was discovered that SVM training classifier performed best when compared with other learning algorithms.

2.2.4. Use of Active learning technique based on Markov Random fields for classification of hyperspectral images

Another method was proposed by the author Sun et al (2015) to overcome the presence of fewer training samples with the objective to improve the classification accuracy. In this methodology, to enlarge the training set, the most important samples were selected from the unlabelled set and are manually labelled. Active Learning method is adopted here wherein manually labelled samples are fed into the classifier in order to retrain it with the additional labels. This method prevented unnecessary and redundant labelling of non-useful samples. Thereby, diminished the labelling expense and time.

It is observed that the spatial information of an image improves the classification accuracy in image classification process. In general, Markov Random Field (MRF) is used as a post-processing step where it utilizes the neighbourhood spatial information to refine the classification. Since it is observed the spatial information of an image improves the classification accuracy in image classification process, the idea proposed by the author Sun et al (2015) emphasizes on the same; the results of MRF has been combined with the Active Learning (AL) framework (considers the results obtained from the probabilistic pixel-wise classifier). Integrating MRF with the classifier is observed to provide better classification accuracy rather than using the classifier alone. However, in conventional AL heuristics, the samples were taken from the output of the probabilistic classifier; the conventional AL was likely to be overconfident on the predictions made even when they seem to be incorrect. Hence, to overcome this issue, a new AL heuristic classifier was designed. Since both MRF and AL is combined, the labels anticipated by the probabilistic classifier in AL and in MRF post-processing step were inconsistent and less certain about the correct classification. The labels

having the highest-class probability was added into the training set and is removed from the unlabelled set.

In order to overcome this problem, Markov Random Field and the pixel-wise classifiers were combined to design a new AL heuristic (Suganuma *et al.* 2017). Overall, the proposed AL technique was seen to give better execution during the AL procedure than the previous strategies.

2.2.5. Use of Multiscale spatial texture features to classify hyperspectral images

Spatial data in the field of image classification has shown significant commitments as of late. Linear Binary Pattern (LBP) descriptor is used to retrieve spatial information from the input image. Since it is not designed to extract both the structural and textural information, the author Sidike et al (2016) proposed a method to utilize Complete Linear Binary Pattern (CLBP) wherein multiscale scheme was incorporated to improve better feature extraction. In the proposed methodology, the LBP code image were generated for individual bands in the hyperspectral image. Histogram of LBP for every pixel were discovered near to its relating neighborhood to locate the spatial attributes of the pixel. Regardless, this technique did not consider the textural and spatial features of pictures. Subsequently, to conquer this issue, multiscale conspire that structures the total LBP at different resolution has been proposed by the author. Further, to improve the grouping precision multiscale analysis is performed on both CLBP and on LBP. To gain rich textural data, different signs and magnitudes of CLBP were combined. The textural information of an image is represented by LBP by considering the sign of the neighbourhood pixel. Whereas, CLBP utilized its center pixel along with the sign and magnitude of the neighbourhood pixels to represent the patterns. A single scale present in both LBP and CLBP are not capable of capturing dominant features and hence a multiscale computation is used to extract the most discriminant features at all resolutions.

In CLBP, the number of neighbors are kept constant while the scale values were changed to find the best results. To assess the performance, an experiment was conducted, and it was observed the performance of CLBP was better when compared with LBP since the edge and

corner of an image were retrieved in detail by the magnitude component enhancing the spatial pattern.

2.2.6. Use of Projection-based random forests for feature independent classification of hyperspectral images

In the method proposed by Zhong et al (2018), filter bank was utilized to filter each spectral band. Since the filters were applied for every spectral band, the computation load increased tremendously. Also, certain features that were significant for the classification task were miscalculated. Hence to overcome these limitations, Hansch and Hellwic (2015) has proposed a method to use Projection-Based random forests wherein meaningful spectral bands are automatically selected by a classifier, directly working on the data itself. This technique totally excludes the utilization of feature extraction as well as any pre-preparing that incorporates manual expulsion of noisy band with the aim to reduce the computation load. Since ProB-RFs i.e., a Predict Random forest are a supervised learning scheme, the class label of each data point is known. In the spatial setting, each data point is three dimensional in nature and belongs to the spectral band followed by the spatial dimension.

In the first stage, spatial projection is implemented on a spectral band and an operator is chosen according to the type of the projection. This operator is implemented in more than one region of the image; the outputs obtained from all the regions are compared with one another. Based on the test results, the propagation of a data node is carried out either to the left or the right node of the tree. Many test functions are carried out by a node to get a split that best aligns with the classification task. From this arrangement of potential splits, the best is chosen based on the impurities from the parent to child node.

To assess the performance, an experiment was conducted wherein there existed no pre-processing step. Instead, the classifier was directly implemented on the bands. Also, automatic selection of significant bands was made by the classifier itself. After carrying out the experiment it was discovered that unlike PCA, did not manually reject noise or meaningless bands before processing, rather implemented the classification system to the original information and still accomplished to obtain the performance of $\kappa = 0.85 \pm 0.03$.

Overall, the proposed technique here legitimately works on the hyperspectral images and results are efficient yet precise in processing of images. The primary advantage found in this methodology unlike the approaches seen so far is, there is no pre-preparing whether manual or programmed is required and this technique reduces the cost and the computation load. Also, the spatial context of images is considered by spatial projections.

2.2.7. Image classification using Visual saliency model

To overcome the long running time of an algorithm and lower grouping precision, the author Tang et al (2016) has proposed a method to utilize visual saliency model. In this methodology, corner detection and region detection operators are utilized to extract the feature from an image. Specific techniques namely Moravec and Harris's corner are utilized to identify corner. Then again, the Difference of Gaussian (DoG) operator is utilized to recognize regions from an image. The visual models namely Hypercomplex Fourier Transform, Phase Spectrum of Quaternion Fourier Transform, Phase Spectrum of Fourier Transform, Spectral Residual are utilized to deal with the significant area from the image, and to eliminate the repetitive background data with the fundamental goal to decrease time complexity and to improve the effectiveness of image processing. To compute the rate of classification, the following equation is used

$$\frac{\text{Rate} = 1 - \text{Error (1)} + \text{Error (2)}}{\text{TOTAL number}} * 100\%$$

Here, Error (1) indicates the misclassification occurred indifferent classes, Error (2) represents misclassification occurred in the same class, Total number represents the number of classes present. To assess the performance of the models used, an experiment was conducted using animal dataset. At the end of the experiment it was observed that, few images were misclassified under different category.

2.2.8. Classifying images using Manifold learning based non-linearity dimensionality reduction

The two most significant measurements in image classification are precision and speed. Despite numerous factors that influence precision and speed, keeping harmony between them ends up being a challenge in the image classification process. With this concern, (Faaeq *et al.* 2018) has proposed a methodology wherein manifold learning strategies. Speed and precision are compared between these strategies and other classification algorithms. Here four diverse manifold learning strategies are used namely multi-dimensionality scaling, Spectral implanting, Local straight inserting strategy and Isomap implanting. Each strategy is explained as follows. Multi-dimensional scaling: Similar objects are likely to possess short distance between themselves in an image than the dissimilar objects. Here, objects refer to color, shape or any other attributes existing in an image. In general, multidimensional scaling is the representation of dissimilarities between objects. This representation is scaled based on how far or near the objects exist in an image.

Locally linear embedding: It is an unsupervised learning procedure for dimensionality reduction utilized in diminishing the dimensions while attempting to keep the features of the first non-linear element structure.

Isomap: It is a method used to embed high dimensionality reduction into low dimensionality reduction. It is profoundly proficient and can be applied for the most data parts.

Spectral embedding: This is a strategy used to deal with the ascertaining linear embedding, it utilizes Laplacian Eigenmaps. The data having low dimension is observed and are utilized by the Eigenmaps. In the classification stage, the machine learning algorithms namely K-nearest neighbor, Support Vector Machine, Random forests and Logistic regression are used. To assess the performance of the proposed approach, the outcomes were compared between manifold strategies and the dimensionality reduction algorithms. Firstly, the performance of the classification algorithms was observed, and it was found that Logistic regression produced a significant outcome securing accuracy of 98%. Whereas, the Support vector machine and KNN algorithms were found to take the second place after logistic regression and the random

forest was found to have gotten the most minimal precision score in terms of speed yet was found to perform best in terms of speed compared with different algorithms.

Secondly, the outcomes subsequent to applying manifold learning with dimensionality reduction algorithm were observed. At first, results subsequent to applying one of the manifold algorithms i.e., Multi-dimensional reduction algorithms were observed, and it was discovered that SVM performed best in terms of classification precision with 0.936 while took the last position in terms of speed. Then again, kNN performed best in speed taking 0.471 seconds and took the subsequent position in terms of precision with 0.933. Random forest was found to have the least exactness comparatively and took third place in terms of speed with 1.0 seconds. Whereas, Logistic regression relapse came just a short of the win in speed securing 1.05 seconds but stood third position in classification accuracy having 91%.

Similarly, the results were observed by applying Isomap algorithm as follows. The quickest and the most exact of all four-classification algorithm was found to be KNN having 0.955 classification precision and 0.434 seconds of execution time. The slowest of all four algorithms were found to be SVM taking 2.654 seconds with 0.94 classification precision and stands third in accuracy comparatively. Whereas, Logistic regression stood last in obtaining better classification accuracy and positioned third in execution speed securing 1.04 seconds. The random forest algorithms stand next in execution speed having 0.98 seconds also, ranked second for obtaining 94% classification precision.

In applying Local Linear Embedding algorithm, the following results were obtained. Random forests algorithm was found to have gotten 73% classification accuracy and was termed to be best when compared with its counterparts. It stood in the third place for securing 0.89 seconds. The algorithm that executed fast was found to be KNN having 0.48 seconds. On the other hand, Logistic regression obtained 32% classification accuracy and was ranked the last among other algorithms but stood secured second position in obtaining 0.62 seconds execution speed. Whereas, the Support Vector Machine (SVM) took 6.87 seconds to complete the execution and therefore it was termed to be the slowest among the other. Also, SVM was ranked second for obtaining the least classification rate after Logistic regression.

Lastly, spectral embedding algorithm was applied, and the following results were observed. In terms of obtaining significant classification and accuracy, KNN was ranked the first among other algorithms having obtained 88% accuracy and 0.45 seconds execution period. Random forests stood next for obtaining 87% accuracy and third position for obtaining 0.90. Logistic regression was ranked the least for obtaining classification accuracy comparatively. Whereas, SVM was again ranked last for obtaining seconds to execute.

Overall, it was discovered that the classification speed would be high without the use of manifold learning with dimensionality reduction algorithm. Through performing experiments, it was seen that the best outcomes were gotten after utilization of Isomap manifold algorithm in combination with kNN classification algorithm when compared with different algorithms.

2.3. Use of Convolutional neural network for image classification

Although several algorithms were used to classify images as outlined in the previous section, yet these methods manually extracted features from images which resulted in losing spatial interaction between pixels; obtaining lower classification accuracy. Hence to automatically extract features from the input image, convolutional neural network is utilized wherein the image is down sampled first by utilizing the pixel information and then the prediction is carried out at the end. This principle behind Convolutional Neural Network surpassed the existing limitation. Different methods have been proposed by authors to classify images using CNN algorithm as discussed in the following section.

2.3.1. Addition of middle output layer for inference to accelerate image classification

To accelerate the image classification the author Lee and Lee (2016) have proposed a method. Accomplishing higher accuracy by reducing the computation budget for the application where time is constraint is essential, there have been attempts made in the field of deep learning to reduce a large neural network to have fewer parameters to facilitate the objective. Having said that, the author here attempts to finish the image classification by adding an output layer in the middle of the convolutional neural network and when the top 1 confidence surpasses

a pre-characterized confidence threshold at the middle output layer, then image classification completes as early as possible.

It is found that frameworks having a wide and profound deep network involves an extremely large number of parameters and mostly consume more time as they need to lead an enormous number of parameters. Apparently, as the number of parameters increases the more memory space is required. With this concern, the author adds one middle output for inference as shown in the figure rather than only depending upon the last output layer. Here, for a given input image to be classified, initially the first output layer is analysed and if the certainty of rank 1 i.e., category of highest probability surpasses a pre-characterized limit i.e., 90% then the inference stops. Otherwise, the inference continues being finished by the last output layer. The fundamental goal of this technique as said before is to classify images as early as possible to reduce the computational burden at inference time.

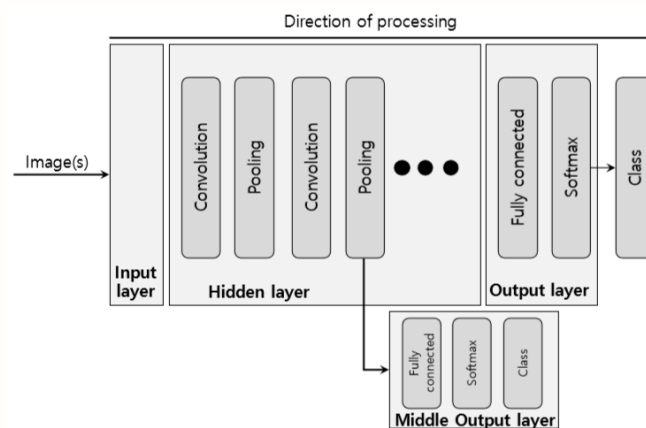


Figure 2.6 Illustration of proposed approach

The input image is fed into the network, firstly the certainty of the image belonging to a category is analysed by the initial output layer and if it surpasses 90% i.e., highest probability then the inference halts. Otherwise, the inference waits to be finished by the last output layer. The fundamental goal of this technique is to complete the classification as early as possible and to reduce the computation overhead. If the classification is inaccurate, then completing the inference at the middle output layer at the earliest would be inappropriate resulting in lower classification precision. Hence to overcome this limitation, the middle and final output

layers are trained by adopting teacher-student model. As the name suggests, this training model trains the network layer-wise like a teacher and if there are layers left untrained, they train themselves like students following the teacher's behaviour. Since all the layers are trained there exists less probability for the misclassification to occur and hence error rate is reduced resulting in high precision rate.

There are few demerits outlined in this methodology and they are as per the following:

- In the instance of classifying complex images, the middle output layer considered as final output layer is limited to classify only simple images like single item, high contrast, high intensity, etc.
- This process is targeted only in the environments where the computation budget is limited. For example, like in mobile phones, low end door. Reducing computational budget might not have a recognizable effect on the performance in places where powerful hardware equipment types like CPU, GPU is committed for image classification.
- The middle output layer may not be prepared to have enough certainty to complete the inference in the case of complex images. In this situation, the final output layer must carry out the classification. In this case, the inference time of the images will be slightly higher than usual. Also, causes an additional overhead by the inference time incurred at the middle output layer.

Even though there exist few demerits that are laid out in this methodology with respect to computation overhead and classifying complex images, yet this methodology demonstrates the possibility of inference completion at the middle layer. Likewise, the experiment results discussed in the next section affirm the possibility of selective inference.

2.3.2. Use of Caffe deep learning framework for image classification

Although a strategy by Lee and Lee (2016) was proposed to increase speed and precision in image classification yet to improve the accuracy, certain adaptations and developments are made by the author (Cengil et al 2017). In this proposed method, a dataset of images having 1000 classes have been utilized and Caffe library is adopted to carry out the classification.

In this methodology, the alexnet model is being utilized and its architecture is as shown in the figure.

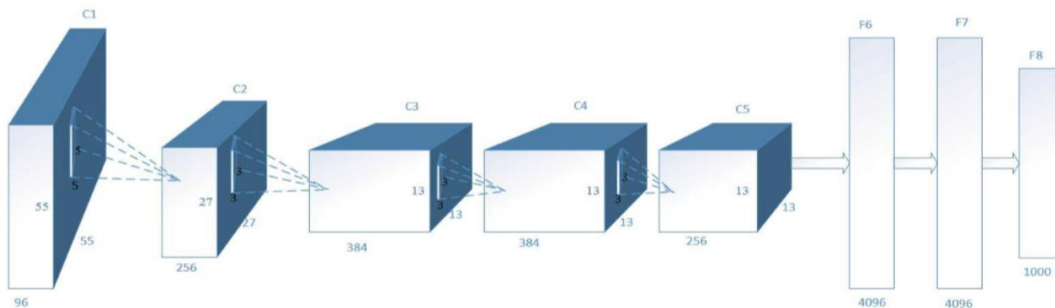


Figure 2.7 Network structure of deep learning

The model comprises of 11 layers, the first two layers are the convolution and max-pooling layers; the third layer is the normalization; the fourth and the fifth layers are again the convolution layers; the sixth layer is repeated to be the max-pooling layer; the seventh is the fully connected layer and the last layer is the SoftMax layer. Since 1000 images are being classified in this method, there exists the same number of neurons in the last layer of the network. On One hand, Caffe library is appropriate for research use as a result of its seclusion property it is also used here to train the neural system. is secluded, it is appropriate for research use and to train the neural system. On the other hand, to classify large number of images high processing power is essential. Therefore, CPU or GPU is used in this method because of its high processing capacity. After conducting an experiment it was observed that although high accuracy was gotten in discovering images of similar and different classes yet images of certain classes in the dataset were discovered erroneous and furthermore the

labels created after training and the classes of test information were not actually equivalent. Hence, precise outcomes were not obtained

2.3.3. Use of Inception-v3 for classifying flowers into different categories

Despite many proposed thoughts to acquire high precision in grouping images, there existed a challenge where features from input images were manually chosen for classification yet fail to get high precision in traditional techniques. With this concern, the author, (Xiaoling Xia *et al.* 2017) has adopted TensorFlow platform; a pre-trained Inception-v3 model and transfer learning technology has been utilized to retrain images to improve the classification accuracy. Deep learning programs requires frameworks that can ease the classification. Since TensorFlow framework is highly available, adaptable and has the improved effectiveness to carry out machine learning computation, it is implemented in the deep learning programs to carry out the classification operations. There are many pre-trained models built to classify images, one such model is Inception-v3 released by Google on TensorFlow's platform. Since it takes days to train this model on a PC having lower configuration, transfer learning technology is applied on the model wherein only the last layer is retrained to accept new input images (Nanni *et al.* 2019). To assess the performance, an experiment was conducted to classify different flowers into their respective categories. At the end of the experiment it was observed the proposed method obtained 95 % classification accuracy and was termed to be a good approach in classifying images.

2.3.4. Use of Multi-channel convolutional neural network to extract invariant feature for image classification

Images are influenced by certain factors like illumination, geometric change and rotations. These factors stand a major challenge while classifying images. Although Scale Invariant Feature Extraction (SIFT) algorithm is robust to illumination and rotation, it cannot be used to represent the whole image as it is a local feature descriptor. One of a building blocks of a Convolutional Neural Network called as 'Max-Pooling' is used to reduce the spatial dimensions of an image with the aim to minimize the computation load of a network. Another operator called TI-pooling that enables CNN to be transformation-invariant, utilizes Max-pooling to get different angles of the input sample in order to work efficiently with all the

rotated images fed as input. Considering this principle, The author Mei et al (2017) proposed a method wherein invariant features from an image was extracted for classification using Multi-channel convolutional neural network. In this approach, two sets of CNN blocks are created in the network where each block helps in extracting features that seem to be rotation-invariant in nature. Firstly, every sample image present in the training dataset is rotated to conduct data augmentation. Secondly, a pair of sample images are taken from the same category and are used to train multi-channel Convolutional Network(mCNN) to detect the uniqueness in the homogenous images (For instance, illumination in images, geometric change). Thirdly, to conduct the test one sample image is fed as input into mCNN to extract invariant features. To evaluate the performance, the two hand-written datasets were used. The figure 2.8 shows one of the hand-written sample dataset



Figure 2.8 Sample in dataset

During the analysis, TI-pooling was taken into consideration for the purpose of comparison and at the end of the analysis, it was observed that the proposed mCNN outperformed TI-pooling as shown in the graph 2.9. The accuracy was 2% more than TI-pooling.

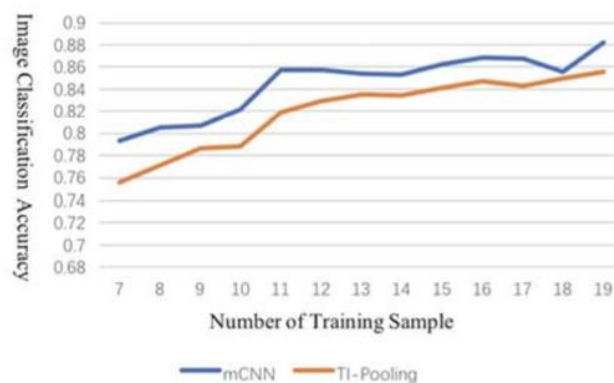


Figure 2.9 Image classification accuracy contrast chart

Also, different channels were set and the analysis were performed as shown in the figure 2.10.

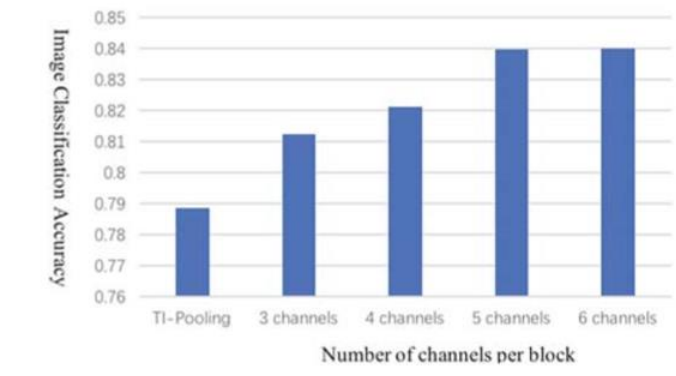


Figure 2.10 Multi-channel accuracy comparison with TI-pooling

It was observed that, as the number of channels increased the classification accuracy raised. The accuracy raised until channel 5 and remained constant after that. Overall, the proposed approach performed better in classifying rotation invariant images and the results showed there was 2% increase in the classification accuracy when compared with TI-pooling

2.4. Conclusion

As discussed so far, several methods were proposed to classify hyperspectral images using different algorithm approach. However, each time features from the images were extracted and fed into the classification algorithm manually. This increased the computation time whilst obtaining output. With this concern, the Convolutional Neural Network (CNN) was introduced to extract features wherein adjacent pixel information were extracted to effectively down sample the image initially and then to use a prediction layer at the end with the main objective of overcoming the challenge of losing spatial interaction between pixels while using an algorithm with the pixel vector. On one hand, method to increase accuracy, speed, and to overcome computation overhead was proposed, and better accuracy percentage was

obtained compared to traditional methods proposed earlier. Besides, to develop more effective model for image classification and to see if the accuracy percentage could still be increased, the implementation to classify images is carried out on Google Cloud Platform instead of executing on local machines that has already been executed as discussed so far. Hence, detail information regarding executing the image classification on Google Cloud Platform with the specific hardware and software requirements is discussed in the next section i.e., in Chapter 3.

3. Design and Implementation

3.1. Introduction

As discussed in the previous chapter, there are several approaches proposed to improve accuracy and speed and to considerably reduce the computation overhead while classifying images into its respective categories. Although there were many methodologies used to improve the overall classification performance yet there were few demerits that arose regarding computing power; as all the proposed methods used local machines to execute the entire image classification implementation. Apparently, which is limited and therefore it must be reduced as much as possible.

3.2. Dataset

The data set contains images of fruits and. These are available in kaggle.com at the link

https://www.kaggle.com/moltean/fruits#fruits-360_dataset.zip

The data set is broken into training dataset and test dataset.

Total number of pictures: 82213.

Training data: 61488 Images (74.79 %).

Test data: 20622 images (25.20 %).

Multi-fruits set size: 103 images (more than one fruit per image)

Number of classes: 120 (fruits).

Image size: 100x100 pixels.

Different varieties of the same fruit are stored as different classes. (Example Apple Red Delicious and Apple Red 1).

3.3. Technology and computing environment

For this work, there were three choices. 1. To use a standalone machine. 2. Use a distributed set of nodes in a data centre. 3. Use a cloud environment. While distributed systems offer great advantage in terms of performance and throughput, it is rigid and costly. Moreover, after the experiment is completed, these machines will have to be repurposed. The cloud option gives an elegant solution. It combines multiple advantages like high throughput, load balancing, auto scalability and one must pay per use, and it is very competitive. Another

reason for choosing cloud is the efficient creation of compute engine including GPU which is essential for deep learning.

Additionally, cloud comes with different platforms and APIs. There are three cloud offerings. AWS, Google Cloud Platform and Microsoft Azure. The cloud that is adopted for this work is Google Cloud Platform.

Advantages of cloud compared to on-premises systems.

- Lower overall cost. We pay for what we use only.
- Highly Scalable.
- Easy Access and flexible.
- Cloud service provider uses user-friendly web-based interface to enable users make optimal use of the services.
- Reducing business risks
- Reduces maintenance cost
- Extremely fast provisioning and deployment.

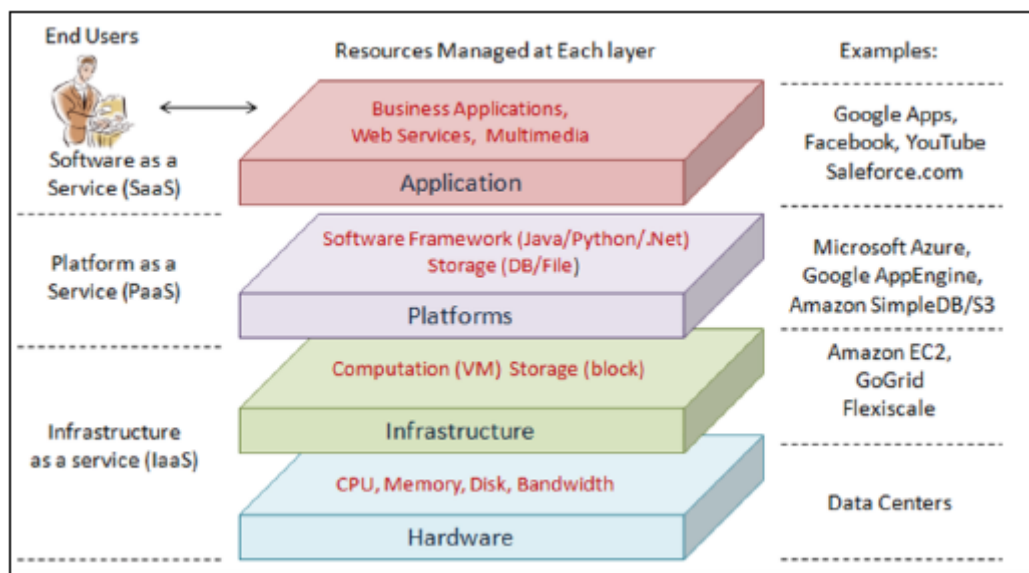


Figure 3.1 General architecture of cloud computing services

One of the most critical aspects of deep learning is the run time. If we do not use a GPU, the deep learning algorithms take a lot of time for the multiple epochs we run. Normal desktop systems and laptop systems do not come with GPU in general and have compatibility problems. Google Cloud Platform offers GPU on demand and it is very flexible.

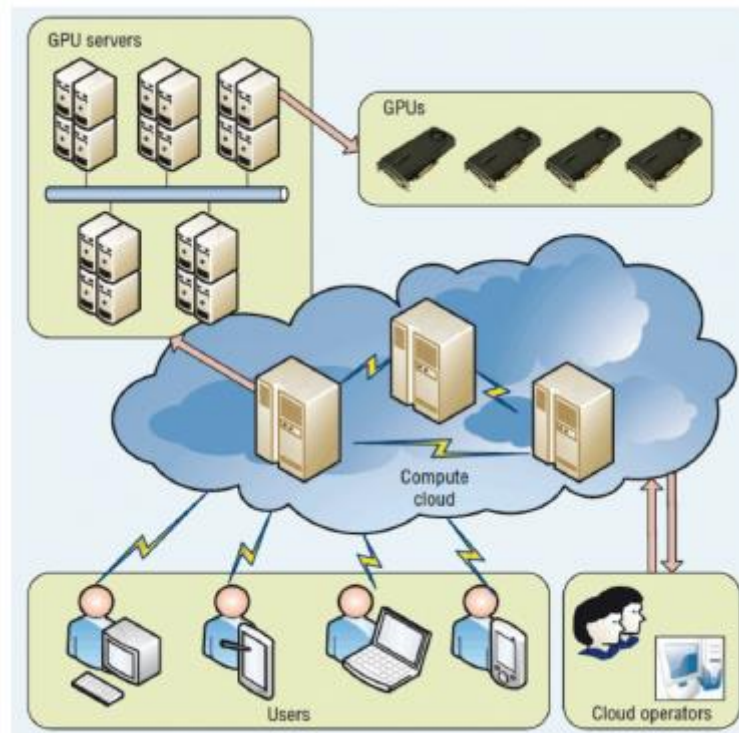


Figure 3.2 GPU based cloud computing framework

The screenshot of the steps for creating a Google Compute Engine is given below.

1. Creation of VM instance

Create an instance

Create a VM instance, select one of the options:

New VM instance

Create a single VM instance from scratch

New VM instance from template

Create a single VM instance from an existing template

Marketplace

Deploy a ready-to-go solution onto a VM instance

Name ⓘ
Name is permanent

instance-1

Region ⓘ
Region is permanent

us-central1 (Iowa)

Zone ⓘ
Zone is permanent

us-central1-a

Machine configuration ⓘ

Machine family

General-purpose Memory-optimized

Machine types for common workloads, optimized for cost and flexibility


Series

N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type

g1-small (1 vCPU, 1.7 GB memory)

	vCPU	Memory
	1 shared core	1.7 GB


✓ CPU platform and GPU

Figure 3.3 Instance creation

2. Creating a boot set up

Container ?
☐ Deploy a container image to this VM instance. [Learn more](#)

Boot disk ?

 New 10 GB standard persistent disk
Image
Debian GNU/Linux 10 (buster) Change

Identity and API access ?

Service account ?

Compute Engine default service account ▼

Access scopes ?

☒ Allow default access
☐ Allow full access to all Cloud APIs
☐ Set access for each API

Firewall ?
Add tags and firewall rules to allow specific network traffic from the Internet

☒ Allow HTTP traffic
☒ Allow HTTPS traffic

✕ [Management, security, disks, networking, sole tenancy](#)

Figure 3.4 Creation of Boot setup

Identity and API access ?

Service account ?

Compute Engine default service account ▼

Access scopes ?

- ☒ Allow default access
- ☐ Allow full access to all Cloud APIs
- ☐ Set access for each API

Firewall ?

Add tags and firewall rules to allow specific network traffic from the Internet

- ☒ Allow HTTP traffic
- ☒ Allow HTTPS traffic

⌵ Management, security, disks, networking, sole tenancy

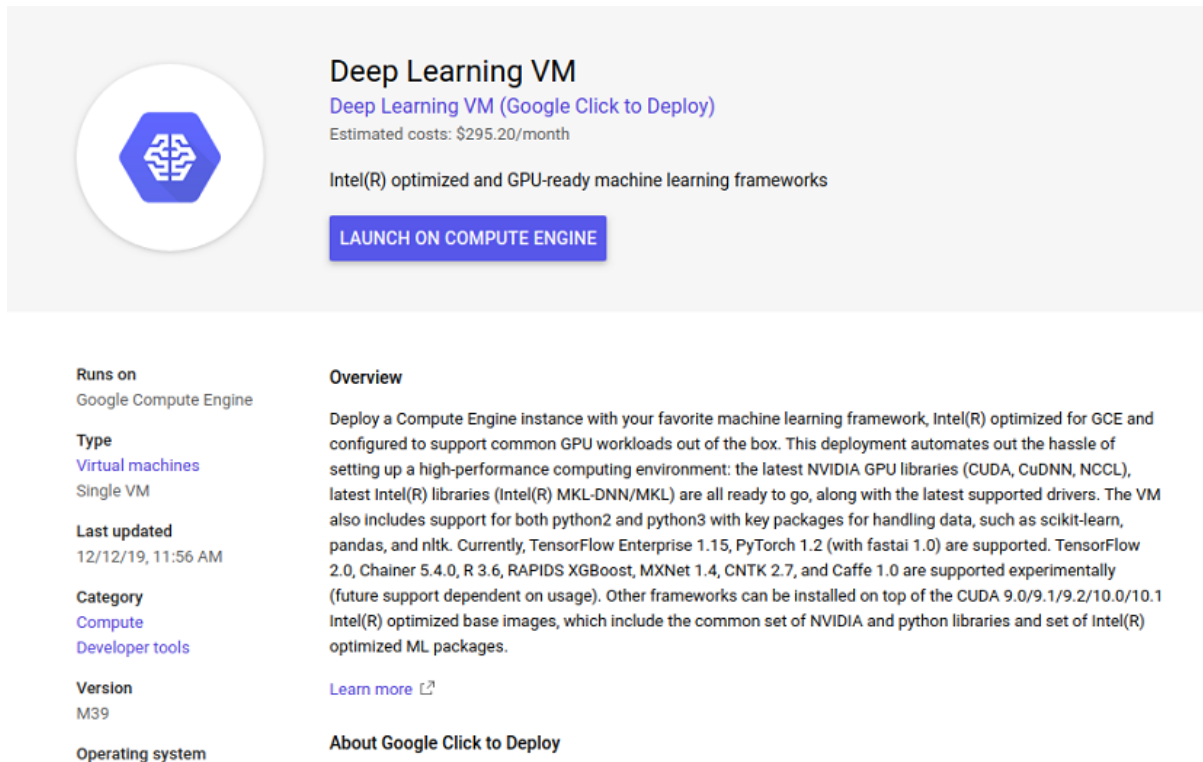
Your free trial credit will be used for this VM instance. [GCP Free Tier](#) ↗

Create Cancel

Equivalent [REST](#) or [command line](#)

Figure 3.5 Creation of compute engine

3. Deep learning VM in Google Cloud



Deep Learning VM
[Deep Learning VM \(Google Click to Deploy\)](#)
Estimated costs: \$295.20/month

Intel(R) optimized and GPU-ready machine learning frameworks

[LAUNCH ON COMPUTE ENGINE](#)

Runs on
Google Compute Engine

Type
[Virtual machines](#)
Single VM

Last updated
12/12/19, 11:56 AM

Category
[Compute](#)
[Developer tools](#)

Version
M39

Operating system

Overview

Deploy a Compute Engine instance with your favorite machine learning framework, Intel(R) optimized for GCE and configured to support common GPU workloads out of the box. This deployment automates out the hassle of setting up a high-performance computing environment: the latest NVIDIA GPU libraries (CUDA, CuDNN, NCCL), latest Intel(R) libraries (Intel(R) MKL-DNN/MKL) are all ready to go, along with the latest supported drivers. The VM also includes support for both python2 and python3 with key packages for handling data, such as scikit-learn, pandas, and nltk. Currently, TensorFlow Enterprise 1.15, PyTorch 1.2 (with fastai 1.0) are supported. TensorFlow 2.0, Chainer 5.4.0, R 3.6, RAPIDS XGBoost, MXNet 1.4, CNTK 2.7, and Caffe 1.0 are supported experimentally (future support dependent on usage). Other frameworks can be installed on top of the CUDA 9.0/9.1/9.2/10.0/10.1 Intel(R) optimized base images, which include the common set of NVIDIA and python libraries and set of Intel(R) optimized ML packages.

[Learn more](#)

[About Google Click to Deploy](#)

Figure 3.6 Deep learning VM

3.4. GCP environment setting for deep learning

The environment is setup on GCP by downloading Anaconda on TensorFlow platform to simplify the deployment in deep learning. The fruit images are copied from the bucket in GCP into a new storage created in the VM instance. Further, the CNN python file is executed as shown in the figure 3.8

```
sudo apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1  
libxcomposite1 libasound2 libxi6 libxtst6  
  
cd ~  
  
mkdir Downloads  
  
wget /home/sushmitha/Downloads "https://repo.anaconda.com/archive/Anaconda3-  
2019.10-Linux-x86_64.sh"  
  
sha256sum /home/sushmitha/Anaconda3-2019.10-Linux-x86_64.sh  
  
cp Anaconda3-2019.10-Linux-x86_64.sh Downloads/  
  
cd Downloads/  
  
bash ~/Downloads/Anaconda3-2019.10-Linux-x86_64.sh  
  
source ~/.bashrc  
  
conda init
```

Figure 3.7 GCP environment setup 1

```
python
anaconda-navigator
conda create -n tensorflow_env tensorflow
conda activate tensorflow_env
gsutil -m cp -r fruit-bin gs://fruit_360
gcloud init
gsutil -m cp -r fruit-bin gs://fruit_360
gsutil -m cp -r fruit_360 gs://fruit-bin
gsutil cp gs://fruit-bin /home/sushmitha/fruit_360
gsutil cp -r gs://fruit-bin /home/sushmitha/fruit_360
gsutil help options
ls fruit_360/
conda activate tensorflow_env
python -W ignore /home/sushmitha/dl2.py
conda install -c menpo opencv=3
python -W ignore /home/sushmitha/dl2.py
pip3 install -U scikit-learn scipy matplotlib
apt install python3-pip
sudo apt install python3-pip
pip install -U scikit-learn scipy matplotlib
python -W ignore /home/sushmitha/dl2.py
pip install keras
python -W ignore /home/sushmitha/dl2.py
```

Figure 3.8 GCP environment setup 2

3.5. Algorithm for classification

There are different techniques available for classification of entities. Deep learning using convolutional neural networks is widely adapted due to its accuracy. Convolutional neural network consists of many layers. Generally, a CNN model contains convolutional layers, pooling layers, ReLU layers, fully connected layers and loss layers.

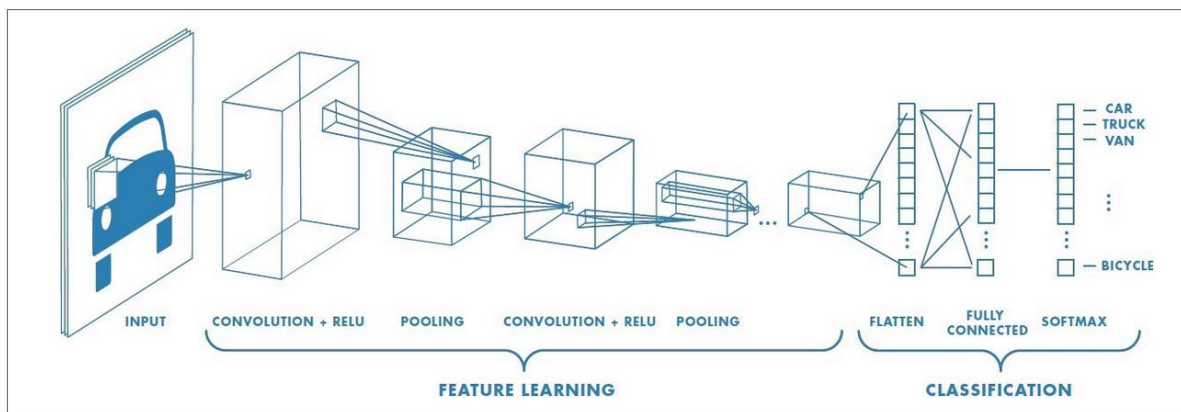


Figure 3.9 A typical Convolutional Neural Network

In a typical CNN system, the first layer is a convolutional layer. After this we have a Rectified Linear Unit (ReLU) layer. Then it is followed by a Pooling layer. After this again one or more convolutional layers and finally it is followed by one or more fully connected layer.

4. Results

4.1. Introduction

As discussed in the previous chapter, multiple fruit image dataset was used to classify them into different categories. The dataset has 50 classes and all the fruit images in each class is shot in different angles. To check if the Convolutional neural network could classify different hyperspectral fruit images regardless shot in varied angles, an experiment was conducted using CNN algorithm with the TensorFlow framework. This experiment was conducted on Google Cloud Platform to reduce the computation load and to execute the process in lesser interval of time. Metrics have been collected with respect to the run times and accuracy for the dataset used and epochs.

4.2. Observations- Runtime in GCP

The virtual instance on GCP was created and the CNN python file stored in home directory was executed as shown in the figure 4.1

```

Connected, host fingerprint: ssh-rsa 0 15:2D:B7:18:3A:C6:58:64:4E:77:87:3B:6A:B8
:35:AC:32:70:3A:7E:43:EB:AC:07:AA:9D:98:77:E9:FA:04:5B
Linux instance-1 4.9.0-11-amd64 #1 SMP Debian 4.9.189-3+deb9u2 (2019-11-11) x86_
64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Dec 29 21:13:03 2019 from 74.125.73.161
sushmithaagowdaa96@instance-1:~$ python -W ignore /home/sushmithaagowdaa96/dl10.py
Using TensorFlow backend.
24705
24705
Training set dims
Validation set dims
24266
24266
Test set dims
106
106
X_train
24705
X_cv
24266
y_train
24705
y_cv
24266
Number of classes
50
  
```

Figure 4.1 Execution of CNN python file

The different layers of CNN used is listed as per the figure and the model is created

```
Data normalized and hot encoded.
WARNING:tensorflow:From /home/sushmithaagowdaa96/.local/lib/python2.7/site-packages/keras/backend/tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /home/sushmithaagowdaa96/dl10.py:141: The name tf.train.RMSPropOptimizer is deprecated. Please use tf.compat.v1.train.RMSPropOptimizer instead.

Model: "sequential_1"

```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 100, 100, 32)	896
dropout_1 (Dropout)	(None, 100, 100, 32)	0
conv2d_2 (Conv2D)	(None, 100, 100, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 50, 50, 32)	0
flatten_1 (Flatten)	(None, 80000)	0
dense_1 (Dense)	(None, 512)	40960512
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 50)	25650

```

Total params: 40,996,306
Trainable params: 40,996,306
Non-trainable params: 0
None
CNN model created.

```

Figure 4.2 Created CNN model

In an epoch, 24705 samples were trained, and 24266 samples were validated. The total time taken to train the samples is observed to be 366 seconds; found to take 15 milliseconds to train each sample. Consequently, the total time taken to validate samples is observed to be 55 seconds; took 2 milliseconds to validate each sample. The figure 4.3 is a snapshot taken while observing the results.

```

WARNING:tensorflow:From /usr/local/lib/python2.7/dist-packages/tensorflow/python/training/rmsprop.py:119: calling __
init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
2020-01-02 13:17:29.142273: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that
this TensorFlow binary was not compiled to use: AVX2 FMA
2020-01-02 13:17:29.262389: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2300000000 Hz
2020-01-02 13:17:29.262868: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x5651418e8730 executing c
omputations on platform Host. Devices:
2020-01-02 13:17:29.262901: I tensorflow/compiler/xla/service/service.cc:175] StreamExecutor device (0): <undefin
ed>, <undefined>
2020-01-02 13:17:29.433192: W tensorflow/compiler/jit/mark_for_compilation_pass.cc:1412] (One-time warning): Not usi
ng XLA:CPU for cluster because envvar TF_XLA_FLAGS=--tf_xla_cpu_global_jit was not set. If you want XLA:CPU, either
set that envvar, or use experimental_jit_scope to enable XLA:CPU. To confirm that XLA is active, pass --vmodule=xl
a_compilation_cache=1 (as a proper command-line flag, not via TF_XLA_FLAGS) or set the envvar XLA_FLAGS=--xla_hlo_pr
ofile.
WARNING:tensorflow:From /home/sushmithaagowdada96/.local/lib/python2.7/site-packages/keras/backend/tensorflow_backend
.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
WARNING:tensorflow:From /home/sushmithaagowdada96/.local/lib/python2.7/site-packages/keras/backend/tensorflow_backend
.py:431: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.
WARNING:tensorflow:From /home/sushmithaagowdada96/.local/lib/python2.7/site-packages/keras/backend/tensorflow_backend
.py:438: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

Train on 24705 samples, validate on 24266 samples
Epoch 1/1
24705/24705 [=====] - 366s 15ms/step - loss: 1.6316 - accuracy: 0.5617 - val_loss: 0.1090 - val_accuracy: 0.9825
24266/24266 [=====] - 55s 2ms/step
Accuracy: 98.25%
Training Done
Evaluating test images
[[4.2004164e-08 6.0625747e-02 2.1247721e-09 ... 1.1237876e-01
 6.2208250e-07 2.2585967e-07]
[5.6023612e-09 5.5017772e-11 2.4905176e-09 ... 5.7269425e-09
 1.4717042e-05 2.2026889e-08]
[1.8572436e-04 2.4564692e-06 1.5764551e-06 ... 6.4744125e-07
 5.4455850e-05 9.0011250e-05]
...
[1.2208464e-08 7.9927456e-01 1.9668109e-09 ... 1.1550655e-04
 2.0625287e-06 4.1848543e-05]
[3.1525022e-03 2.5551737e-04 6.6998913e-03 ... 1.4683703e-04
 3.2331966e-02 5.6990068e-03]
[2.7809292e-02 3.1469619e-05 9.1653736e-03 ... 1.4030784e-05
 4.4644778e-03 4.3905354e-01]]
Predictions Done
sushmithaagowdada96@instance-1:~$

```

Figure 4.3 Snapshot of time taken to train and test CNN model

After training and validating step, the accuracy of the classification is observed to be 98.25%. The snapshot of the same has been taken as shown in the figure 4.4

```

Train on 24705 samples, validate on 24266 samples
Epoch 1/1
24705/24705 [=====] - 366s 15ms/step - loss: 1.6316 - accuracy: 0.5617 - val_loss: 0.1090 - val_accuracy: 0.9825
24266/24266 [=====] - 55s 2ms/step
Accuracy: 98.25%
Training Done
Evaluating test images
[[4.2004164e-08 6.0625747e-02 2.1247721e-09 ... 1.1237876e-01
 6.2208250e-07 2.2585967e-07]
[5.6023612e-09 5.5017772e-11 2.4905176e-09 ... 5.7269425e-09
 1.4717042e-05 2.2026889e-08]
[1.8572436e-04 2.4564692e-06 1.5764551e-06 ... 6.4744125e-07
 5.4455850e-05 9.0011250e-05]
...
[1.2208464e-08 7.9927456e-01 1.9668109e-09 ... 1.1550655e-04
 2.0625287e-06 4.1848543e-05]
[3.1525022e-03 2.5551737e-04 6.6998913e-03 ... 1.4683703e-04
 3.2331966e-02 5.6990068e-03]
[2.7809292e-02 3.1469619e-05 9.1653736e-03 ... 1.4030784e-05
 4.4644778e-03 4.3905354e-01]]
Predictions Done
sushmithaagowdada96@instance-1:~$

```

Figure 4.4 Classification accuracy obtained

According to the literature survey, all the existing approaches to classify images executed Convolutional Neural Network model on local machines. This resulted in two major drawbacks the first one was high computation load and the second one, long running time as it took days to train and test the model. In order to overcome these challenges, cloud platform is utilized to execute CNN model. Since 300 credits were given for free trial to use Google Cloud Platform, an attempt was made to execute CNN model on this platform to observe if it could overcome the limitations discussed.

To execute CNN on Personal machines, the machine should have minimum specifications as follows. The preferred RAM size is 32GB; minimum 16GB RAM is expected and 2TB Hard Disk

Drive is expected. Also, adaption of GPU is essential to run parallel operation in deep learning tasks (Gokkulnath TS 2017). The PC was of 8GB RAM and adopting the required specifications was found to incur high cost therefore, CNN model on local machine could not be executed to compare the runtime and classification accuracy with the GCP. However, a blog written by Jennifer Villa (2018) emphasizes on the same concept wherein the comparison is made to know the advantages and disadvantages of running CNN model on both cloud and on standalone machines as shown in the figure 4.5

		Performance	Price	Setup	Maintenance	Security
Cloud	Traditional	✓✓	\$\$\$\$	✓	✓✓	—
	ML-Specific	✓✓	\$\$\$\$	✓✓	✓✓	—
On-Prem	Pre-Built	✓✓	\$\$	✓	✓	✓✓
	Built-from-scratch	✓✓	\$	—	—	✓✓

Figure 4.5 Cloud vs On-Prem

Accordingly, it is observed that cloud is the best platform in terms of performance, price, setup, maintenance and in security compared with the personal machines.

Also, to know the performance and the corresponding upfront costs, different cloud platforms have been compared with each other as shown in the figure.

	Low End Performance, 10% Utilization	Low End Performance, 100% Utilization	Mid Tier Performance, 10% Utilization	Mid Tier Performance, 100% Utilization	High-End Performance, 10% Utilization	High-End Performance, 100% Utilization
AWS	\$1.84	\$0.93	N/A	N/A	\$3.07	\$1.24
GCP	\$1.32	\$0.84	\$1.88	\$1.23	\$2.90	\$1.95
Paperspace	\$0.67	\$0.42	\$0.92	\$0.55	\$2.32	\$1.38
Pre-Built DL Server	\$1.56	\$0.16	\$1.67	\$0.17	\$3.42	\$0.34
Built-from-scratch DL Server	\$1.17	\$0.12	\$1.26	\$0.13	\$2.57	\$0.26

Figure 4.6 Performance in different Cloud Platform

4.3. Conclusion

Based on the results obtained after executing CNN model to classify hyperspectral multiple fruit images, the classification accuracy obtained was 98.25%. This shows that CNN model could significantly detect and classify fruit images into 50 classes even though the images were shot in different angles. Also, to reduce the computation load and to reduce the long running time the model was executed on Google Cloud Platform and it was observed the time duration to train and validate the model was completed in seconds unlike local machines. Using Google Cloud Platform has considerable improvement in terms of run times, and it is observed to be stable without crashes. Overall, the results were positive in terms of obtaining higher classification accuracy in less interval of time with less computation overhead.

5. CONCLUSION

computation overhead This chapter talks about the conclusions and lists recommendations for future work. Deep learning is being heavily used in the industry in many areas. Industry can benefit from further research and newer techniques with increased accuracy. Based on the extensive literature review and the tests conducted, it is learnt that decreasing cost of hardware and availability of bigdata enables research organisations and companies to extensively use deep learning to solve their various problems and derive business benefits from the same. Earlier the capability rested solely with big organizations and research and academic institutes only. Now it is available for even smaller business organisations. After conducting the experiment to classify multiple fruit images, it was found that Convolutional neural network could detect and classify fruit images even when they were shot in different angles with 98 % accuracy and hence, the research question ‘Do Convolutional Neural Network predict different fruit images even when the images are shot in different angles are fed as an input?’ has been answered.

The limitations that existed earlier also included the number of output classes one could use at a time, and the number of epochs we can run without having to wait for a long time for the algorithms to run. After executing the CNN algorithm on Google Cloud Platform, it is observed that the time taken for both training and validation is in seconds highlighting the fact that using distributed and scalable systems and using cloud environments (like Google Cloud Platform) , we can scale horizontally and increase the number of output classes, increase the quality of classification, and increase the depth of the layers in CNN and reduce the run times for modelling and prediction. This means that the accuracy of prediction can be increased at lower cost and manageable time frames and hence, the sub question of the research ‘Do Google Cloud Platform facilitate Convolutional neural network to classify images in lesser interval of time with higher accuracy?’ has been answered positively

5.1. RECOMMENDATION

answered positively with the availability of multiple cloud offerings, one of the recommendations is to create a comparative chart of accuracy and run times for the same problem using the three major providers – GCP, Microsoft Azure and AWS. Using similar hardware configurations will enable us to understand if their underlying architecture can have a difference in performance. Due to time and cost constraints, this study could not be taken at this point of time. As more users are adopting Deep Learning libraries like Keras, TensorFlow and PyTorch, it is expected that more modular and powerful libraries would come up in the future and a study could be carried out if the libraries offer an edge over the existing ones.

Appendix A: References

- ‘aitrends’ (2017) available: <http://aitrends.com>. Google launches GPU support for its CloudPlatform. <http://aitrends.com/big-data/google-launches-gpu-supportcloud-platform/>. Online; accessed 2 February 2017. 2017.
- ‘Architecture of cloud computing services’ (2017) available: <http://cloudcomputingnet.com>. Cloud Computing Architecture. <http://cloudcomputingnet.com/cloud-computing-architecture/>. Online; accessed 27 February 2017. 2015.
- Dhok, S., Bhurane, A., Kothari, A. (2019) ‘Automated Hyperspectral Image Classification Using Spatial-Spectral Features’, in *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, Presented at the 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN), IEEE: Noida, India, 184–189, available: <https://ieeexplore.ieee.org/document/8711579/> [accessed 20 Nov 2019].
- Dustin W. Stout (2019) available: <https://dustinstout.com/social-media-statistics/>.
- Faaeq, A., Guruler, H., Peker, M. (2018) ‘Image classification using manifold learning based non-linear dimensionality reduction’, in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, Presented at the 2018 26th Signal Processing and Communications Applications Conference (SIU), IEEE: Izmir, 1–4, available: <https://ieeexplore.ieee.org/document/8404441/> [accessed 28 Jul 2019].
- Fang, L., He, N., Li, S., Plaza, A.J., Plaza, J. (2018) ‘A New Spatial–Spectral Feature Extraction Method for Hyperspectral Images Using Local Covariance Matrix Representation’, *IEEE Transactions on Geoscience and Remote Sensing*, 56(6), 3534–3546.
- Gokkulnath TS (2017) ‘Choosing components for Personal Deep learning Machine’, available: <https://medium.com/mlreview/choosing-components-for-personal-deep-learning-machine-56bae813e34a>.
- Hamouda, M., Ettabaa, K.S., Salim Bouhlel, M. (2018) ‘Modified Convolutional Neural Network based on Adaptive Patch Extraction for Hyperspectral Image Classification’, in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Presented at the 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE: Rio de Janeiro, 1–7, available: <https://ieeexplore.ieee.org/document/8491647/> [accessed 20 Nov 2019].
- Hansch, R., Hellwich, O. (2015) ‘Feature-independent classification of hyperspectral images by projection-based random forests’, in *2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Presented at the 2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), IEEE: Tokyo, Japan, 1–4, available: <http://ieeexplore.ieee.org/document/8075462/> [accessed 28 Jul 2019].
- He, N., Paoletti, M.E., Haut, J.M., Fang, L., Li, S., Plaza, A., Plaza, J. (2019) ‘Feature Extraction With Multiscale Covariance Maps for Hyperspectral Image Classification’, *IEEE Transactions on Geoscience and Remote Sensing*, 57(2), 755–769.

- Hussain, I., He, Q., Chen, Z. (2018) 'AUTOMATIC FRUIT RECOGNITION BASED ON DCNN FOR COMMERCIAL SOURCE TRACE SYSTEM', 15.
- Jennifer Villa (2018) 'Choosing deep learning infrastructure: Cloud vs on-prem', available: <https://determined.ai/blog/cloud-v-onprem/>.
- Khaing, Z.M., Naung, Y., Htut, P.H. (2018) 'Development of control system for fruit classification based on convolutional neural network', in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Presented at the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), IEEE: Moscow, 1805–1807, available: <http://ieeexplore.ieee.org/document/8317456/> [accessed 12 Jun 2019].
- Lee, H., Kwon, H. (2017) 'Going Deeper With Contextual CNN for Hyperspectral Image Classification', *IEEE Transactions on Image Processing*, 26(10), 4843–4855.
- Lee, H., Lee, B.-T. (2016) 'Selective inference for accelerating deep learning-based image classification', in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, Presented at the 2016 International Conference on Information and Communication Technology Convergence (ICTC), IEEE: Jeju, 135–137, available: <http://ieeexplore.ieee.org/document/7763453/> [accessed 28 Jul 2019].
- Loussaief, S., Abdelkrim, A. (2016) 'Machine learning framework for image classification', in *2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, Presented at the 2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), IEEE: Hammamet, Tunisia, 58–61, available: <http://ieeexplore.ieee.org/document/7939841/> [accessed 28 Jul 2019].
- Mei, S., Jiang, R., Ji, J., Sun, J., Peng, Y., Zhang, Y. (2017) 'Invariant feature extraction for image classification via multi-channel convolutional neural network', in *2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Presented at the 2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), IEEE: Xiamen, China, 491–495, available: <http://ieeexplore.ieee.org/document/8266528/> [accessed 28 Jul 2019].
- Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., Hodjat, B. (2019) 'Evolving Deep Neural Networks', in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, Elsevier, 293–312, available: <https://linkinghub.elsevier.com/retrieve/pii/B9780128154809000153> [accessed 6 Nov 2019].
- Nanni, L., Brahnam, S., Maguolo, G. (2019) 'Data Augmentation for Building an Ensemble of Convolutional Neural Networks', in Chen, Y.-W., Zimmermann, A., Howlett, R.J. and Jain, L.C., eds., *Innovation in Medicine and Healthcare Systems, and Multimedia*, Springer Singapore: Singapore, 61–69, available: http://link.springer.com/10.1007/978-981-13-8566-7_6 [accessed 20 Nov 2019].
- Naveen Joshi (2018) 'understanding computer vision and image processing', available: <https://www.allerin.com/blog/understanding-the-difference-between-computer-vision-and-image-processing>.

- Shijie, J., Ping, W., Peiyi, J., Siping, H. (2017) 'Research on data augmentation for image classification based on convolution neural networks', in *2017 Chinese Automation Congress (CAC)*, Presented at the 2017 Chinese Automation Congress (CAC), IEEE: Jinan, 4165–4170, available: <http://ieeexplore.ieee.org/document/8243510/> [accessed 28 Jul 2019].
- Sidike, P., Chen, C., Asari, V., Xu, Y., Li, W. (2016) 'Classification of hyperspectral image using multiscale spatial texture features', in *2016 8th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Presented at the 2016 8th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), IEEE: Los Angeles, CA, USA, 1–4, available: <http://ieeexplore.ieee.org/document/8071767/> [accessed 28 Jul 2019].
- Su, H., Du, P. (2015) 'A novel dynamic classifier ensemble algorithm for hyperspectral image classification', in *2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Presented at the 2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), IEEE: Tokyo, Japan, 1–4, available: <http://ieeexplore.ieee.org/document/8075420/> [accessed 28 Jul 2019].
- Suganuma, M., Shirakawa, S., Nagao, T. (2017) 'A genetic programming approach to designing convolutional neural network architectures', in *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '17*, Presented at the the Genetic and Evolutionary Computation Conference, ACM Press: Berlin, Germany, 497–504, available: <http://dl.acm.org/citation.cfm?doid=3071178.3071229> [accessed 20 Nov 2019].
- sumit saha (2018) 'A comprehensive guide to Convolutional neural network', available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- Sun, S., Zhong, P., Xiao, H., Liu, F., Wang, R. (2015) 'An active learning method based on Markov random fields for hyperspectral images classification', in *2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Presented at the 2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), IEEE: Tokyo, Japan, 1–4, available: <http://ieeexplore.ieee.org/document/8075483/> [accessed 28 Jul 2019].
- Tang, J., Ge, Y., Liu, Y. (2016) 'Application of visual saliency and feature extraction algorithm applied in large-scale image classification', in *2016 International Conference on Communication and Electronics Systems (ICCES)*, Presented at the 2016 International Conference on Communication and Electronics Systems (ICCES), IEEE: Coimbatore, India, 1–6, available: <http://ieeexplore.ieee.org/document/7889903/> [accessed 28 Jul 2019].
- Xia, J., Chanussot, J., Du, P., He, X. (2016) 'Rotation-Based Support Vector Machine Ensemble in Classification of Hyperspectral Data With Limited Training Samples', *IEEE Transactions on Geoscience and Remote Sensing*, 54(3), 1519–1531.
- Xiaoling Xia, Cui Xu, Bing Nan (2017) 'Inception-v3 for flower classification', in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, Presented at the 2017 2nd International Conference on Image, Vision and Computing (ICIVC), IEEE:

Chengdu, China, 783–787, available: <http://ieeexplore.ieee.org/document/7984661/> [accessed 28 Jul 2019].

Zhao, G., Liu, G., Fang, L., Tu, B., Ghamisi, P. (2019) ‘Multiple convolutional layers fusion framework for hyperspectral image classification’, *Neurocomputing*, 339, 149–160.

Zhong, S., Chang, C.-I., Zhang, Y. (2018) ‘Iterative Support Vector Machine for Hyperspectral Image Classification’, in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Presented at the 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE: Athens, 3309–3312, available: <https://ieeexplore.ieee.org/document/8451145/> [accessed 28 Jul 2019].

Appendix B: Code Listing

The code for the CNN is shown below.

```
import numpy as np

import csv

import cv2

import os, glob

import random

import sklearn

from sklearn.model_selection import train_test_split

import time

from keras.datasets import cifar10

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Flatten

from keras.constraints import maxnorm

from keras.optimizers import SGD

from keras.layers import Conv2D

from keras.layers.convolutional import MaxPooling2D

from keras.utils import np_utils

from matplotlib import pyplot as plt

from subprocess import check_output
```

```
#Create the function for load data from folder

def loadDataFromFolder(data_dir):

    # Get all the subfolders. A subfolder is there for each output class

    labels = []

    images = []

    category = 0

    # Loop through the directories and get the data in
    # two lists, labels and images.

    for d in directories:

        label_dir = os.path.join(data_dir, d)

        file_names = [os.path.join(label_dir, f)

                       for f in os.listdir(label_dir)

                       if f.endswith(".jpg")]
```

```
# Tweak speed using stop if needed

    #stop = 0

    for f in file_names:

        img = cv2.imread(f)

        imresize = cv2.resize(img, (100, 100))

        #plt.imshow(imresize)

        images.append(imresize)

        labels.append(category)

        # remove this to use full data set

        #if stop > 30:

        #break

        #stop += 1

        #print(stop)

        # end early stop

    category += 1

return images, labels

data_dir = "/home/sushmitha/fruits_360/fruit-bin/train"

training_images, training_labels = loadDataFromFolder(data_dir)
```

```
# confirm that we have the correct data

print(len(training_images))

print(len(training_labels))

print("Training set dims")

#cv2.imshow(str(training_labels[random.randint(0,(len(training_labels)-1))]),
training_images[random.randint(0,(len(training_images)-1))])

#cv2.waitKey(0)

#cv2.destroyAllWindows()


data_dir = "/home/sushmitha/fruits_360/fruit-bin/valid"

validation_images, validation_labels = loadDataFromFolder(data_dir)
```

```
# confirm that we have the datain

print("Validation set dims")

print(len(validation_images))

print(len(validation_labels))

#cv2.imshow(str(validation_labels[random.randint(0,(len(validation_labels)-1))]),
validation_images[random.randint(0,(len(validation_images)-1))])

#cv2.waitKey(0)

#cv2.destroyAllWindows()


data_dir = "/home/sushmitha/fruits_360/fruit-bin/test1"

testing_images, testing_labels = loadDataFromFolder(data_dir)
```

```
# confirm that we have the data

print("Test set dims")

print(len(testing_images))

print(len(testing_labels))

#cv2.imshow(str(testing_labels[random.randint(0,(len(testing_labels)-1))]),
testing_images[random.randint(0,(len(testing_images)-1))])

#cv2.waitKey(0)

#cv2.destroyAllWindows()


# normalize/standardize inputs from 0-255 and 0.0-1.0
X_train = np.array(training_images).astype('float32')
X_cv = np.array(validation_images).astype('float32')
X_test = np.array(testing_images).astype('float32')
X_train = X_train / 255.0
X_cv = X_cv / 255.0
X_test = X_test / 255.0
```

```
# Represent in one hot encode outputs
y_train = np.array(training_labels)
y_cv = np.array(validation_labels)

# y_test = np.array(testing_labels) %%% Irrelevant Data
y_train = np_utils.to_categorical(y_train)
y_cv = np_utils.to_categorical(y_cv)

# y_test = np_utils.to_categorical(y_test) %%% Irrelevant Data
num_classes = y_cv.shape[1]
```

```
# print details for debugging

print("X_train")

print(len(X_train))

print("X_cv")

print(len(X_cv))

print("y_train")

print(len(y_train))

print("y_cv")

print(len(y_cv))

print("Number of classes")

print(num_classes)

print("Data normalized and hot encoded.")
```

```
def create_cnn_model(num_classes, lr_rate):

    # Create cnn model

    model = Sequential()

    model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(100, 100, 3), padding='same',
activation='relu', kernel_constraint=maxnorm(3)))

    model.add (Dropout(0.2))

    model.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu',
kernel_constraint=maxnorm(3)))

    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())

    model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))

    model.add(Dropout(0.5))

    model.add(Dense(num_classes, activation='softmax'))
```

```
# Compile the model

epochs = 1

decay = lr/epochs

sgd = SGD(lr=lr, momentum=0.9, decay=decay, nesterov=False)

model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

print(model.summary())

return model, epochs
```

```
# Create CNN model

model, epochs = create_cnn_model(num_classes, 0.01)

print(" CNN model created. ")


# fit and run the cnn model

seed = 7

np.random.seed(seed)

model.fit(X_train, y_train, validation_data=(X_cv, y_cv), epochs=epochs, batch_size=64)
```



```
# Final evaluation of the model

scores = model.evaluate(X_cv, y_cv, verbose=1)

print("Accuracy: %.2f%%" % (scores[1]*100))

print("Training Done")


# Test the ConvNet


print("Evaluating test images ")

cnn_prediction = model.predict_on_batch(X_test)

print(cnn_prediction)

print("Predictions Done")
```


6.