

Deep Learning for Image Processing in WEKA Environment

Zanariah Zainudin¹, Siti Mariyam Shamsuddin² and Shafaatunnur Hasan³

^{1,2,3} School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia,
81310 Skudai, Johor

e-mail: zanariah86@gmail.com, sitimariyams@gmail.com, and
shafaatunnur@gmail.com

Abstract

Deep learning is a new term that is recently popular among researchers when dealing with big data such as images, texts, voices and other types of data. Deep learning has become a popular algorithm for image processing since the last few years due to its better performance in visualizing and classifying images. Nowadays, most of the image datasets are becoming larger in terms of size and variety of the images that can lead to misclassification due to human eyes. This problem can be handled by using deep learning compared to other machine learning algorithms. There are many open sources of deep learning tools available and Waikato Environment for Knowledge Analysis (WEKA) is one of the sources which has deep learning package to conduct image classification, which is known as WEKA DeepLearning4j. In this paper, we demonstrate the systematic methodology of using WEKA DeepLearning4j for image classification on larger datasets. We hope this paper could provide better guidance in exploring WEKA deep learning for image classification.

Keywords: *Image Classification, WEKA DeepLearning4j, WEKA Image, Deep Learning, Convolutional Neural Network.*

1 Introduction

Nowadays, image processing is becoming a popular approach in many areas such as engineering, medical and computer science area. Image processing is the most important part in many domains especially in medical imaging for object detection. Image processing is an approach where an image will be extracted to get useful information for image classification and detection or the image can be enhanced to get more information inside it. The basic step in image processing is where firstly, we need to import the image using any tool; after that we analyse the image using machine learning and compare the result with another benchmark dataset.

Machine learning is a popular approach for image classification and image detection. For an example, in this paper, one dataset containing butterfly and owl images is given. Using machine learning algorithms, a model has been developed to classify whether all the images are butterfly or owl. This step is known as image classification where the model learns from the training images, and classifies the testing images into their classes. Due to the increase in the image size and quality, an enhanced machine learning called deep learning have been introduced to tackle the current issues in image processing. Deep learning has been chosen for most image classification problems as it is more accurate in terms of classification due to the massive learning from the network itself. Deep learning is also known as the deep machine learning. The deep learning subset of the machine learning is illustrated in Fig 1 [1], [2]:

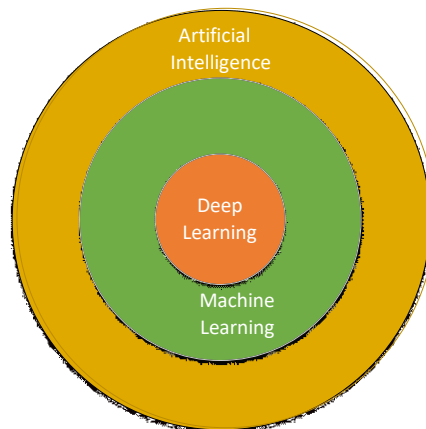


Fig 1: Deep Learning in Machine Learning

In other word, based on Fig 1, deep learning is summarized as follows [3]–[5]: Deep learning has many layers of the processing units for the input of the image. Each layer has massive sub-layers of hidden layers. This algorithm is not only for supervised but also applicable for unsupervised.

- If the deep learning is used for the unsupervised usually for pattern analysis, the deep learning will consist multiple levels of the features of input data.
- Subset of machine learning algorithms
- Will have a multiple representations that correspond to different levels/hierarchy of the levels.

There are many types of deep learning which is listed in Fig 2 as mentioned by Fei-Fei Li [6]:

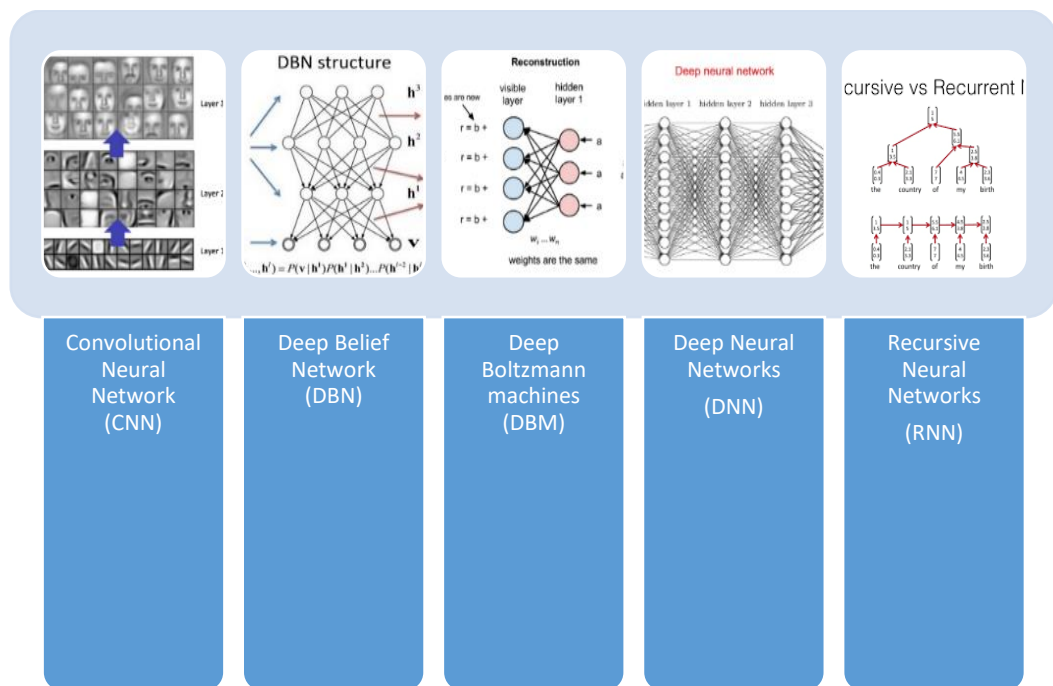


Fig.2 Type of Deep Learning Algorithm

As illustrated in Fig 3, there are many deep learning open source tools that can be used in image processing in image classification and object detection. The most popular open source tools are Tensorflow, followed by Keras. As we can see, other open source tools have also been developed by other research and widely used such as Caffe, DeepLearning4j, MXNet and GPULib. In this paper, we will focus more on Weka Deeplearning4j especially deep learning algorithm Convolutional Neural Network (CNN). We also use the Convolutional Neural Network to conduct the image processing for classification problem in this study. In addition, we use the popular MNIST dataset as a benchmark for comparison with another dataset. Detail explanations on Convolutional Neural Network will be described in the next sub-section.

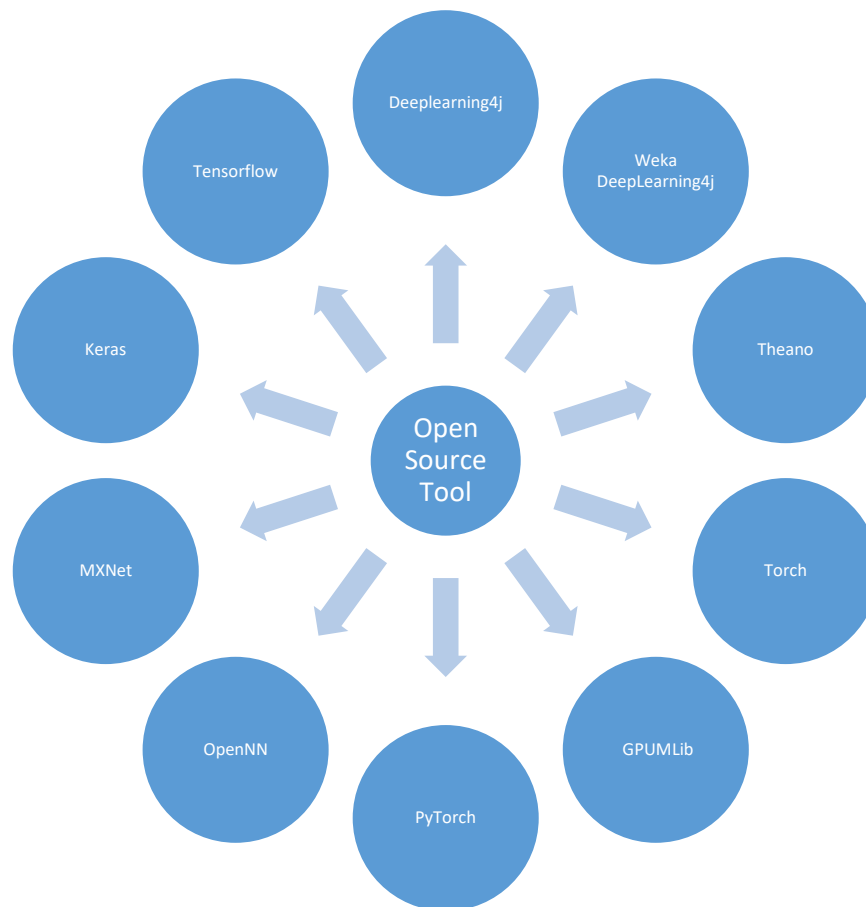


Fig 3: Deep Learning Open Source Tool

2 How does Convolutional Neural Network Work?

In deep learning algorithm, Convolutional Neural Network (CNN) is one of the algorithms used for the image classification. Convolutional Neural Network uses a feed forward Artificial Neural Network where it has successfully been applied in image detection and classification. Convolutional Neural Network (CNN) has been designed to get advantages on feed forward Artificial Neural Network.

Convolutional Neural Network (CNN) was developed based on the neuron inside the animal visual cortex (Cat) by Yann LeCun et al. in 1998 [8]. It was based on the animal neuron where neurons responded to the catalyst, the restrained area of the visual field known as the receptive field. The receptive fields of different neurons will somewhat overlap layer by layer as they cover all parts in the visual field, Yann LeCun et al. [7], and Yann LeCun et al. [8]. Convolutional neural network uses the same concepts where all layers process and classify the images. The deep learning algorithm called Convolutional Neural Network (CNN) is used

in this experiment. Convolutional Neural Network (CNN) architecture are organized in multiple layer and consists of convolution layer, pooling layer and normalization layer as illustrated in Fig 4 [7].

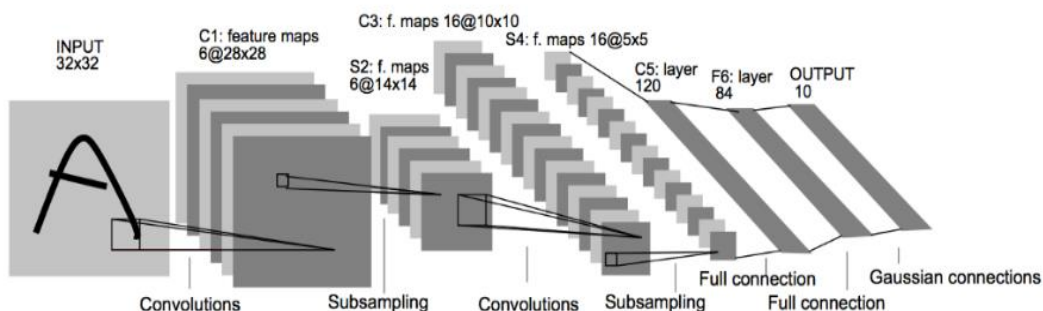


Fig 4: First Convolutional Neural Network (CNN) architecture by Yann LeCun et al.[7]

Mathematical formulation for Convolutional Neural Network (CNN) is based on Equation 1 [9], [10]:

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l\right) \quad (1)$$

where M_j represents a selection of input maps,

- Kernel is a smaller-sized matrix in comparison to the input dimensions of the image, which consists of real valued entries. Inside a convolution layer, all feature maps of the previous layers are convolved with kernels with activation function to form the output feature map. Each output map may merge with multiple input maps.
- Each output map has been given a parameter such as bias b ; however, for some of the output map, the input maps will be convolved with specific kernels. For an example, let say output map j and map k are both sum over input map i , then the kernels applied to map i are different for output maps j and k .

The summarization of Convolutional Neural Network (CNN) layer architecture for this paper has been summarized below [7], [11]–[20]:

- *Input Layer*: Inside input layer where the size and the pixel of the input image has been defined. The height, weight, and the number of colour channels of that image has been sorted out; either RGB (3) or grayscale (1). In this layer, the network has been specified to execute data augmentation, where the data has been transformed, flipped or cropped in the training set. In this layer, we are using $9 \times 9 \times 1$ as the input image X as shown in Fig 5.

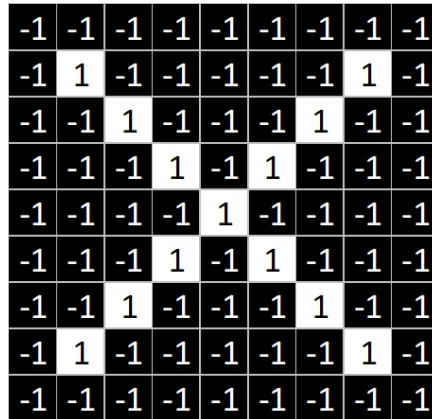


Fig 5: Input Image size 9x9

- Convolutional Layer:** This layer will convolve all the input images from the input layer to several feature maps in the output image. For this model, we used three layers where the first two layers will learn all the feature maps and initialize the Gaussian distribution with standard deviation of 0.001. Then, the third layer will learn all the filters from the Gaussian distributions with standard deviation of 0.001. The outcomes from the neurons of a convolutional layer more often will pass in a form of nonlinearity model. All outputs that are connected to local regions in the input will be computed with each computing. The neurons between their weights and a small region are connected to another layer. This may produce the results in three feature maps [7x7x3] if we decide to use 3 strides as shown in Fig 6.

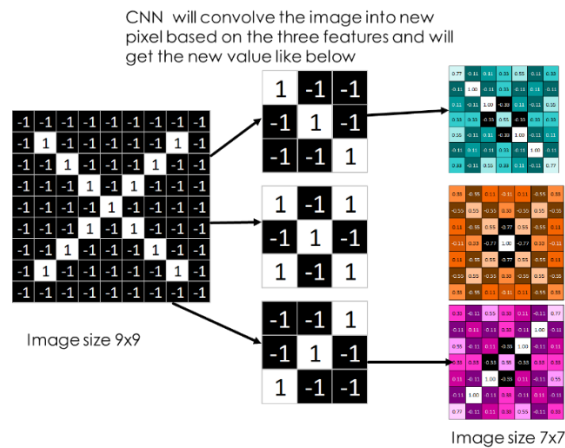


Fig 6: Convolutional Layer

- Subsampling Layer:** Inputs from the convolution layer can be improved to reduce the sensitivity of the filters to noise and variations. This process is called subsampling, and can be achieved by taking averages or taking the maximum over a sample of the dataset. Examples of subsampling methods (for image

dataset) include reducing the size of the image, or reducing the colour contrast across red, green, blue (RGB) channels as illustrated in Fig 7.

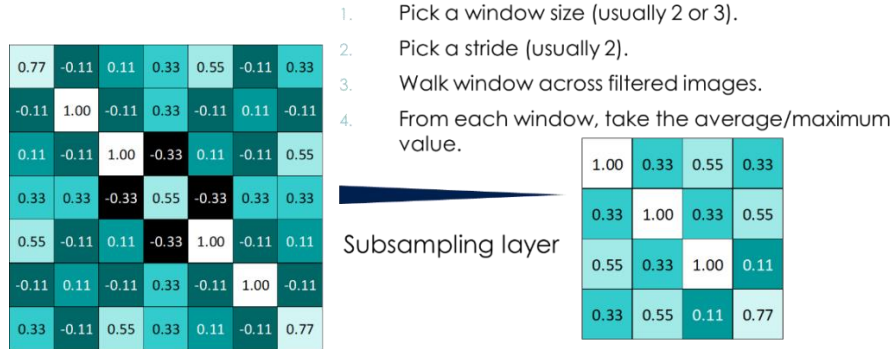


Fig 7: Subsampling Layer

- *ReLU layers:* ReLU is an activation function for non-linearity model. The ReLU layer will perform a threshold operation on each element, where the input value is less than zero and will be set to zero based on the Equation 2 below:

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

This layer will not change the dimension of output. Fig 8 will illustrate ReLU Layer.

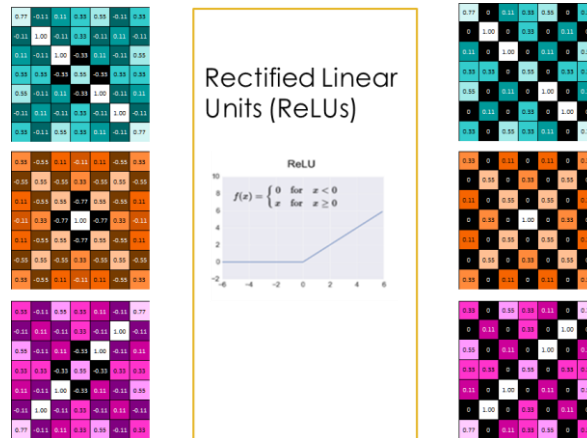


Fig 8: ReLU layer

- *Dense Layer:* simply a layer where each data is connected to each data in the next layer shown in Fig 9.

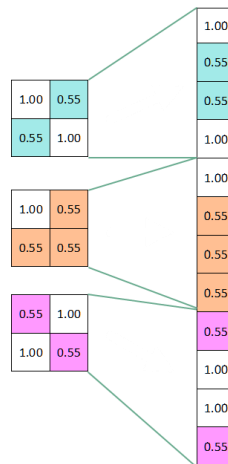


Fig 9: Dense layer

- *Output Layer*: The last layer for CNN is called Output Layer. It depends on the output of the classification, for example in this research we used three image classification datasets, where the output classifies all the classes by using *OutputLayer* function shown in Fig 10.

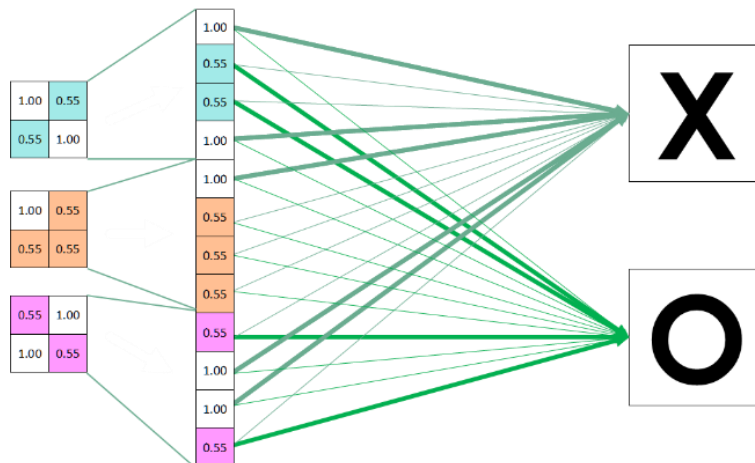


Fig 10: Output layer

3 Deep Learning4J using WEKA

WEKA is created by a group of researchers from University of Waikato. It is the best-known free tool for Machine Learning algorithm as shown in Fig 5. WEKA is a software from open source where it can do data mining and machine learning algorithms, including: pre-processing of data, classification, clustering, association rule extraction and others jobs [21]–[24]. It is incredibly powerful for researchers who are not familiar with programming environments. Many analysts use WEKA

due to its simplicity and open-source tool. Inside WEKA, there are five applications which are WEKA Explorer, WEKA Experimenter, WEKA KnowledgeFlow, WEKA Workbench and Simple CLI. The WEKA interface is shown in Fig 11 [22], [25]:

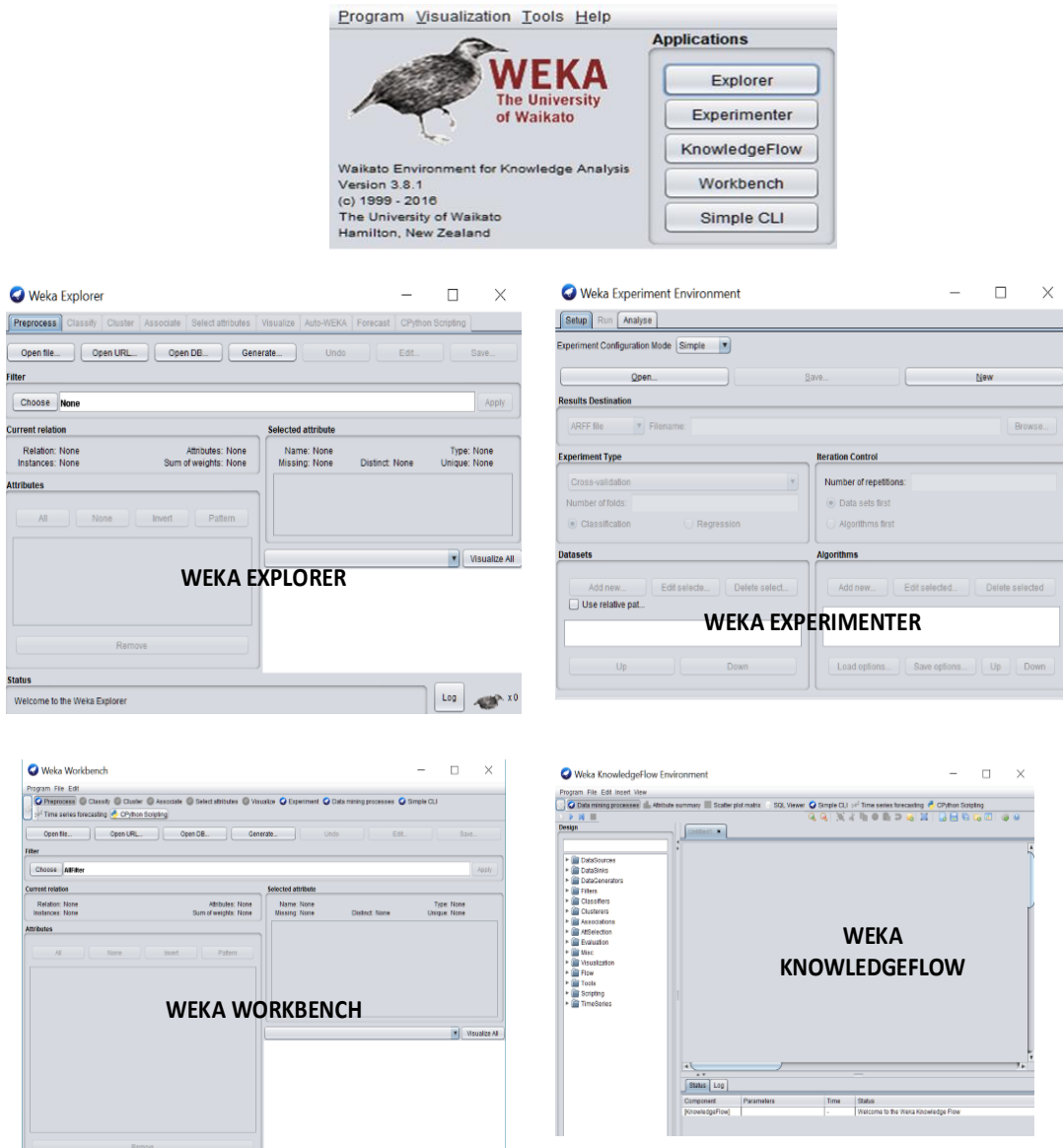


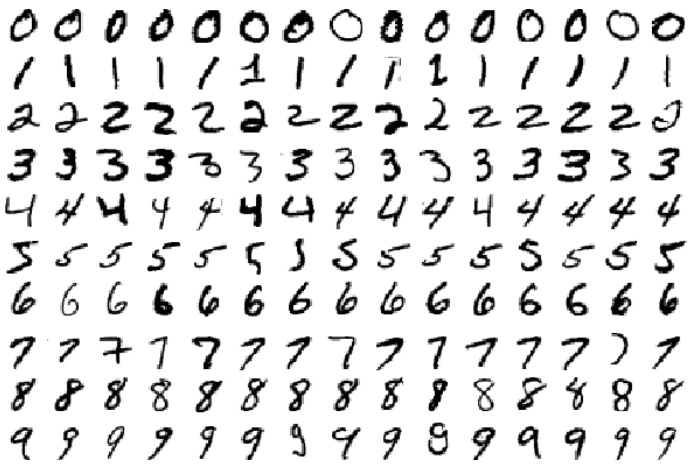
Fig 11: WEKA User Interface and Environments

WEKA's goal is to create a machine learning process which is extremely easy to use and integrate with another type of programming. In WEKA, there are many types of machine learning algorithms that analyse the datasets. Researchers

can find the best machine learning algorithm to get the best solution to solve the classification or regression problem to achieve their goals. In this study, three datasets are obtained from public database [26]–[28]. Tables 1 to 3 give the explanation of each dataset. The datasets contain different images and classes. The famous dataset used for deep learning benchmark is MNIST Dataset. The dataset are been used in this experiments are listed below:

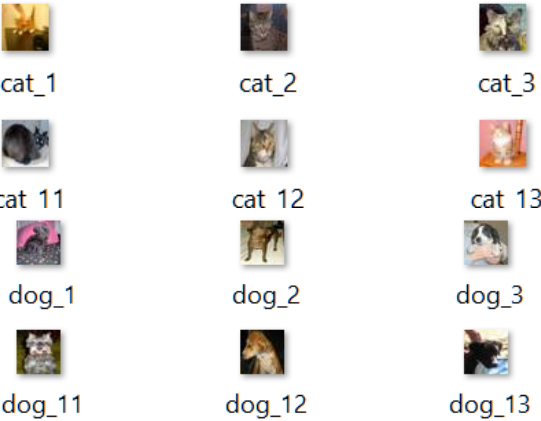
- MNIST Dataset (Table 1)
- Dog vs Cat Dataset (Table 2)
- Butterfly vs Owl (Table 3)
- AMIDA-13 Dataset (Table 4)

Table 1: Information on MNIST dataset

Dataset Name :	MNIST
Description :	The MNIST database was constructed from MNIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits
Size Image	28x28 (original size)
Image:	60000 Training 10000 Testing
Distribution dataset:	The dataset are balanced on the between each class
Sample of dataset:	

The second dataset used for this experiment is dog_cat dataset from Kaggle. All the information about the dataset has been described in Table 2 below:

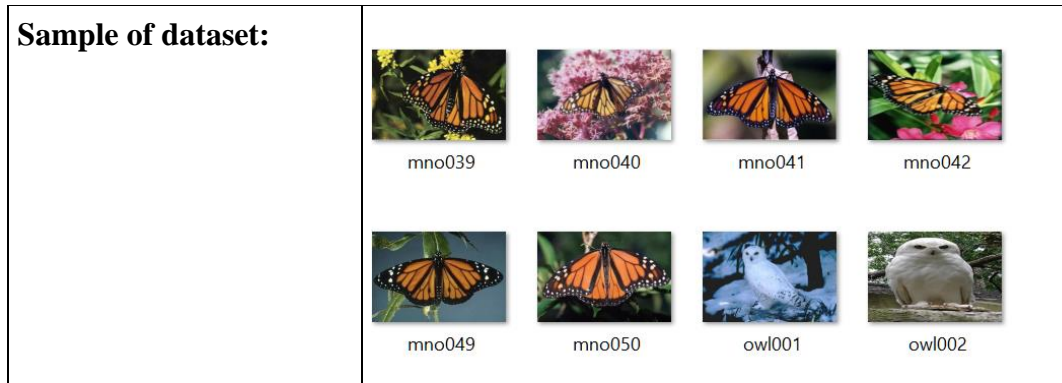
Table 2: Information on dog_cat dataset

Dataset Name :	dog_cat
Description :	Images of dog and images of cat
Image Size:	32x 32 (resized image)
No of Images	17500 images for training 7500 images for testing
Distribution dataset:	The dataset are balanced for the two classes
Sample of dataset:	

The third dataset used for this experiment is butterfly_owl dataset from Github WEKA Deeplearning4j. All the information about the dataset has been described in Table 3 below:

























Table 3: Information on butterfly_owl dataset

Dataset Name :	butterfly_owl
Description :	Images of butterfly and images of owl
Image Size:	350 x 235 (resized image)
No of Images	80 images for training 20 images for testing
Distribution dataset:	The dataset are balance on the between each class -50 butterfly -50 owl



The fourth dataset used for this experiment is AMIDA-13 dataset from medical challenges. All the information about the dataset is described in Table 4:

Table 4: Information on AMIDA-13 dataset

Dataset Name :	AMIDA-13								
Description :	Images of histopathology medical images								
Image Size:	80x80 (extracted patch from 2048x2048 slide images)								
No of Images	698 images for training 466 images for testing								
Distribution dataset:	The dataset are imbalance on the between each class -539 mitosis -625 nonmitosis								
Sample of dataset:	<table border="0"> <tr> <td> mitosis_316</td> <td> mitosis_317</td> <td> mitosis_318</td> <td> mitosis_319</td> </tr> <tr> <td> nonmitosis_2</td> <td> nonmitosis_3</td> <td> nonmitosis_4</td> <td> nonmitosis_5</td> </tr> </table>	 mitosis_316	 mitosis_317	 mitosis_318	 mitosis_319	 nonmitosis_2	 nonmitosis_3	 nonmitosis_4	 nonmitosis_5
 mitosis_316	 mitosis_317	 mitosis_318	 mitosis_319						
 nonmitosis_2	 nonmitosis_3	 nonmitosis_4	 nonmitosis_5						

4 WEKA Deeplearning4j for Image Classification

WEKA DeepLearning4j is originally developed by Mark Hall for the past few year in year 2017. This package contains deep learning package for the WEKA workbench. The package provides classifier for feedforward networks, including convolutional networks, and Deep learning trained using the Deeplearning4j library.

For this experiment, we just use CPU package support. We can fork the script at [Github](https://github.com/Waikato/wekaDeeplearning4j) using the link stated here: <https://github.com/Waikato/wekaDeeplearning4j>. In this experiment we run WEKA with console to see more detail information on accuracy for every epoch. If the dataset consists of a set of image files, it is necessary to prepare a meta-data ARFF file in the following format as shown in Fig 12:

```

1 @relation butterfly_vs_owl
2 @attribute filename string
3 @attribute class {BUTTERFLY,OWL}
4 @data
5 mno001.jpg,BUTTERFLY
6 mno002.jpg,BUTTERFLY
7 mno003.jpg,BUTTERFLY
8 mno004.jpg,BUTTERFLY

```

Fig 12: ARFF format for butterfly_owl dataset

The package provides so called Instance Iterators to load the given dataset in a correct shape as listed below:

- Default Instance Iterator
- Convolution Instance Iterator
- Image Instance Iterator
- Convolutional Instance Iterator

In this study, we used *Default Instance Iterator* to classify the entire image using the WEKA Deeplearning4j as shown in Fig 13:

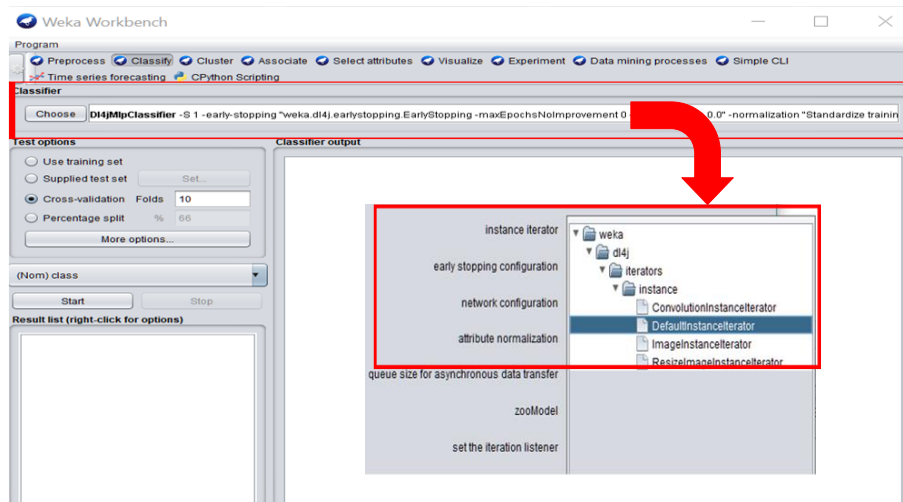


Fig 13: WEKA Deeplearning4j Classifier

We have to set the height, width and number of channels so that the internals can interpret the flattened vector of each image in the ARFF file. This file informs the internals about the association between the image files and their labels. The

parameter setting is for the ImageInstanceIterator which consists of 4 parameters shown in Fig 14 below:

- desired height: Height of the images
- desired width: Width of the images
- desired number of channels: Depth of the image (example: RGB images with a depth of 3, whereas Grayscale images have a depth of 1)
- directory of images: The absolute path to the location of the images listed in the meta-data ARFF file

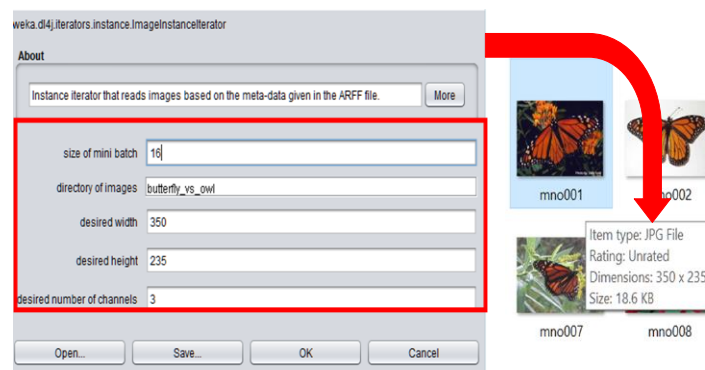


Fig 14: Setting up the parameter for Weka Deeplearning4j Classifier

The following Convolutional Neural Network Layers have been set based on the Convolutional layer, Subsampling Layer, Dense Layer and Output layer and we can compare with 3, 4, 5, 6 and 7 and the layers which are more suitable for each dataset. We can see each dataset performance measure which will be elaborated more in the next section.

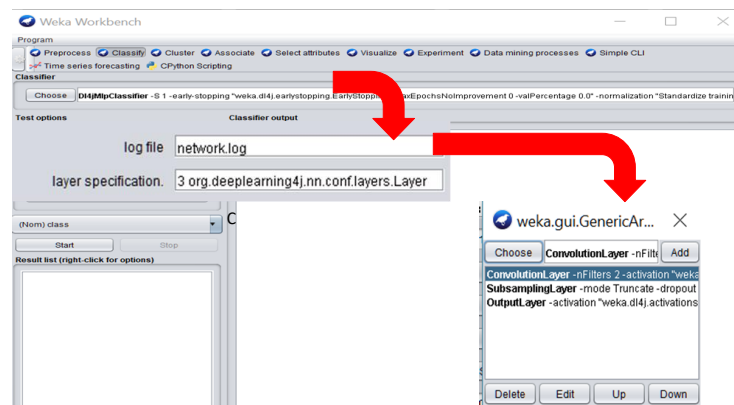
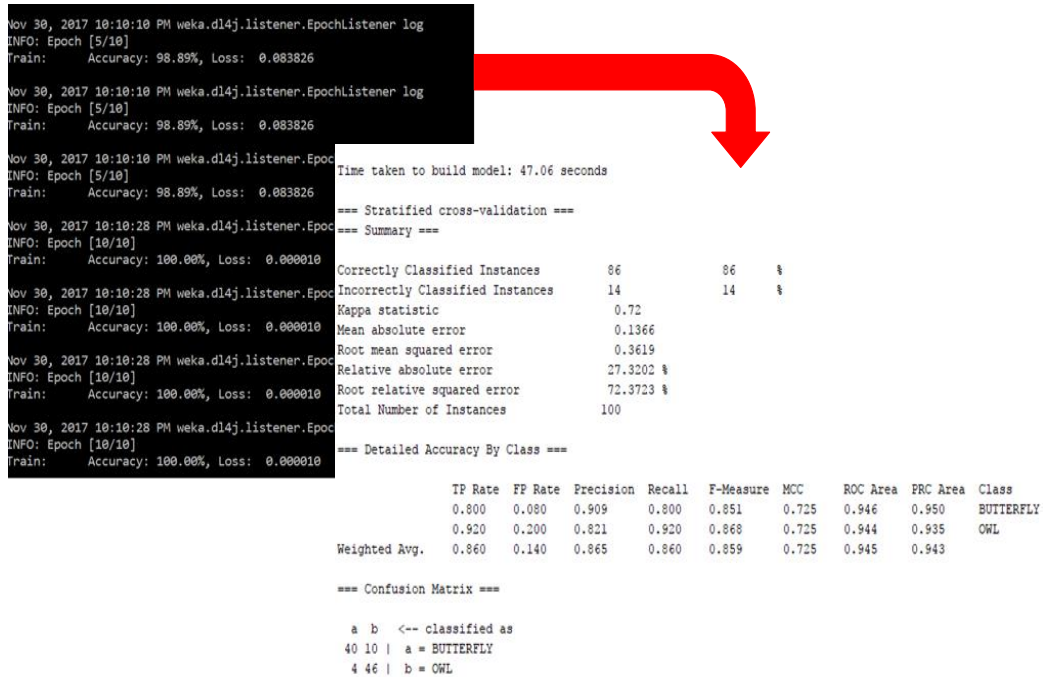


Fig 15: Layer Setup for WEKA Deeplearning4j Classifier

The results from the WEKA Deeplearning4j can be seen in the console of WEKA and the information for every epoch for the accuracy and loss are stated there. For WEKA, we can see the summary of the correctly classified instances, TP

Rate, FP Rate, Precision, Recall, F-Measure, MCC, ROC area and PRC Area. The results from this experiment as shown in Fig 16 will be briefly explained in the next section.



```

Nov 30, 2017 10:10:10 PM weka.d14j.listener.EpochListener log
INFO: Epoch [5/10]
Train: Accuracy: 98.89%, Loss: 0.083826

Nov 30, 2017 10:10:10 PM weka.d14j.listener.EpochListener log
INFO: Epoch [5/10]
Train: Accuracy: 98.89%, Loss: 0.083826

Nov 30, 2017 10:10:10 PM weka.d14j.listener.EpochListener log
INFO: Epoch [5/10]
Train: Accuracy: 98.89%, Loss: 0.083826
Time taken to build model: 47.06 seconds

Nov 30, 2017 10:10:28 PM weka.d14j.listener.EpochListener log
INFO: Epoch [18/10]
Train: Accuracy: 100.00%, Loss: 0.000010

Nov 30, 2017 10:10:28 PM weka.d14j.listener.EpochListener log
INFO: Epoch [18/10]
Train: Accuracy: 100.00%, Loss: 0.000010

Nov 30, 2017 10:10:28 PM weka.d14j.listener.EpochListener log
INFO: Epoch [18/10]
Train: Accuracy: 100.00%, Loss: 0.000010

Nov 30, 2017 10:10:28 PM weka.d14j.listener.EpochListener log
INFO: Epoch [18/10]
Train: Accuracy: 100.00%, Loss: 0.000010

==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances      86      86  %
Incorrectly Classified Instances    14      14  %
Kappa statistic                    0.72
Mean absolute error                 0.1366
Root mean squared error             0.3619
Relative absolute error             27.3202 %
Root relative squared error         72.3723 %
Total Number of Instances          100

==== Detailed Accuracy By Class ====
              TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
0.800  0.080  0.909  0.800  0.851  0.725  0.946  0.950  BUTTERFLY
0.920  0.200  0.821  0.920  0.868  0.725  0.944  0.935  OWL
Weighted Avg.  0.860  0.140  0.865  0.860  0.859  0.725  0.945  0.943

==== Confusion Matrix ====

a b <-- classified as
40 10 | a = BUTTERFLY
4 46 | b = OWL

```

Fig 16: Result from experiment using WEKA Deeplearning4j classifier

5 Experimental Result using WEKA Deeplearning4j

This section discusses the experimental result using WEKA Deeplearning4j Convolutional Neural Network (CNN). Table 4 shows the parameter setup for this experiment.

Table 4: Details of CNN layer for the experiment

Type of layer	Layer Detail
CNN 3-Layer	Convolutional Layer Subsampling Layer Output Layer
CNN 4-Layer	Convolutional Layer Subsampling Layer Dense Layer Output Layer
CNN 5-Layer	Convolutional Layer Subsampling Layer Convolutional Layer Subsampling Layer

	Output Layer
CNN 6-Layer	Convolutional Layer Subsampling Layer Convolutional Layer Subsampling Layer Dense Layer Output Layer
CNN 7-Layer	Convolutional Layer Subsampling Layer Convolutional Layer Subsampling Layer Dense Layer Dense Layer Output Layer

The dataset is divided into training (70%) and testing (30%). This is important to ensure the learning generalization has been achieved by feeding the classifier to new datasets. Table 4 show the experimental result on the accuracy, precision, sensitivity F-Measure of using Convolutional Neural Network (CNN) based on the MNIST dataset, cat_dog dataset, and butterfly_owl dataset. Based on the table below, we can see that suitable layer can increase the performance measure for all three datasets. Based on Fig 17 MNIST dataset, we can see that CNN with 5 layers is given the best accuracy, precision, sensitivity and f-measure compared to CNN with other layer. In conclusion, the best MNIST dataset, we can see that using CNN-5 layer gave the best result for MNIST performance measure.

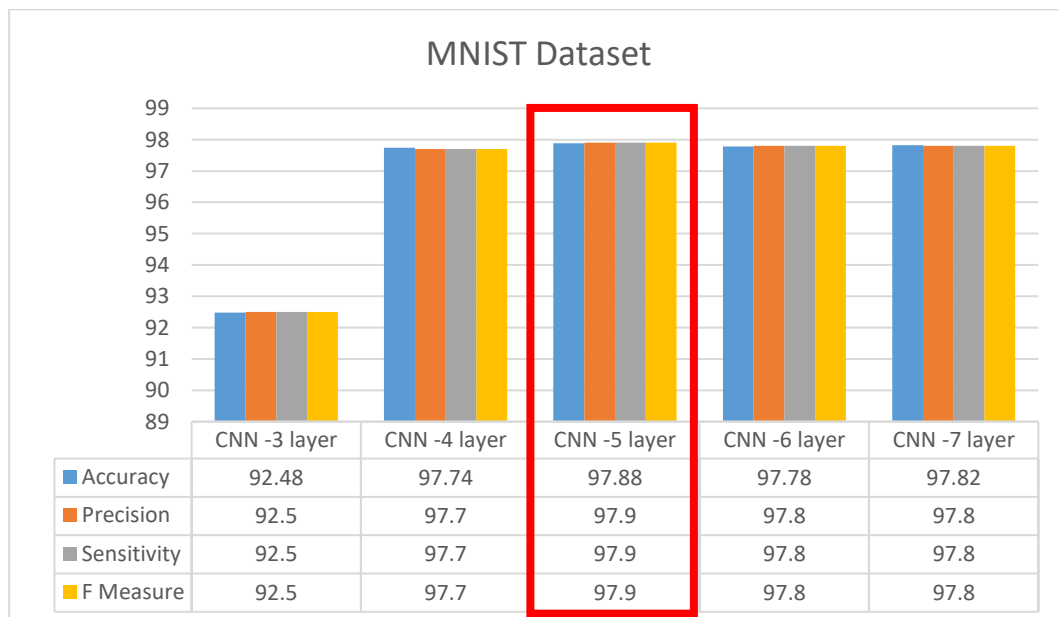


Fig 17: Result for MNIST dataset for each layer

Fig 18 shows the results of dog_cat dataset which indicate that CNN with 7 layers gave the best accuracy, precision, sensitivity and f-measure compared to MNIST dataset where CNN with 5 layers proved to be the best result. In conclusion, in reference to the best MNIST dataset, we can see that using CNN with 7 layers produced the best result for dog_cat dataset.

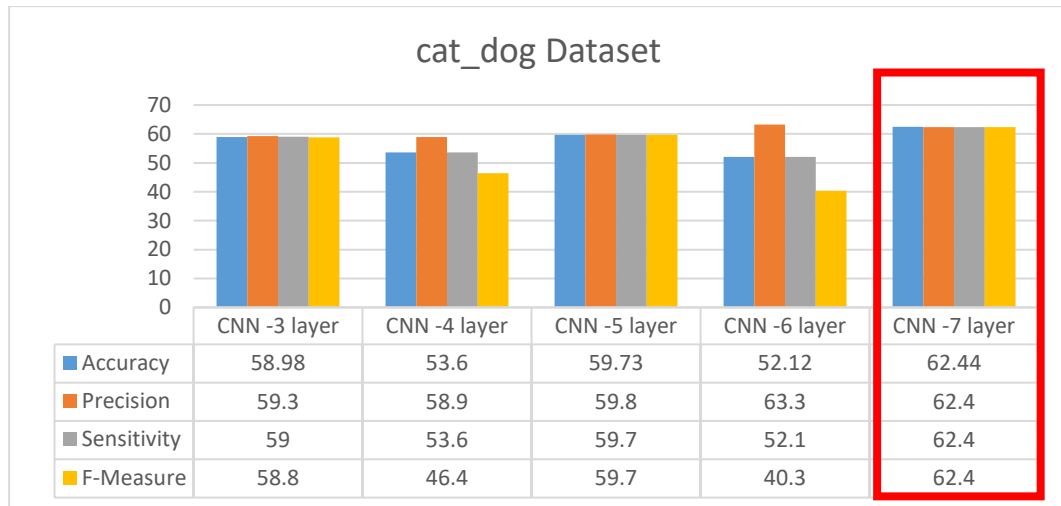


Fig 18: Result for cat_dog dataset for each layer

The third dataset shown in Fig 19 is butterfly_owl. We can see that CNN with 7 layers achieved the best accuracy, precision, sensitivity and f-measure. In conclusion, the best butterfly_owl dataset, we can see that using CNN with 7 layers gave the best result for butterfly_owl dataset.

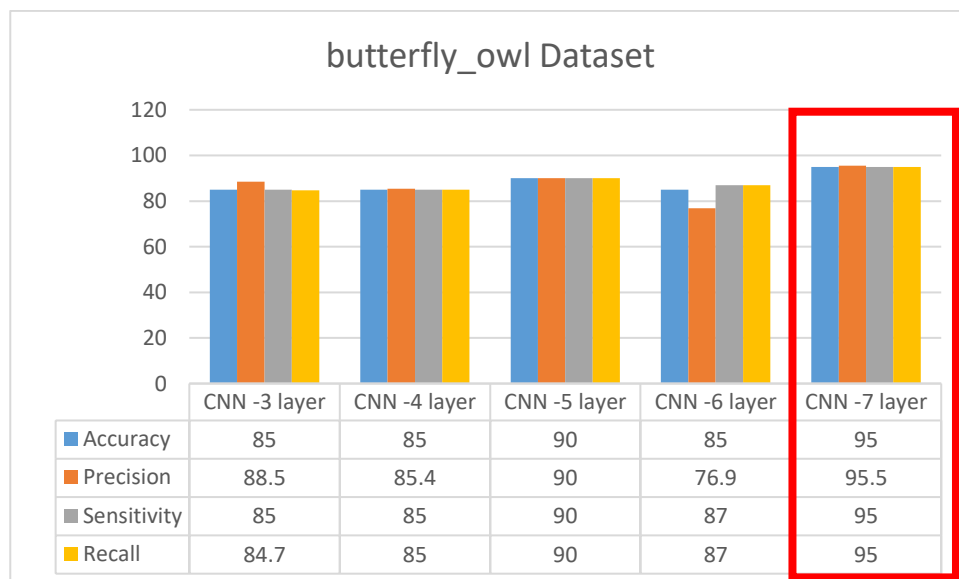


Fig 19: Result for butterfly_owl dataset for each layer

The final dataset shown in Fig 20 is AMIDA-13. We can see that CNN with 7 layers had the best accuracy, precision, sensitivity and f-measure. In conclusion, referring to the best AMIDA-13 dataset, we can see that using CNN with 7 layers produced the best result for AMIDA-13 dataset. In conclusion, it is proven that using the right CNN layer can give optimum and best result based on the dataset itself.

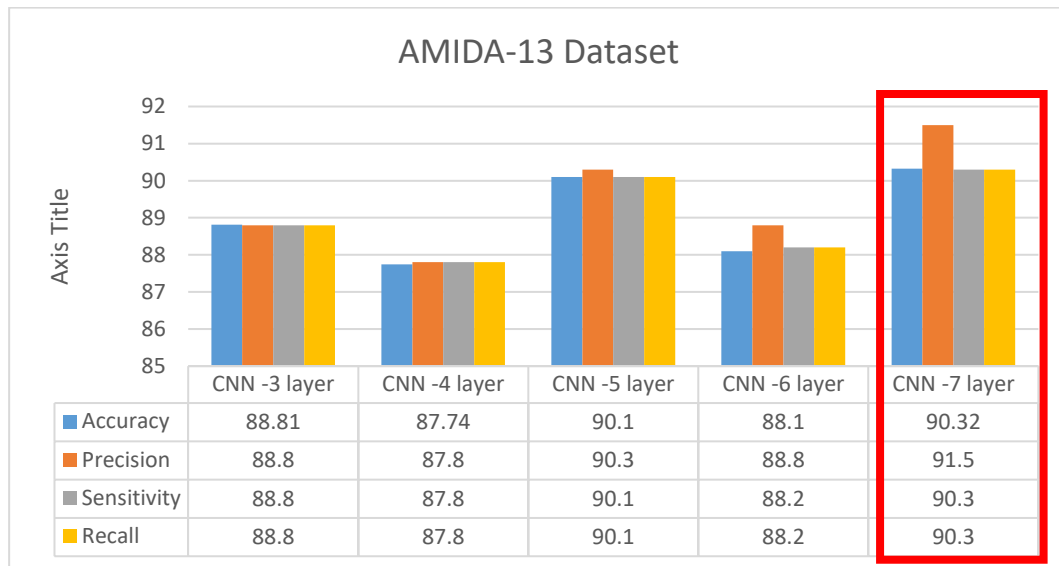


Fig 20: Result for AMIDA-13 dataset for each layer

6 Conclusion

In this study, we used deep learning algorithm of Convolutional Neural Network (CNN) to classify all images from three public datasets. We can see that by adding suitable layer for each dataset, we increased Accuracy, Precision, Sensitivity, and F-Measure. This is because CNN architecture allows more training inside the hidden layers which included the model using the correct Convolutional Layer, ReLU layer, Dense Layer, Subsampling layer and Fully Connected Layer which enabled the model to learn more features on the image.

As we can see from the experiments, CNN was also suitable with larger image segments because in CNN we used larger image patch which was 350x235x3 in size and other datasets where the image patch sizes were smaller: 32x32x3 and 28x28x1, respectively. We can see the difference of the result based on Tables 5 to 7. We also think that the computational time can be improved by using Graphical Processing Unit (GPU) or High Performance Computer (HPC). Therefore, our future work will be included as stated such as:

- To use WEKA GPU to run deep learning of CNN algorithm for better computational time.
- To investigate the results if more hidden layers in CNN like 15, 30 or more layers are added.
- To use another public dataset in another domain such as medical images and other domains.

ACKNOWLEDGEMENTS.

This work is supported by Ministry of Education (MOE), Malaysia, Universiti Teknologi Malaysia (UTM), Malaysia and ASEAN-Indian Research Grant. This paper is financially supported by MYBRAIN, University Grant No. 17H62, 03G91, and 04G48. The authors would like to express their deepest gratitude to all researchers for their support in providing the MNIST, butterfly_owl, dog_cat and AMIDA-13 datasets to ensure the success of this research, as well as School of Computing, Faculty of Engineering for their continuous support in making this research possible.

References

- [1] W. Sun, B. Zheng, and W. Qian, "Automatic feature learning using multichannel ROI based on deep structured algorithms for computerized lung cancer diagnosis," *Comput. Biol. Med.*, no. April, pp. 1–10, 2017.
- [2] K. Alex, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Inf. Process. Syst.*, pp. 1097–1105, 2012.
- [3] Y. LeCun and M. Ranzato, "Deep learning tutorial," *Icml*, 2013.
- [4] P. Noorzad, "Feature Learning with Deep Networks for Image Classification Feature representation : pixels."
- [5] A. Ng, "Deep Learning," pp. 1–80, 2014.
- [6] F. Li, "CS519 : Deep Learning," pp. 1–38, 2016.
- [7] LeCun Yann, Cortes Corinna, and Burges Christopher, "THE MNIST DATABASE of handwritten digits," *Courant Inst. Math. Sci.*, pp. 1–10, 1998.
- [8] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553. pp. 436–444, 2015.
- [9] J. Bouvrie, "Notes on convolutional neural networks," *In Pract.*, pp. 47–60,

2006.

- [10] S. L. Phung and A. Bouzerdoum, "Matlab Library for Convolutional Neural Networks," *Vis. Audio Signal Process. Lab Univ. Wollongong*, no. November, pp. 1–18, 2009.
- [11] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 512–519, 2014.
- [12] R. Socher, B. Huval, B. Bhat, C. D. Manning, and A. Y. Ng, "Convolutional-Recursive Deep Learning for 3D Object Classification," *Adv. Neural Inf. Process. Syst.* 25, no. i, pp. 665–673, 2012.
- [13] G. Montavon, "A Machine Learning Approach to Classification of Low Resolution Histological Samples," 2009.
- [14] V. K. Kakade and N. Kumari, "Based On Convolutional Neural Network," pp. 3306–3311, 2016.
- [15] P. Kainz, M. Pfeiffer, and M. Urschler, "Semantic Segmentation of Colon Glands with Deep Convolutional Neural Networks and Total Variation Segmentation," pp. 1–15, 2015.
- [16] N. Wahab, A. Khan, and Y. S. Lee, "Two-phase deep convolutional neural network for reducing class skewness in histopathological images based breast cancer detection," *Comput. Biol. Med.*, vol. 85, no. April, pp. 86–97, 2017.
- [17] N. Tajbakhsh and K. Suzuki, "Comparing two classes of end-to-end machine-learning models in lung nodule detection and classification: MTANNs vs. CNNs," *Pattern Recognit.*, vol. 63, no. October 2016, pp. 476–486, 2017.
- [18] J. Ma, F. Wu, J. Zhu, D. Xu, and D. Kong, "A pre-trained convolutional neural network based method for thyroid nodule diagnosis," *Ultrasonics*, vol. 73, pp. 221–230, 2017.
- [19] H. Su, F. Liu, Y. Xie, F. Xing, S. Meyyappan, and L. Yang, "Region segmentation in histopathological breast cancer images using deep convolutional neural network," *2015 IEEE 12th Int. Symp. Biomed. Imaging*, pp. 55–58, 2015.
- [20] Z. Zainudin, S. M. Shamsuddin, S. Hasan, and A. Ali, "Convolution Neural Network for Detecting Histopathological Cancer Detection," *Adv. Sci. Lett.*,

vol. 24, no. 10, pp. 7494–7500, 2018.

- [21] I. H. Witten, “Data Mining with Weka (Class 3) - 2013,” 2013.
- [22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software,” *ACM SIGKDD Explor.*, vol. 11, no. 1, pp. 10–18, 2009.
- [23] E. Frank *et al.*, “Weka-A Machine Learning Workbench for Data Mining,” in *Data Mining and Knowledge Discovery Handbook*, 2009, pp. 1269–1277.
- [24] E. Frank *et al.*, “WEKA: A Machine Learning Workbench for Data Mining,” *Springer US. Data Min. Knowl. Discov. Handb.*, pp. 1305–14, 2005.
- [25] T. Credit and C. Screening, “Data Mining A Tutorial-Based Primer Chapter Five using WEKA,” pp. 1–12, 2011.
- [26] E. On, “Multi-Task Learning on Mnist Image Datasets,” in *ICLR2018*, 2018, no. 1998, pp. 1–7.
- [27] Y. Lecun, C. Cortes, and C. J. C. Burges, “The MNIST Database,” *Courant Institute, NYU*, 2014. [Online]. Available: <http://yan.lecun.com/exdb/mnist>.
- [28] L. Deng, “The MNIST database of handwritten digit images for machine learning research,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.