ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse256952020.pdf
https://doi.org/10.30534/ijatcse/2020/256952020

Performance Evaluation of Convolutional Neural Networks and Optimizers on Wildlife Animal Classification

Johanes M.B Latupapua[1], Iman Herwidiana Kartowisastro[2,3]
[1] Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta,11480, Indonesia, johanes.latupapua@binus.ac.id
[2]Computer Science Department, BINUS Graduate Program - Doctor of Computer Science, Bina Nusantara University, Jakarta, 11480, Indonesia, imanhk@binus.edu
[3]Computer Engineering Department, Faculty of Engineering, Bina Nusantara University, Jakarta, 11480, Indonesia, imanhk@binus.edu

ABSTRACT

Selecting optimizer and hyperparamater settings are able to affect accuracy achieved significantly. Selected values are generally performed by trial and error. This study evaluates performance of Convolutional Neural Network (CNN) and two optimizers combined with changes in 6 learning rate values. Researched optimizers are Adaptive Moment Estimation and RMS Prop in wildlife animal classification domain. Other hyperparameter values are set same. The architectures are implemented on DenseNet 121, ResNet 50 and AlexNet. This study recommends best value of learning rate for training process which will be conducted with conditions similar to this study and provide other insights during study. The images in this study used wild animals with various positions from front side, back side and some part of bodies that describe real condition. In contradiction to other studies which generally use images in the hundreds of thousands to millions, the number of images used is 47, 841 taken from the Serengeti Season 1 Snapshot with imbalance conditions. Obtained result with the highest recall reached 79%.

Key words: Convolutional Neural Networks, animal classification,learning rate, accuracy.

1. INTRODUCTION

Object recognition using the Convolutional Neural Network (CNN) continues increasing along with development of deep learning in the world. Various objects in the form of images have different characteristics which have attracted many researchers to create recognition that can achieve high accuracy. CNN architecture has continued to evolve since Krizhevsky et. al[1]were able to create an AlexNet architecture which recognizes objects in ImageNet. It is a milestone in the revival of deep learning. Afterwards, the results of other studies invented new architectural variants such as Inception [2], ResNet[3], DenseNet[4] and other architectures.

The recognition of animals and plants in nature using camera trap output is an area which can be resolved by CNN [5], [6]. This process eases ecologists, biologists to identify animals and plants in an area, including endangered animals [7], [8],[9]. Furthermore, ecologist research can be focused on monitoring animal habits or plant growth [10]. This is one of goals of Serengeti Snapshot project as one of the largest collection projects of wild animal images in the world [11]. In recent years, there have been several projects that collected animal and plant images in several places in the world with camera traps such as Wildlife Spotter, Zooniverse, Mammalweb, Instantwild[12]. The identification process involves many communities in society.

To obtain high accuracy, selection of CNN architecture or design will determine beside other factors such as number of images in training and testing datasets, preprocessing process, determining of hyperparameters for training, normalization, regularization and optimization techniques. This study evaluated state of the art CNN architecture by focusing on the impact of changing hyperparameter values to accuracy. Architectures selected wereAlexNet, ResNet 50 and DenseNet 121. The reasons of selecting these 3 architectures wereAlexNet as an architecture which has been used by previous studies [9], [8]; ResNet 50 has the number of layers more than AlexNet and has been used in previous studies [5], [13]; DenseNet 121 is a fairly lightweight architecture with not as many parameters as ResNet, so that training process will be faster [4].

This study specifically discussed the configuration of learning rate in optimizer as a training hyperparameter to achieve high accuracy. Learning rate will control how fast the model learns. In their book, Goodfellow et. al [14] wrote learning rate might be the most important hyperparameter if we have time to tune one hyperparameter only. It cannot do exact calculation to obtain the optimal learning rate for existing models and datasets. The biggest challenge is determining hyperparameter values used in an architecture [15]. A trial and error process must be performed.

This study used Adaptive Moment Estimation (Adam) and RMSProp as a gradient descent optimization algorithm that depended on learning rate value, applied to 3 CNN architectures so that it can provide recommendations for learning rate values in animal classification, particularly in wildlife animal. The selection basis of these two optimizers was that RMSPropis able to work fast for online nonstationary by maintaining the moving average of square gradient. Adam was selected because it can accelerate the updating of parameters [16], it is widely used in current research and this optimizer is still being improved with the emergence of Adam variants [17], [18]. Adam and RMSProp are able to provide faster and more stable convergence to achieve high accuracy values [16], [19]. Both optimizers work in wildlife animal classification where image characteristics vary widely and have been studied by many researchers in biology and information technology in recent years [5], [7],[10], [12]. Animal images were taken from front view or various other positions present challenges at research to achieve the best possible accuracy. The camera trap method was used to take photos of animals at several points in a certain area within a certain period of time [20]. Several studies on the accuracy calculation were implemented wildlife animal classification have been conducted. Details will be described more in Section II.

According to description above, the authors proposed the performance evaluation study of 3 state of the art CNN architectures, namely AlexNet, ResNet 50 and DenseNet 121 with changes in learning rate values using Adam and RMSProp optimizers on wildlife animal classification. From this study, it is expected that the results will produce knowledge when selecting learning rate, optimizer and architecture to achieve high accuracy, especially in animal classification.
submission.

2. RELATED WORKS

A study by Trnovsky et. al [21] examined classification of five animal species namely wolf, fox, bear, pig and deer. Their study compared CNN with Support Vector Machine (SVM), Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Local Binary Patterns Histograms (LBPH). A total of 500 test pictures with 100 of each type with pictures of 5 types of animals were taken from the front view only. The highest accuracy in each class with CNN was at 91 - 97%.

With a greater number of classifications and varied wildlife animal positions, Nguyen et. al [9] conducted a study with distribution of each type significantly different or imbalance class. Totally there were 18 types of animals with a total number of 72,498 images from the Wildlife Spotter dataset where 80% composition for training and 20% for validation. Animal positions were taken from various perspectives in wild nature. This team compared 3 CNN architectures namely Lite AlexNet, VGG and ResNet 50 using Adam optimizer. At this experiment, the VGG-16 architecture achieved the highest accuracy of 96.6% for detecting animals, and classifying animals achieved the highest accuracy of 96.6% by ResNet 50.

Gomez et. al [8] also used Serengeti Snapshot dataset for 26 animal species by creating a balanced class of 26,000 training images and 6240 test images. The 8 architectures used wereAlexNet, VGG Net, GoogLenet, ResNet 50, ResNet 101, ResNet 152, AlexNet FT and GoogLenet FT. The dataset was divided into 4 groups that were imbalances, balances, foreground and segmented. Top-1 and Top-5 measurements were performed for animal recognition. Consecutively, from smallest to highest level of accuracy results were imbalances, balances, foreground and segmented. Overall tests with an accumulation of 4 different dataset groups, the accuracy results on ResNet 101 were able to reach the highest accuracy with 98.1%.

Another study which is similar to Gomez et. al using same source dataset, were conducted by Norouzzadeh et. al [5]. The focus of dataset selection is at images with 1 animal only while having various animal positions. They used AlexNet, Network in Network, VGG, GoogLenet, ResNet 18, ResNet 34, ResNet 50, ResNet 101 and ResNet 152. Optimizerapplied Stochastic Gradient Descent (SGD) where learning rate worked from 0.001 to 0.0001 that were adjusted following range of epoch numbers. The highest accuracies of Top 1 and Top 5 with the same values were achieved by ResNet 101 and ResNet 152 at 93.8% and 98.8%.

Wili et. al [10] compared accuracy results with transfer learning and without it, which took 3 dataset sources coming from Serengeti Snapshot, Camera Catalog and Elephant Expedition. Architecture used ResNet 18 by performing 2 tasks, detecting animals and identifying types of animals. For animal detection, the highest accuracies with and without transfer learning were at 97.7% and 98.0% respectively using Camera Catalog. For animal identification, the highest accuracies with and without transfer learning were at 96.6% and 92.7 respectively using Elephant Expedition.

Another interesting study using Deep CNN (DCNN) with a focus on cows only was conducted by Zin et. al [22]. They took 45 cows whose data were recorded in video with a frame rate of 30 fps. Images were taken from live video in either a full or partial image of cow body, either from top or front view. The DCNN architecture used was quite simple with 1 input layer, 3 convolutional - pooling, 1 fully connected layer and 1 output layer. Accuracy result in training produced 98.87% and in testing accuracy of 97.01%, that were implemented for a full body cow. For partial bodies, the score was 86.8%. Zin et. al used SGD optimizer with a learning rate of 0.0001.

Chen et. al [7] implemented DCNN using 3 convolutional and 3 max pooling. Chen et. al compared DCNN with Bag of visual Words (BoW) based on LDA using 20 classes of wild animals' dataset in North America with a total of 14,346 training images and 9,530 test images. The result was 38,315%, still far below other studies.

DCNN study was also conducted by Verma et. al [23] using 5 convolutional - pooling and 3 fully connected layers, similar to AlexNet and used SGD optimizer. The dataset took the same dataset with Chen et. al [7]. A total of 1110 images of which were 90% for training and 10% for testing. This study also measured SVM and KNN models, where DCNN accuracy reached 91%.
Tabak et. al [6] undertook to use over 3 million images as a dataset from 5 location in US, Canada and Serengeti Snapshot, Tanzania. It was implementing generalization where Canada dataset was carried out for testing only, the other two were for training and testing. This classification used ResNet 18 with a Macinstosh laptop 16 GB RAM. Optimization implemented SGD with Momentum while taking learning rates from 0.01 - 0.0001. The result of training accuracy reached 98% for US dataset, and validation result were 82% from Canada and 94% from Tanzania.

Another generalization study was conducted by Scheneider et. al [24] with a total dataset of 47,000 from 5 locations with 5 architectures, namely DenseNet, Inception ResNet, Inception v3, MobileNet and Exception. Scheneider et. al used Adam optimizer. Highest accuracy was at 95.6% (Top 1) with DenseNet 201 for the same testing dataset source during training. The highest accuracy was 68.7% (Top 1) with DenseNet 201 for different testing dataset sources during training.

A study using gradient class activation procedure was to extract the most prominent pixels in the last convolutional layer by Miao et. al [13]. Dataset was taken from wild animals also with 20 classes from Gorongosa, Mozambique. The architecture used VGG and ResNet 50 where the highest accuracy is at 87.5% with ResNet 50.

Schneider et. al [25] compared the YOLO v2 and Faster RCNN applications using Serengeti Gold Standard Snapshot of 4,432 images and the Reconyx Camera of 7,193 images. The highest accuracy was achieved by YOLO v2 and Faster RCNN at 65.0% and 93.0%, respectively.

In application of animal detection and monitoring which were integrated with other systems, Hansen et. al [26] developed for monitoring of pigs; Guzhva et. al [27], Rivas et. al [28] for monitoring of cattle; monitoring with animal scanner software by Yousif et. al [29], and road situation monitoring was developed by Antonio et. al [30].

3. METHODOLOGY

3.1 Proposed Method

This study evaluated and analyzed impact of changes in learning rate values on accuracy using Adam and RMSProp optimizers in animal classification. The implementation was performed using 3 state of the art CNN architectures, namely DenseNet121, ResNet50 and AlexNet. The Serengeti Snapshot dataset [11] is an imbalance class which is common condition recently [31], [32]. The technique to reduce imbalance was carried out by selecting a class that has a minimum of 1000 images [8], [31] and applying data augmentation [5], [6], [24] before the dataset was loaded into data loader. The domain of wildlife animal classification was selected because this study would help ecologists, biologists or environmentalists to identify classes and research the habits of animals in natural environment [5] - [7], [9], [10].

The idea to evaluate performance of Adam and RMSProp were referring to Goodfellow's et. al book [14]. There is no consensus to determine any value of learning rate at each training. Learning rates which has a range of 0 to 1 are used in adaptive learning rate optimizer setting. Hyperparameter values are very crucial in training [15], [33]. Adam and RMSProp are able to provide faster and more stable convergence to achieve high accuracy values [16], [19]. For example, both were able to achieve the highest accuracy compared to other optimizers on LFW dataset with smaller losses [33]. There were 6 tested learning rate values starting from 0.1, decreasing to one-tenth out of previous value, namely 0.1; 0.01; 0.001, 0.0001, 0.00001 and 0.000001. The study uses range values as following Bengio's research recommendations [34]. A total of 36 scenarios as shown in Table 1.

Table 1: Experiment Scenario

Scenario No	Learning Rate	Optimizer	Architecture
1	0.1	Adam	AlexNet
2	0.1	Adam	ResNet 50
3	0.1	Adam	DenseNet 121
4	0.1	RMSProp	AlexNet
5	0.1	RMSProp	ResNet 50
6	0.1	RMSProp	DenseNet 121
7	0.01	Adam	AlexNet
8	0.01	Adam	ResNet 50
9	0.01	Adam	DenseNet 121
10	0.01	RMSProp	AlexNet
11	0.01	RMSProp	ResNet 50
12	0.01	RMSProp	DenseNet 121
13	0.001	Adam	AlexNet
14	0.001	Adam	ResNet 50
15	0.001	Adam	DenseNet 121
16	0.001	RMSProp	AlexNet
17	0.001	RMSProp	ResNet 50
18	0.001	RMSProp	DenseNet 121
19	0.0001	Adam	AlexNet
20	0.0001	Adam	ResNet 50
21	0.0001	Adam	DenseNet 121
22	0.0001	RMSProp	AlexNet
23	0.0001	RMSProp	ResNet 50
24	0.0001	RMSProp	DenseNet 121

25	0.00001	Adam	AlexNet
26	0.00001	Adam	ResNet 50
27	0.00001	Adam	DenseNet 121
28	0.00001	RMSProp	AlexNet
29	0.00001	RMSProp	ResNet 50
30	0.00001	RMSProp	DenseNet 121
31	0.000001	Adam	AlexNet
32	0.000001	Adam	ResNet 50
33	0.000001	Adam	DenseNet 121
34	0.000001	RMSProp	AlexNet
35	0.000001	RMSProp	ResNet 50
36	0.000001	RMSProp	DenseNet 121

A few parameter values were set same. Batch size was 32, epoch was 55 and loss function used Cross Entropy loss because of multi-class. The batch size value followed [34]. The epoch value was adopted from studies by Norouzaddeh et. al [5] and Tabak et. al [6]. Optimizer momentums took default value, which were decay rate β1 of 0.9, decay rate β2 of 0.999 in Adam and decay rate γ of 0.9 in RMS Prop.

This study applied early stopping as a regularization technique to reduce overfitting [34], [35]. This technique stopped training when overfitting began. In training and validation loops, early stopping was put with patience value of 20. Patience is the maximum number of epochs to retain training and validation when validation loss continued increasing or was bigger than the smallest validation loss.

The system was implemented using PyTorch library and run on Floydhub (www.floydhub.com), one of cloud-based deep learning platforms which commonly used today. The authors used 2 types of Nvidia GPU products, those are Tesla K80 RAM 16 GB and V100 RAM 16 GB. Complete flow was shown in Figure 1.

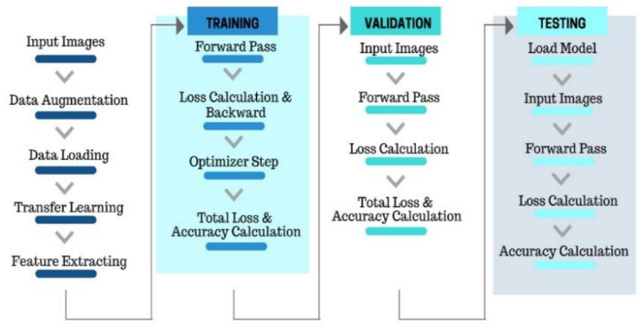

Figure 1: System Flow

Percentages of number of training, validation and testing images were 80%, 10% and 10%. Input file size was set to 224 x 224 adjusted to AlexNet, Resnet 50 and DenseNet 121 input sizes. Each image was located at training, validation and testing folders was transformed as data augmentation such as random horizontal flip, vertical flip, random crop, color jitter and normalize. Random crop and flip are 2 transform types which have the highest accuracy [36]. Later, dataloader ran to load customized images. Training dataloader worked at first, then was followed by validation dataloader.

The authors ran transfer learning which was started by pretrained to ImageNet [5], [6], [8] -[10], with feature extracting at each architecture. The system would download and update parameters before they started training to animal dataset. At this point, the system already had a model at beginning. After being pretrained, feature extracting was carried out. It only took gradients without initial parameters from pretrained result and modified the final output layer according to classes number and architecture types.

In training, forward pass was performed, then ran backpropagation as well as calculating loss. Training would stop within specified counter limit when validation loss was greater than the smallest validation loss from previous epoch. When ran step optimizer, parameter updates were done. At the end of training, system calculated total loss and accuracy. Then it was repeatedly starting from forward pass onwards. In validation with evaluation mode, hyperparameter tuning is performed to optimize the best setting of training. Within validation and testing, their sequences were almost same with forward pass, however without backpropagation. Losses and accuracy were accumulated as per one epoch.

In classification, each testing will calculate the accuracy and might be provided into confusion matrix. As shown in Table 2, confusion matrix mapped actual and prediction values to binary classes[37], [38], [39]. True positive (TP) is number of correctlyactual data, which are predicted ascorrect, false positive (FP) is number of incorrectlyactual data which are predicted as correct. False negative (FN) is number of incorrectlyactual data which are predicted as incorrect and true negative (TN) is number of correct actual datawhich are predicted as incorrect.

Table 2: Confusion Matrix

Confusion Matrix		Actual	
		True	False
Predicted	True	TP	FP
	False	FN	TN

For multiple classes, the actual conditions for the predictions in each class will be obtained. From the performance metrics of this research, the following is the formula for each metric in Eq. (1) - (3) relates actual to prediction [38], [40].

1. Precision = $\frac{TP}{TP+FP}$ (1)

The number of predictions that were correct out of total number of predictions.

2. Recall = $\frac{TP}{TP+FN}$ (2)

The number of predictions that are exactly correct from actual correct total number.

3. F1-score = $\frac{Precision\ x\ Recall}{(Precision+\ Recall)} = \frac{2TP}{2TP+FP+FN}$ (3)

The F1-score shows the weighted harmonic mean ratio of weighted average of precision and recall.

The metrics used in this study are overall accuracy, average precision, average recall, average F1-score with classification reports and confusion matrix. In imbalance class, precision, recall and F1-score represented more appropriate as an accuracy of prediction[37]. This performance metrics in animal classification were also implemented by Tabak et. al [6] and Schneider et. al[24]. The authors generated classification report on program that are provided precision, recall, F1-score information per class.

3.2 Dataset

Snapshots Serengeti is very huge that currently has reached around 3.2 million images from 11 seasons [11]. Of the total, it was only 20% having animal objects. The authors took Season 1 dataset only. In preprocessing before program was executed, the authors curated images with 3 steps:

1. Separated animal pictures or without animals.
2. Took pictures at morning, afternoon and evening only.
3. Took a class with a minimum number of 1000 of images.

Table 3: Class Detail

Class	Total	Training	Validation	Testing
gazelleThomsons	25012	20011	2501	2500
gazelleGrants	4139	3312	413	414
zebra	3882	3106	388	388
guineaFowl	3152	2524	314	314
warthog	2546	2037	254	255
hartebeest	1884	1508	188	188
giraffe	1553	1243	155	155
lionFemale	1226	982	122	122
hyenaSpotted	1179	944	118	117
buffalo	1147	918	114	115
elephant	1121	897	112	112
Total	**46841**	**37482**	**4679**	**4680**

The authors used 46,841 images which were previously selected manually with details per class in Table 3.

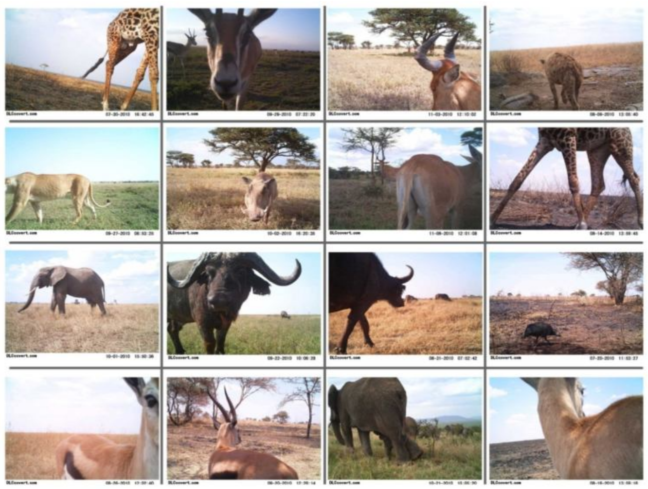
Figure 2: Animals from Snapshot Serengeti Dataset

The total number of classes were 11 classes and imbalance. Many images that were taken, showed incomplete full body. However, this condition became a research challenge. The taking angle was taken from few positions as shown in Figure 2.

4. RESULT AND DISCUSSION

In multi-class imbalance condition, performance metric measures precision, recall and F1-score. All were measured with weight (weighted). The weighted of each class was different following number of images in each class according to Table 3. Based on the scenario in Table 1, it was obtained weighted precision, weighted recall and weighted F1-score as shown in Table 4 with the lowest value of 0 to the highest value of 1. At this condition, accuracy was same with weighted recall on Recall column.

Table 4: Result of All Scenario

Scenario No	Precision	Recall	F1-score	Early Stopping
1	0.64	0.59	0.59	Yes
2	0.29	0.53	0.37	Yes
3	0.72	0.67	0.66	Yes
4	0.67	0.52	0.56	Yes
5	0.29	0.53	0.37	Yes
6	0.81	0.47	0.53	Yes
7	0.65	0.55	0.57	Yes
8	0.29	0.53	0.37	Yes
9	0.75	0.73	0.69	Yes
10	0.71	0.61	0.59	Yes
11	0.29	0.53	0.37	Yes
12	0.74	0.69	0.66	Yes
13	0.7	0.64	0.65	Yes
14	0.77	0.77	0.75	Yes
15	0.76	0.76	0.75	Yes
16	0.69	0.69	0.64	Yes
17	0.75	0.74	0.7	Yes
18	0.75	0.75	0.72	Yes
19	0.72	0.71	0.68	Yes
20	**0.79**	**0.79**	**0.77**	**Yes**
21	0.76	0.76	0.75	No
22	0.71	0.71	0.69	Yes
23	0.78	0.78	0.77	Yes
24	0.75	0.76	0.74	Yes
25	0.71	0.69	0.66	No
26	0.73	0.73	0.69	Yes
27	0.73	0.73	0.7	No
28	0.69	0.68	0.64	Yes
29	0.73	0.73	0.7	Yes

30	0.73	0.73	0.71	No
31	0.65	0.64	0.57	No
32	0.53	0.59	0.46	Yes
33	0.44	0.57	0.43	Yes
34	0.6	0.6	0.49	Yes
35	0.46	0.59	0.46	Yes
36	0.45	0.57	0.43	Yes

From Table 4, the highest accuracy was achieved by ResNet 50 using Adam with learning rate of 0.0001 to 0.79. Learning rates are grouped to know the accuracy achieved; it is shown in Figure 3.

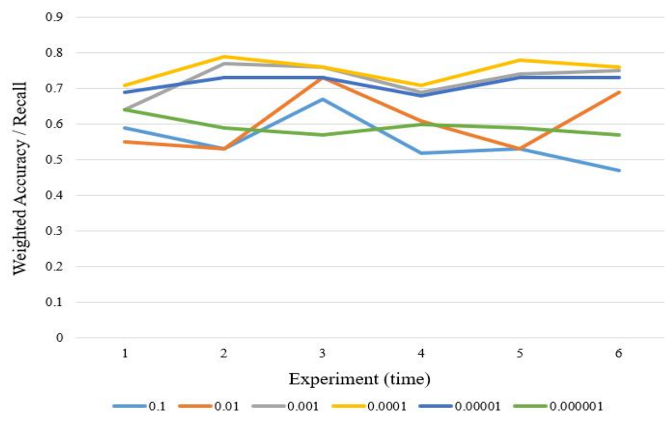

Figure 3: Learning Rate Against Recall

The highest accuracy was achieved by learning rate of 0.0001 then followed by learning rate of 0.001. If accuracies were grouped per optimizer, graphs can be shown Figure 4 and Figure 5.

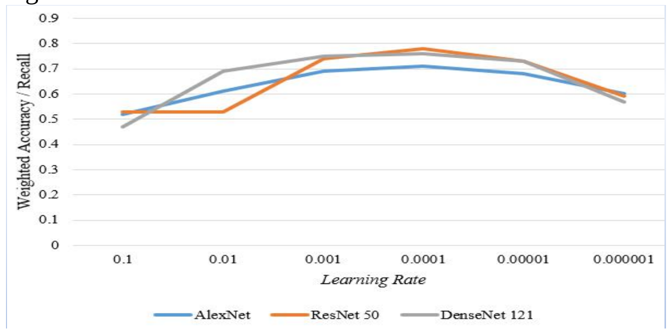

Figure 4: Learning Rate Against Recall using Adam

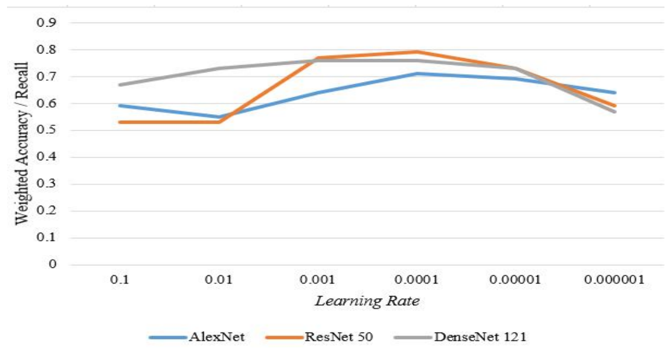

Figure 5: Learning Rate Against Recall using RMSProp

Taken from Figure 4 and Figure 5, using either Adam or RMS Prop, learning rate of 0.0001 reached the highest accuracy with ResNet 50 which was followed by DenseNet 121.
In other side, lowest accuracy was achieved using Resnet 50 also at learning rates of 0.1 and 0.01. It could be shown in detail per class in classification report issued by program. Table 5 is ResNet 50 classification report with Adam, learning rate 0.01.

Table 5: Classification report of Resnet 40, Adam, Learning Rate of 0.01

Class	Precision	Recall	F1-score	Support
buffalo	0	0	0	115
elephant	0	0	0	112
gazelleGrants	0	0	0	414
gazelleThomsons	0.53	1	0.7	2500
giraffe	0	0	0	155
guineaFowl	0	0	0	314
hartebeest	0	0	0	188
hyenaSpotted	0	0	0	117
lionFemale	0	0	0	122
warthog	0	0	0	255
zebra	0	0	0	388
accuracy	-	-	0.53	4680
macro average	0.05	0.09	0.06	4680
weighted average	0.29	0.53	0.37	4680

ResNet 50 was able to do learning 1 class only, gazelleThomsons. The other ten classes failed with class distribution in Table 5, even though these 10 classes have images at least 1000 respectively. Support column showed number of images per class. GazelleThomsons dominated 53% of total images.

Second example were taken from recall 0.79 with ResNet 50, using Adam with learning rate of 0.0001 in Table 6. In Table 6, it could be shown that model was able to do learning for all classes.

Table 6: Classification Report of Resnet 50, Adam, Learning Rate of 0.0001

Class	Precision	Recall	F1-score	Support
buffalo	0.67	0.78	0.72	115
elephant	0.87	0.78	0.82	112
gazelleGrants	0.71	0.32	0.44	414
gazelleThomsons	0.81	0.95	0.87	2500
giraffe	0.93	0.6	0.73	155
guineaFowl	0.82	0.65	0.73	314
hartebeest	0.73	0.61	0.67	188
hyenaSpotted	0.92	0.61	0.73	117

lionFemale	0.82	0.56	0.66	122
warthog	0.63	0.65	0.64	255
zebra	0.7	0.74	0.72	388
accuracy	-	-	0.79	4680
macro	0.78	0.66	0.7	4680
weighted	0.79	0.79	0.77	4680

In multi classes, confusion matrix is a dimensional matrix of class number. Using Scikit-learn library, confusion matrix could be generated. It was taken an example of result from ResNet 50, Adam, learning rate of 0.0001 in Table 7 by showing the number of images on each box.

Class A to K are buffalo, elephant, gazelleGrants, gazelleThomsons, giraffe, guineafowl, hartebeest, hyena Spotted, lionFemale, warthog and zebra, respectively. The value in each box indicated number of images were predicted by a class that is right as its class. The authors took an example to calculate precision, recall and F1-score from class H (hyenaSpotted). As shown in Table 2, based on meaning of true positive (TP), false negative (FN), false positive (FP) and true negative (TN) in Section 3.1 Proposed Method and Singh[39] are

1. True positive is value resulted from meeting of vertical line as predicted class with horizontal line as actual class. True positive of each class forms a diagonal line from top left corner to the bottom right corner of table, which are indicated on yellow box.
2. False negative is sum of all values in one class prediction line minus true positive, indicated on green box.
3. False positives are sum of all values in actual column of class minus true positive, indicated on blue box.
4. True negative is sum of all values in table minus true positive, false negative and true negative.

According to these 4 points, TP of 71; FN of 46; FP of 6 and TN of 4557. Then we could obtain precision, recall and F1-score in Classification Report.

Table 7: Confusion Matrix of Resnet 50, Adam, Learning Rate of 0.0001

Class		**Actual**										
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
Predicted	**A**	90	3	0	3	0	2	1	0	0	11	5
	B	3	87	0	5	3	1	4	0	0	6	3
	C	1	0	131	247	0	2	9	0	4	6	14
	D	2	0	37	2366	2	15	10	3	3	20	42
	E	3	3	5	34	93	0	4	0	0	2	11
	F	9	2	1	48	2	204	1	1	2	23	21
	G	3	1	7	48	0	3	115	0	1	2	8
	H	1	0	0	24	0	1	0	71	3	14	3
	I	2	1	1	29	0	4	7	2	68	6	2
	J	6	1	1	48	0	12	3	0	2	167	15
	K	14	2	1	70	0	4	3	0	0	6	288

5. CONCLUSION

The authors analysed the performance of 3 architectures combined with Adam and RMSProp optimizers which were focusing on changes of 6 learning rates. According to the authors experiments, the proposed method achieved the highest accuracy of 0.79 using ResNet 50, then was followed by DenseNet 121 with 0.76. Learning rate of 0.0001 was able to provide the highest accuracy in 3 architectures for Adam and RMS Prop optimizers, followed by learning rate accuracy of 0.001. It was getting highest accuracy with learning rate of 0.0001 was same with study by Norouzzadeh et. al [5] and Zin et. al [22] where they applied SGD optimizer. Finally, class domination over other classes will affect the model ability to do learning at learning rate of 0.1 - 0.01, even though other classes have a minimum dataset of 1000 images.

REFERENCES

1. A. Krizhevsky, I. Sutskever and G. E. Hinton, **ImageNet Classification with Deep Convolutional Neural Networks,** Advances in neural information processing systems, pp. 1097-1105, 2012, DOI: https://doi.org/10.1145/3065386.
2. C. Szegedy, V. Vanhoucke, S. Ioffe and J. Shlens, **Rethinking the Inception Architecture for Computer Vision,** in IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 2016, DOI: https://doi.org/10.1109/CVPR.2016.308.
3. K. He, X. Zhang, S. Ren and J. Sun, **Deep Residual Learning for Image Recognition,** in Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 2016, DOI: https://doi.org/10.1109/CVPR.2016.90.

4. G. Huang, Z. Liu and L. van der Maaten, **Densely Connected Convolutional Networks,** in IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, 2017, DOI: https://doi.org/10.1109/CVPR.2017.243.
5. M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. Palmer, C. Packer and J. Clune, **Automatically Identifying, Counting, and Describing Wild Animals in Camera-Trap Images with Deep Learning**, PNAS, 2017, DOI: 10.1073/pnas.1719367115.
6. M. Tabak, M. S. Norouzzadeh, D. W. Wolfson, S. J. Sweeney, K. C. Vercauteren, N. P. Snow, J. M. Halseth, P. A. Di Salvo, J. S. Lewis, M. D. White, B. Teton, J. C. Beasley, P. E. Schlichting and Bought, **Machine Learning to Classify Animal Species in Camera Trap Images: Applications in Ecology,** Methods in Ecology and Evolution, pp. 585-590, 2019, DOI: https://doi.org/10.1101/346809.
7. G. Chen, T. X. Han, Z. He, R. Kays and T. Forrester, **Deep Convolutional Neural Network Based Species Recognition for Wild Animal Monitoring,** in International Conference on Image Processing, Paris, 2014, DOI: https://doi.org/10.1109/ICIP.2014.7025172.
8. A. Gomez, A. Salazar and F. Vargas, **Toward Automatic Wild Animal Monitoring: Identification of Animal Species in Camera Trap Images using Very Deep Convolutional Neural Networks,** Ecological Informatics, pp. 24-32, 2017, DOI: https://doi.org/10.1016/j.ecoinf.2017.07.004.
9. H. Nguyen, T. D. Nguyen, T. Nguyen and S. J. Maclagan, **Animal Recognition and Identification with Deep Convolutional Neural Network for Automated Wildlife Monitoring,** in International Conference on Data Science and Advanced Analytics, Tokyo, 2017, DOI: https://doi.org/10.1109/DSAA.2017.31.
10. M. Willi, R. Pitman, A. Cardoso, C. Locke, A. Swanson, A. Boyer, M. Veldthuis and L. Fortson, **Identifying Animal Species in Camera Trap Images Using Deep Learning and Citizen Science,** Methods in Ecology and Evolution, pp. 80-91, 2018, DOI: 10.1111/2041-210x.13099.
11. A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith and C. Packer, **Snapshot Serengeti, High-Frequency Annotated Camera Trap Images of 40 Mammalian Species in an African Savanna,** Scientific Data, 2015, DOI: 10.1038/sdata.2015.26.
12. S. Green, J. Rees, P. Stephens, R. Hill and A. Giordano, **Innovations in Camera Trapping Technology and Approaches the Integration of Citizen Science and Artificial Intelligence,** Animals, pp. 1-16, 2020, DOI: 10.3390/ani10010132.
13. Z. Miao, K. Gaynor, J. Wang, Z. Liu, M. M. Norouzzadeh, A. McInturff, R. Bowie, R. Nathan, S. Yu and W. Getz, **Insights and Approaches Using Deep Learning to Classify Wildlife,** Scientific Reports, 2019, DOI: 10.1038/s41598-019-44565-w.
14. I. Goodfellow, Y. Bengio and A. Courville, **www.deeplearningbook.org,** MIT Press Book, November 2016. [Online]. Available: www.deeplearningbook.org. [Accessed 10 January 2018].
15. T. Hinz, N. Navarro-Guerrero, S. Magg and S. Wermter, **Speeding Up the Hyperparameter Optimization of Deep Convolutional Neural Network,** International Journal of Computational Intelligence and Applications, 2018, DOI: https://doi.org/10.1142/S1469026818500086.
16. D. Kingma and J. L. Ba, **Adam: A Method for Stochastic Optimization,** in nternational Conference on Learning Representations, San Diego, 2015.
17. L. Luo, Y. Xiong, Y. Liu and X. Sun, **Adaptive Gradient Methods with Dynamic Bound of Learning Rate,** in International Conference on Learning Representations, Lousiana, 2019.
18. U. Khaire and R. Dhanalaskhmi, **High Dimensional Microarray Dataset Classification using an Improved Adam Optimizer (iAdam),** Journal of Ambient Intelligence and Humanized Computing, 2020.
19. S.-H. Park, Y.-B. Bae, B. Fidan and H.-S. Ahn, **Distance-Based Mobile Node Localization of Fixed Beacons Using RMS Prop,** in 58th Annual Conference of The Society of Instrument and Control Engineers of Japan, Hiroshima, 2019, DOI: https://doi.org/10.23919/SICE.2019.8859957.
20. Z. He, R. Kays, Z. Zhang, G. Ning, C. Huang, T. Han, J. Miilspaugh, T. Forrester and W. McShea, **Visual Informatics Tools for Supporting Large-Scale Collaborative Wildlife Monitoring with Citizen Scientists,** IEEE Circuits and System Magazine, pp. 73-86, 2016, DOI: https://doi.org/10.1109/MCAS.2015.2510200.
21. T. Trnovszky, P. Kamencay, R. Orjesek, M. Benco and P. Sykora, **Animal Recognition System Based on Convolutional Neural Network,** Digital Image Processing and Computer Graphics, vol. 15, no. 3, pp. 517-525, 2017, DOI: 10.15598/aeee.v15i3.2202.
22. T. T. Zin, C. N. Phyo, P. Tin, H. Hama and I. Kobayashi, **Image Technology Based on Cow Identification System Using Deep Learning,** in International MultiConference of Engineers and Computer Scientists, Hongkong, 2018.
23. G. K. Verma and P. Gupta, **Wild Animal Detection from Highly Cluttered Images Using Deep Convolutional Neural Network,** International Journal of Computational Intelligence and Applications, 2018, DOI: https://doi.org/10.1142/S1469026818500219.
24. S. Schneider, S. Greenberg, G. Taylor and S. Kremer, **Three Critical Factors Affecting Automated Image Species Recognition Performance for Camera Traps,** Ecology and Evolution, pp. 1-15, 2020, DOI: https://**doi**.org/10.1002/ece3.6147.
25. S. Schneider, G. W. Taylor and S. C. Kremer, **Deep Learning Object Detection Methods for Ecological Camera Trap Data,** in Conference on Computer and Robot Vision, Toronto, 2018, DOI: https://doi.org/10.1109/CRV.2018.00052.
26. M. Hansen, M. Smith, L. Smith, M. Salter, E. Baxter, M. Farrish and B. Grieve, **Towards on-Farm Pig Face Recognition Using Convolutional Neural Networks,**

Computers in Industry, pp. 145-152, 2018, DOI: https://doi.org/10.1016/j.compind.2018.02.016.

27. O. Guzhva, H. Ardo, M. Nilsson, A. Herlin and L. Tufvesson, **Convolutional Neural Network Based on Tracker for Dairy Cows,** Frontiers in Robotics and AI, 2018, DOI: https://doi.org/10.3389/frobt.2018.00107.
28. A. Rivas, P. Chamoso, A. G. Briones and J. M. Corchado, **Detection of Cattle Using Drones and Convolutional Neural Networks,** Sensors, pp. 1-15, 2018, DOI: https://doi.org/10.3390/s18072048.
29. H. Yousif, J. Yuan, R. Kays and Z. He, **Animal Scanner: Software for Classifying Humans, Animals, and Empty Frames in Camera Trap Images,** Ecology and Evolution, pp. 1-12, 2019, DOI: 10.1002/ece3.4747.
30. W. Antônio, M. Da Silva, R. Miani and J. Souza, **A Proposal of an Animal Detection System Using Machine Learning,** Applied Artificial Intelligence, 2019, DOI: https://doi.org/10.1080/08839514.2019.1673993.
31. M. Buda, A. Maki and M. A. Mazurowski, **A Systematic Study of The Class Imbalance Problem in Convolutional Neural Networks,** Neural Networks, pp. 249-259, 2018, DOI: https://doi.org/10.1016/j.neunet.2018.07.011.
32. C. Shorten and K. T. M., **A Survey on Image Data Augmentation for Deep Learning,** Journal of Big Data, 2019, DOI: 10.1186/s40537-019-0197-0.
33. D. Soydaner, **A Comparison of Optimization Algorithms for Deep Learning,** International Journal of Pattern Recognition and Artificial Intelligence, 2019, DOI: https://doi.org/10.1142/S0218001420520138.
34. Y. Bengio, **Practical Recommendations for Gradient-Based Training of Deep Architectures,** in Neural Networks: Tricks of the Trade, Berlin, Springer, 2012, pp. 437-478, DOI: https://doi.org/10.1007/978-3-642-35289-8_26.
35. L. Prechelt, **Early Stopping But When?,** in Neural Networks: Tricks of the Trade, Berlin, Springer, 2012, pp. 53-67, DOI: https://doi.org/10.1007/978-3-642-35289-8_5.
36. L. Taylor and G. Nitschke, **Improving Deep Learning with Generic Data Augmentation,** in IEEE Symposium Series on Computational Intelligence, Bangalore, 2018, DOI: https://doi.org/10.1109/SSCI.2018.8628742.
37. M. Sokolova and G. Lapalme, **A Systematic Analysis of Performance Measures for Classification Tasks,**Informations Processing and Management, pp. 427-437, 2009, DOI: https://doi.org/10.1016/j.ipm.2009.03.002.
38. A. Luque, A. Carrasco, A. Martin and A. de las Heras, **The Impact of Class Imbalance in Classification Performance Metrics Based on The Binary Confusion Matrix,** Pattern Recognition, pp. 216-231, 2019, DOI: https://doi.org/10.1016/j.patcog.2019.02.023.
39. N. Nasharuddin, N. Yusoff and S. Ali, **Multi-Feature Vegetable Recognition using Machine Learning Approach on Leaf Images,** International Journal of Advanced Trends in Computer Science and Engineering, vol. VIII, no. 4, pp. 1789-1794, 2019, DOI:https://doi.org/10.30534/ijatcse/2019/110842019.
40. H. Gadade and D. Kirange, **Machine Learning Approach Towards Tomato Leaf Disease Classification,** International Journal of Advanced Trends in Computer Science and Engineering, vol. IX, no. 1, pp. 490-495,2020, DOI: https://doi.org/10.30534/ijatcse/2020/67912020.
41. S. Singh, **Confusion Matrix in Machine Learning,** Medium, 14 December 2018. [Online]. Available: https://medium.com/@shubhanshi.shubh860/confusion-matrix-in-machine-learning-cd7333d72f5.