JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 8 Number 1 | June 2023: 52-60
DOI: 10.15575/join.v8i1.1007

Catbreedsnet: An Android Application for Cat Breed Classification Using Convolutional Neural Networks

Anugrah Tri Ramadhan[1], Abas Setiawan [2]
[1]Department of Informatics Engineering, Universitas Dian Nuswantoro, Indonesia
[2]Department of Computer Science, Faculty of Mathematics and Natural Science, Universitas Negeri Semarang, Indonesia

Article Info

Article history:

Received Dec 29, 2022
Revised May 15, 2023
Accepted May 11, 2023

Keywords:

Accuracy
Cat breeds
Convolutional Neural Network
MobilenetV2
Transfer Learning

ABSTRACT

There are so many cat races in the world. Ignorance in recognizing cat breeds will be dangerous if the cat being kept is affected by a disease, which allows mishandling of the cat being kept. In addition, many cat breeds have different foods from one race to another. The problem is that a cat caretaker cannot easily recognize the cat breed. Therefore, technology needs to help a cat caretaker to treat cats appropriately. In this study, we proposed a Machine Learning approach to recognize cat breeds. This study aims to identify the cat breed from the cat images then deployed on an Android smartphone. It was tested with data from cat images of 13 races. The classification method applied in this study uses the Convolutional Neural Network (CNN) algorithm using transfer learning. The base models tested are MobilenetV2, VGG16, and InceptionV3. The results tested using several models and through several experimental scenarios produced the best classification model with an accuracy of 82% with MobilenetV2. The model with the best accuracy is then embedded in an application with the Android operating system. Then the application is named Catbreednet.

Corresponding Author:

Abas Setiawan,
Department of Computer Science, Faculty of Mathematics and Natural Science, Universitas Negeri Semarang,
D5 Building Second Floor, Matematika dan IPA Faculty, Sekaran, Gunung Pati, Semarang, Indonesia
Email: abas.setiawan@mail.unnes.ac.id

1. INTRODUCTION

Biologists have classified cat races into several types, from those that humans usually adopt, such as the Angora, Persian, Bengal, and others. Besides, there are classified as wild and cannot be raised by humans, such as tigers [1]. However, considering so many types of cats, it won't be easy to distinguish even manually using the human eye. It is because some cats have a nearly similar physical appearance.

Some people who keep cats do not know their pet cat's breed, which can be fatal because each type of cat must have treatment according to its type when exposed to an illness [2]. Therefore, it is necessary to know the breed of cats so that the maintenance of cats can be adjusted according to their species, adjustments to food, size of cages, routine trimming of fur, and actions if there is a disease in cats. It's another case with wild cats like tigers, they must also be treated more carefully, and not just anyone can do it. That is why there needs to be a system that can automatically classify each cat breed so that each treatment is appropriate.

With the development of increasingly advanced technology, animal classification can be solved by classifying images, which is one of the machine learning techniques [3]–[6]. As in [7], the improvisation of the conventional Local Binary Pattern (LBP) and Histogram of Oriented Gradient techniques resulted in an accuracy of 79.25% using the CNN model, which resulted in an accuracy of 96.75% in the Classification of 133 dog breeds. Besides, another recent study [8] proved that using transfer learning using RestNet50 could produce better results, resulting in an accuracy of 93.53 % and 90.86 % on different datasets on the Classification of dog breeds. Then the VGGNet [9] was used, with an error rate of 7.3%. In addition, to transfer learning, there is also a fine-grained classification technique, which can classify primary and secondary, primary for animal species, and secondary for animal races for Classification of animals through an installed camera trap [10].

The VGG base model again has been used for cat breeds and achieved an accuracy of 90%. Besides, only used five classes of cat breeds for the training dataset [11]. Another approach is recognizing the cat breed for Maine Coon and European Shorthair based on their voice [12]. Two methods have been proposed to train that acoustic data: k-Nearest Neighboor and Multilayer perceptron. Karlita *et al.* classified cat breed with the EfficientNet-B0 pre-trained model, which resulted in more than 90% accuracy but only for the selected nine types of cats [5]. The high accuracy of 95% was proposed by Qatrunnada *et al.* using Xception Architecture, but it was only trained for five classes [13].

Because transfer learning has successfully handled image classification, it is also applied to our work. We use three base model scenarios, such as MobileNetV2 [11], VGGNet [9], and InceptionV3 [12]. This study has two contributions. First, we create a neural network architecture with the base model on our CNN model to recognize the 13 classes cat breed images. Second, the model that best fits the problems in this study will be selected to be deployed on the Android application so that the user can easily use it. In addition, we hoped that applying the proposed method to the dataset would gain good accuracy and help solve problems related to cat breed classification, such as treating cats according to their respective breeds.

2. METHOD

2.1. *Research Stages*

The flow of research methodology is a stage in conducting research that will be carried out, with the aim that the research carried out later does not deviate from the planned objectives. Figure 1 depicts the research methodology. First is data preparation, namely the dataset preparation process. The dataset in this research uses cat image data, totaling 13 classes. Second, data preprocessing is the process of processing data according to needs after the data has been successfully collected. In the study, the data obtained is then organized according to each class. After the data is ready in each category, it is continued by resizing the image to 224×224. The third is the data augmentation process, which uses image-processed dataset additions by adding data to the training data without allocating memory on the user's computer. Fourth, build the CNN architecture that will be used, either by making the own model or using the base model. Fifth, after creating the desired model architecture, training or the algorithm training process is carried out to produce the desired parameters according to what is needed for image classification. Sixth is the evaluation and deployment, conducted by measuring the algorithm's success in classifying cat breed images. After selecting the best model, the final step is the deployment process on the Android platform.

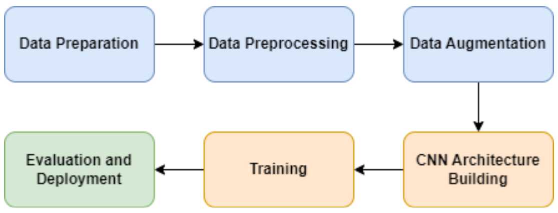

Figure 1. Cat Breed Classification Methodology

2.2. *Data Preparation*

The dataset in this study was obtained using data crawling techniques. Where the data collection technique is received on a website page with or without including an API (Application Programming Interface), another meaning is the data extraction from a website. Usually, this data is extracted into a new file format. For example, data from a website can be pulled into an excel spreadsheet or images. Usually, the greedy algorithm was conducted to do this crawling. The data samples taken through this technique is the Norwegian Forest Cat. Data crawling is done using the python library called *Icrawler Library*.

The other dataset was obtained from the Oxford-IIIT Pet Dataset [14], which consisted of 25 dog and 12 cat breeds. However, in this study, we are not using all the data. The data samples will only use cat breeds, each class consisting of approximately 200 images of cats. Data will be divided from the total images, consisting of a train folder that places the dataset used for training and the test folder used for validation in the testing process. Each sub-folder in each test and train folder contains folders containing data in the form of cat images according to their respective classes. In this study, the ratio of the data in the train and test folders is 80:20. It means that 80% of the data is for the training process and 20% is for the testing process.

2.3. Data Preprocessing

The collected data is given a correct name according to their respective classes in the labeling process. Then, the preprocessing operation is conducted to equalize the pixel size between one image and another. The goal is to reduce the image value and make it more accessible during the training process because deep learning requires relatively expensive computational costs and takes a lot of time. The resizing function is carried out using the TensorFlow library when training is about to be carried out. Finally, the size of each image is changed to 224x224 pixels, so the computational process is relatively lighter.

2.4. Data Augmentation

Data augmentation is a process to enrich data by performing image processing techniques from an existing dataset. It is because deep learning sometimes requires extensive data to get good accuracy. On the other hand, because the data in the research that will be carried out tend to be small for deep learning, data augmentation is carried out to get more data to achieve the desired accuracy and reduce overfitting problems [15] Figure 2 shows the application of data augmentation. Some of the transformations carried out on the augmentation data for this study are as follows:

a. *Rescale*: This is a way to change the value used to compare the data before another appears. For example, color images or RGB (Red, Green, Blue) have a value range of 0-255, which will impact the current training data. So, to deal with this problem, it can be changed by scaling 1/255 so that the results obtained have a value range of 0-1. This process is also known as data normalization.
b. *Horizontal Flip*: This is a way to flip the image horizontally.

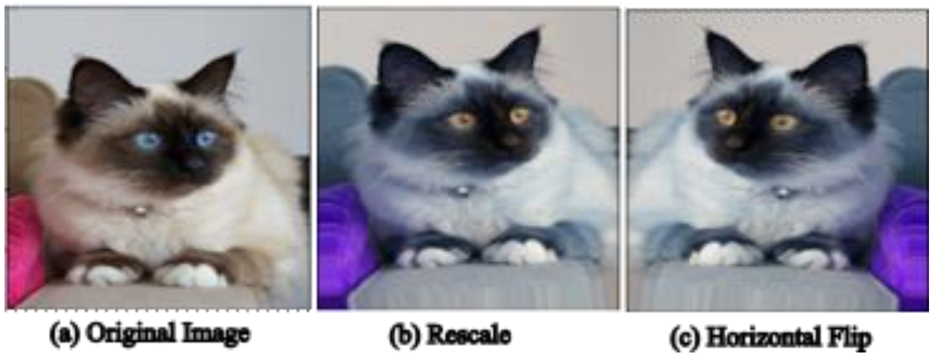

Figure 2. Data Augmentation (a) Original Image (b) Rescaled Image (c) Flipped Image

2.5. CNN Architecture Building

Deep learning techniques are part of machine learning, which was first introduced through a neural network, one of the supervised learning algorithms. One of the most frequently used machine learning algorithms is a neural network. This algorithm created the Neural Network consisting of many neurons that send data signals to each other. Neural network architecture generally consists of 3 layers, the first layer is the input layer, the second layer is the hidden layer, and the last layer is the output layer. The development of neural networks is deep learning.

One deep learning algorithm is CNN (Convolutional Neural Network). It is often used to process, identify, detect, and classify data in images because it can recognize important features without human assistance [16]. Convolutional Neural Network (CNN) is a development of neural networks. The CNN layer consists of a convolutional layer and a fully connected layer. In addition, the convolutional layer consists of several pooling layers that are useful for reducing input spatially. With transfer learning, CNN uses a pre-trained model trained on relatively big data. The model can then identify tasks without retraining from the beginning [17].

To be adapted to new data, several layers have their architecture, and some have their architecture trained by others. Later the base model will be used to classify relatively small data in this research using the most appropriate weight because it has been retrained (on the new layers). Figure 3 depicts our proposed architecture.

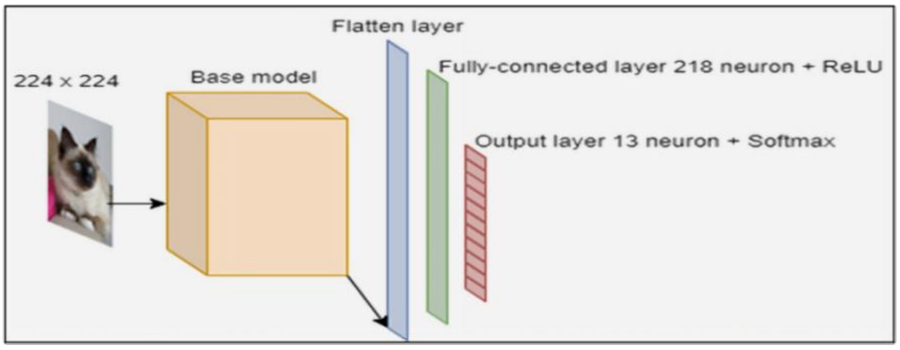
224 x 224
Base model
Flatten layer
Fully-connected layer 218 neuron + ReLU
Output layer 13 neuron + Softmax

Figure 3. Proposed CNN Architecture

In our case, three selective base models will be applied. Those are popular architectures, such as MobileNetV2 [18], VGGNet [9], and InceptionV3 [19]. There is a reason why we choose those models. For example, VGGNet is relatively easy to understand and has been tested on dog and cat classification problems. Subsequently, InceptionV3 has several convolution filters that are used to minimize computational costs, high performance on CNN, the training process is relatively faster than VGG, and the model size is relatively small. Then, MobileNetV3 is relatively lightweight compared to some of the other CNN models because it uses a single convolution channel applied to each color channel, drastically reducing the computational effect and the model size.

In the Fully Connected layer, previously, the data in the form of tensors was converted into vector form flattened. Subsequently, one Fully Connected layer is applied to recognize the implied patterns in each vector received from the previous layer. The number of neurons in that layer is 216. The final output layer consists of 13 neurons. It is because there are 13 cat breeds classified, such as Abyssinian, Bengal, Birman, Bombay, Norwegian Forest Cat, British Shorthair, Egyptian Mau, Maine Coon, Persian, Ragdoll, Russian Blue, Siamese, and Sphynx.

The activation function is used to calculate non-linear data in the above model. The activation functions used in the above architecture are ReLU and softmax, which are non-linear activation functions. In addition, the ReLU activation function [20] is used at each convolution layer. The ReLU activation function has several advantages of not activating all neurons simultaneously to save time during the training process. Eq 1 is ReLU activation function mathematical equation,

$$y = \max(0, x). \tag{1}$$

The mathematical equation of ReLU consists of y as the output value. The max (0, x) is a function to determine if the data x is a number less than 0. The data will be 0, but if the data is more than 0, the value does not change—another activation function used in the final output layer is the softmax activation function. Softmax function, similar to sigmoid activation, which has output value arranged from 1 to 0, is a multi-class version of the sigmoid. Eq 2 is the mathematical equation of the Softmax activation function.

$$s(Xi) = \frac{e^{Xi}}{\sum_{j=1}^{n} e^{Xj}} \tag{2}$$

where e^{Xi} is the exponential value for each input or enumerator divided by $\sum_{j=1}^{n} e^{Xj}$ The total exponential value of each input given will later produce the probability of the distribution.

2.6. Training

It is necessary to have a training process to get high accuracy in classifying cat breeds. In addition, it is required to make an optimal and efficient model so that the objectives and results of this study are as expected. The trained model is expected to recognize the characteristics of each class in the dataset. The convolution layer has a max-pooling layer which aims to reduce the incoming cat image as input. Later, for each 4x4 matrix in the input image, the most prominent element among the four elements in the matrix will be selected so that it becomes a scalar.

The training is carried out so that the data on each layer will run using a feed-forward propagation algorithm. This algorithm flows information from the input layer in the form of an image measuring 128×128 pixels to the output layer. Each time it passes through every single layer, the data sent will pass through an activation function. After the linear calculation process, the results will be calculated in the activation function. In each neuron, there is a weight and a bias value. Both are filled with random values at the beginning of the iteration. After the computed data arrives at the output layer, an error value will be obtained by previously applying the softmax activation function. After the error at the output layer is received, the error, weight, and bias values will be returned to the input layer by passing through the hidden layers. This process is called back-forward propagation.

In our case, the back-propagation process was only conducted on the Fully Connected layer. But, again, it is because we already used the base model, which was previously trained. In this process, the error values that enter each layer will be calculated. Where every epoch or iteration process, which includes feed-forward propagation and back-forward propagation processes, the weight and bias values will be updated using the optimization function. So that the best results will get the lowest or optimal error value, during the back-forward propagation process, the error value carried by the previous layer will be minimized by using the optimization function. The optimization function that will be used in this study is Adam [21]. Adam was chosen because, compared to other optimizations, it tends to be more efficient at computation time and works well on large datasets. Adam combines several other optimizers, such as RMSProp and stochastic gradient descent. Eq 3 is the mathematical equation of the Adam optimization function.

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t \tag{3}$$

In Eq 3, w_t, the weight spread in each back-forward propagation process. Furthermore, variable η is the learning rate or step size in the gradient descent process, while ϵ is epsilon, one of the parameters in the adam optimization function. Lastly, variable m_t, the average of each value used for the gradient moving towards the most optimal point, and v_t, the average of all gradient values squared.

3. RESULTS AND DISCUSSION

3.1. Experimental Scenarios

We evaluate the algorithm with several scenarios depicted in Table 1. The experiment begins by using a self-made model without using a base model. In Scenario-ID 1-10, we use a pure CNN model instead of adding a base model. In contrast, a trained base model was applied on Scenario-ID 11-19. Scenario-ID 1, 2, 3, and 4 using the same model, only a few hyperparameters were changed, such as batch size, learning rate, and the optimizer used. As a result, the model cannot generalize the problem to be classified or overfitting. Especially in Scenario-ID 3, the model is still very underfitting when it was trained using the same epoch as Scenario-ID 1, 2, and 4. In Scenario-ID 5, 6, 7, and 8, the model is created using more convolution filters with the aim that the feature map can read the essential features of an image because the image size is relatively large in deep learning, namely 224x224. Because if you use a filter that has few main elements in the sample image, it will be difficult to read. And it was also using dropout to avoid the possibility of overfitting. Scenario-ID 6 was built using a dropout of 0.1, Scenario-ID 7 was built using 0.2 by reducing the learning rate so that when given a dropout, the model could still learn well, and Scenario-ID 8 was created using a dropout of 0.3. The results of these three Scenarios were also unsuccessful because still overfitting. The dropout size is limited to 0.3 because if it is larger, the number of unreadable neurons during training will also be huge.

Table 1. Experimental Scenarios Setting

ID	*Base Model*	*Filter Numbers*	*FC unit*	*Lr*	*Opt*	*Bs*	*Dt*
1	-	8,16,32,64	218	1e-3	Adam	32	-
2	-	8,16,32,64	218	1e-3	Adam	16	-
3	-	8,16,32,64	218	1e-5	Adam	64	-
4	-	8,16,32,64	218	1e-3	RMSProp	32	-
5	-	32,64,128	218	1e-3	RMSProp	32	-
6	-	32,64,128	218	1e-3	Adam	32	0.1
7	-	32,64,128	218	1e-5	Adam	32	0.2
8	-	32,64,128	218	1e-5	Adam	32	0.3
9	-	32,64,128	512, 128	1e-3	Adam	32	0.1
10	-	32,64,128	512, 128	1e-5	Adam	32	0.3
11	MobilenetV2	-	218	1e-5	Adam	16	-
12	MobilenetV2	-	218	1e-3	Adam	32	-
13	MobilenetV2	-	218	1e-4	Adam	32	0.15
14	VGG16	-	218	1e-4	Adam	32	0.15
15	VGG16	-	218	1e-3	Adam	32	0.25
16	VGG16	-	218	1e-3	Adam	64	0.2
17	InceptionV3	-	218	1e-3	Adam	32	-
18	InceptionV3	-	218	1e-3	Adam	32	0.2
19	InceptionV3	-	218	1e-5	Adam	32	0.2

Notes: FC (Fully Connected), Lr (Learning Rate), Opt (Optimizer), Bs (Batch Size), and Dt (Dropout Rate)

Another scenario is Scenario-ID 9 and 10. It is made by making the model deeper, such as adding a fully connected layer in the hope that the model can better recognize patterns in images and combine with dropouts to minimize the risk of overfitting and reduce the learning rate to minimize risk. The loss function is stuck at the saddle point when optimizing parameters using the adam optimizer. In the ID 9 scenario, it is made with 512 neurons in the first layer, followed by 128 neurons in the next layer before the output layer is in the Fully Connected Layer, and uses a dropout of 0.1 with a learning rate of 1e-3. In the ID 10 scenario, it is made with 512 neurons in the first layer, followed by 128 neurons in the next layer before the output layer is in the Fully Connected Layer, and uses a dropout of 0.3 with a learning rate of 1e-5. The results of the two scenarios show only a slight increase in accuracy but are still very overfit.

In Scenario-ID 11, 12, and 13, use the MobilenetV2 base model with a freeze model. Freeze model means using parameters on the convolution layer trained on Imagenet so that there is no gradient calculation on the convolution layer. The model can generalize problems relatively better than models built from scratch. These three scenarios change the output layer from 1000 classes to 13 classes. Before the output layer, it is given a flattened after layer with 218 neurons. When compared to the scenario that has been done before, the use of MobilenetV2 is very effective in classifying cat images with only a small dataset.

Instead of using only the mobilenetV2 base model, this study also uses VGG16 for Classification. VGG16 was tested on Scenarios-ID 14, 15, and 16. However, the VGG model is known for its weight and long training time compared to MobilenetV2. Therefore, the results obtained in these three scenarios are no better than the MobilenetV2 that has been tested.

Scenario-ID 17, 18, and 19 were created using InceptionV3. That base model is smaller than the VGG model and still more prominent than MobilenetV2. It is interesting to test in the case of Classification with many classes and a small dataset. Unfortunately, the results in the scenarios with this model are not optimal in solving the problems encountered in this study.

3.2. Evaluation and Deployment

The results of all experiments carried out in the scenario table show that Scenario-ID 12 is the best result because it has the highest accuracy and the lowest loss of all the scenarios that have been carried out. The error function results are 0.1171 in training and 0.6417 in testing, respectively. At the same time, the accuracy result was 0.9203 in the training dataset and 0.8199 in the testing dataset, respectively. Therefore, the model in Scenario-ID 12 will also be used for deployment on Android, which has the best result. The overall accuracy result is described in Table 2.

Table 2. Overall Accuracy Result

	Scenario-ID																		
	1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***16***	***17***	***18***	***19***
Acc. (%)	33	34	38	35	40	40	34	38	35	34	78	82	80	61	62	62	80	66	79

The model seems stable here in Scenario-ID 12 epochs only up to 15. Because in the 50th epoch, the accuracy of the testing data dropped slightly, until 0.8180, with an error function of 0.9213. Meanwhile, if the epoch is continued until the 100th, the model will also be increasingly unstable. It is proven that the accuracy of the testing dataset decreases to 0.7941, while the error function also increases drastically by 7.7541. Therefore, when tuning the hyperparameter, it was decided to use a model that uses epoch 15 because the model is more stable and the error function converges well. Figure 4 shows the training process of the best scenario (Scenario ID 12) on the following accuracy and loss. Figure 4 looks overfitting and almost underfitting because it has only a few image samples on our dataset, and some cat breeds are similar.

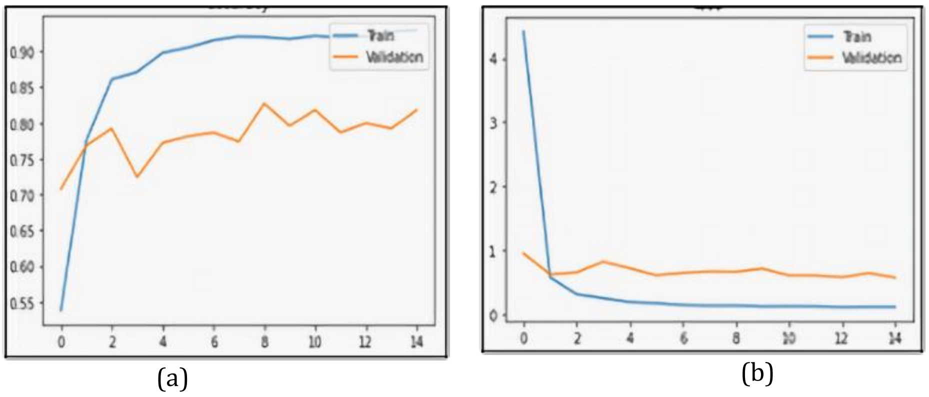

Figure 4. (a) Training accuracy (b) Training loss

After the best model is selected, the first step in the android deployment is to add Tensorflow Lite to the project to be created, starting with adding the TensorFlow lite dependency to build the Gradle file of the android project. Creating an Android-based classifier system takes several stages, starting from making the UI and proceeding with the preprocessing of the image before the image is classified and then passing it on to the Tensorflow lite classifier downloaded. Finally, the application runs with the user being able to press which image to recognize. If the user touches it, a pop-up text will appear on the bottom side of the cellphone. Some crucial steps during the inference process include initializing the interpreter with the model to be used. Figure 5 shows the android application during recognizing cat breed images. After that, we call the application Catbreednet.

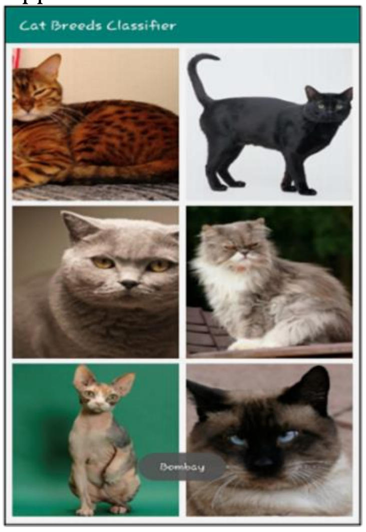

Figure 5. Screenshot of Catbreednet Application

4. CONCLUSION

The Catbreednet android application has been created to recognize the cat breed from a given image. It is classified using the Convolutional Neural Network algorithm, especially the transfer learning technique with the base model. We conduct several scenarios to find the best model result. The three base models used in our CNN model are MobileNetV2, VGGNet, and InceptionV3. We added one more fully connected layer instead of only using the base model in our architecture. After evaluating every scenario, the best base model for categorizing the cat breeds is MobilenetV2, which is in the ID-12 scenario with an accuracy of 82%. We also deploy the model to the Android application. As a result, the system can recognize several cat breeds with unique characteristics. However, the best accuracy results are not yet high because the number of datasets is small, and several types of cats have very similar features that are difficult to distinguish. Therefore, in future work, it is better to increase the number of cats breed data samples so that it is expected to reduce overfitting and increase accuracy. Later, changing the neural network architecture to the newest fashion will likely get a better model.

ACKNOWLEDGEMENTS

This research was supported by IntSys Lab (Intelligent System Research Laboratory) Universitas Dian Nuswantoro Semarang. We greatly appreciate the insight and support for the continuation of this study.

REFERENCES

[1] V. J. Crossley, A. Debnath, Y. M. Chang, R. C. Fowkes, J. Elliott, and H. M. Syme, "Breed, Coat Color, and Hair Length as Risk Factors for Hyperthyroidism in Cats," J Vet Intern Med, vol. 31, no. 4, pp. 1028–1034, Jul. 2017, doi: 10.1111/jvim.14737.

[2] D. van Lent, J. C. M. Vernooij, and R. J. Corbee, "Kittens That Nurse 7 Weeks or Longer Are Less Likely to Become Overweight Adult Cats," Animals, vol. 11, no. 12, p. 3434, Dec. 2021, doi: 10.3390/ani11123434.

[3] F. Alharbi, A. Alharbi, and E. Kamioka, "Animal species classification using machine learning techniques," MATEC Web of Conferences, vol. 277, p. 02033, 2019, doi: 10.1051/matecconf/201927702033.

[4] A. Vecvanags et al., "Ungulate Detection and Species Classification from Camera Trap Images Using RetinaNet and Faster R-CNN," Entropy, vol. 24, no. 3, p. 353, Feb. 2022, doi: 10.3390/e24030353.

[5] T. Karlita, N. A. Choirunisa, R. Asmara, and F. Setyorini, "Cat Breeds Classification Using Compound Model Scaling Convolutional Neural Networks," in International Conference on Applied Science and Technology on Social Science, 2022.

[6] Y. Lee, "Image Classification with Artificial Intelligence: Cats vs Dogs," in 2021 2nd International Conference on Computing and Data Science (CDS), IEEE, Jan. 2021, pp. 437–441. doi: 10.1109/CDS52072.2021.00081.

[7] P. Borwarnginn, K. Thongkanchorn, S. Kanchanapreechakorn, and W. Kusakunniran, "Breakthrough Conventional Based Approach for Dog Breed Classification Using CNN with Transfer Learning," in The 11th International Conference on Information Technology and Electrical Engineering, 2019.

[8] S. Varghese and S. Remya, "Dog Breed Classification Using CNN," in Security Issues and Privacy Concerns in Industry 4.0 Applications, Wiley, 2021, pp. 195–205. doi: 10.1002/9781119776529.ch10.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.1556

[10] S. Divya Meena and L. Agilandeeswari, "An Efficient Framework for Animal Breeds Classification Using Semi-Supervised Learning and Multi-Part Convolutional Neural Network (MP-CNN)," IEEE Access, vol. 7, pp. 151783–151802, 2019, doi: 10.1109/ACCESS.2019.2947717.

[11] Y. Zhang, J. Gao, and H. Zhou, "Breeds Classification with Deep Convolutional Neural Network," in ACM International Conference Proceeding Series, Association for Computing Machinery, Feb. 2020, pp. 145–151. doi: 10.1145/3383972.3383975.

[12] W. Raccagni and S. Ntalampiras, "Acoustic Classification of Cat Breed Based on Time and Frequency Domain Features," in 2021 30th Conference of Open Innovations Association FRUCT, IEEE, Oct. 2021, pp. 184–189. doi: 10.23919/FRUCT53335.2021.9599975.

[13] N. Qatrunnada, M. Fachrurrozi, and A. S. Utami, "Cat Breeds Classification using Convolutional Neural Network for Multi-Object Image," Sriwijaya Journal of Informatic and Applications, vol. 3, no. 2, pp. 26–35, 2022, [Online]. Available: http://sjia.ejournal.unsri.ac.id

[14] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. v Jawahar, "Cats and Dogs," in IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[15] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," J Big Data, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0197-0.

[16] L. Alzubaidi et al., "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," J Big Data, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[17] F. Zhuang et al., "A Comprehensive Survey on Transfer Learning," Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.02685

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Jan.

2018, [Online]. Available: http://arxiv.org/abs/1801.04381

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.00567

[20] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in International Conference on Machine Learning, 2010. doi: 10.5555/3104322.3104425.

[21] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980