UNIVERSITÀ DEGLI STUDI DI MILANO

Draft version December 24, 2022
Typeset using LaTeX default style in AASTeX631

cats and dog (Project 1)*

ELAHEH ESFANDI [1]

[1] *Master Student of Milan University, data science in economics (DSE), Matriculation number: 963762, elaheh.esfandi@studenti.unimi.it.*

ABSTRACT

in this paper, I Used Tensorflow 2 to train a neural network for the binary classification of cats and dogs based on images from the given data set. Images transformed from JPG to RGB (or grayscale) pixel values and scaled down. Experiment with different network architectures and training parameters documenting their influence on the final predictive performance. in the end, 5-fold cross-validation was used to compute the estimated risk. the estimates computed according to the zero-one loss.

Keywords: binary classification, pictures, Tensorflow 2, neural network.

1. INTRODUCTION

Deep learning algorithms were inspired by how the human brain works, using an enormous number of neurons linked by a massive number of connections to execute complex activities including speaking, moving, thinking, and seeing. Most deep learning architectures are artificial neural networks composed of multiple layers, thus being called "deep", and a basic element called neuron.Goodfellow (2016) VGG stands for Visual Geometry Group and refers to the deep conventional network (ConvNet) models either with 16 layers (VGG-16) or 19 layers (VGG-19). These models were shown to provide excellent accuracy on ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) classification and localization tasks. The network consists of a stack of convectional layers followed by three fully connected (FC) layers with a final 1000-way softmax.

In this paper, I want to train a neural network for the binary image classification of cats and dogs. for this matter and According to the introduced database features, I will use the Visual Geometry Group (VGG) model which will use the Kernel Perception classification. The novelty of this study lies in the following endeavors. Firstly, by using the method that I will introduce in the 3rd section, I am trying to classify the Kernel Perception according to the cats and dogs pic "Data set". Secondly, I would test the accuracy of the model that we use in order to find the best method to fit this data set. The rest of the paper is organized as follows. The next section deals with the Data description and the methodology.in section 2, I will provide you with the data that I will use during the modeling. while, Section 3 will provides a background of the user model, the general framework, and the Results and findings of the model. Finally, Section 4 will go through the most important key finding, discussion, and conclusion.

* "Statistical Methods for Machine Learning - Projects valid through May 2023", 2022-2023, Released on November, 16th, 2022

2. RESEARCH DATA

We will use the cats and dogs data set [1], which is published by the professor, we will go through "the overview of the chosen data set" and "how data have been organized for experiments". This data set has 25000 images of cats and dogs. The images are 16*16 colorful pixels.

The Asirra (animal species image recognition for restricting access) data set was introduced in 2013 for a machine learning competition. The data set includes 25,000 images with equal numbers of labels for cats and dogs.

with the python coding, I tried to link the Kaggle database to the googlecolab [2], so I could run the codes whenever I want without the need of the changing in the repository and observe any different.

we can use many ways for adding the data set into python, based on the huge memory needed for our data set and the time-consuming of loading the data set of the data set, I decided to upload the data set into my googlecolab with the help of "!wget" code which will connect to download the data set from the web page that professor put the data set. after entering the data set from the website, I unzipped the file containing all the images in two different folders of "cats" and "dogs".

3. THE ANALYSIS WITH WISE COMMENTARY AND THEORETICAL BACKGROUND OF EACH USING MODEL

In this section, we will go through the applied pre-processing techniques, the metrics used for evaluating performances, and the experimental methodology like VGG and its implementations. then experimental results as plots and/or tables will be present.

3.1. *Data preparation*

The data were unconstrained for the writer, style, and method of preparation. These characteristics help overcome the limitations of earlier databases that contained only isolated characters or were prepared in a laboratory setting under prescribed circumstances. the first step to start analyzing any data set is to do the pre-processing of data.

for this data set, based on the models that we will use in the next sections, we need to find and Remove any images with a size of 0. for this reason we started with Defining the paths, the glob module is used to retrieve files(path names) matching a specified pattern.to retrieve paths recursively from inside the directories(files) and sub-directories(sub-files). then with the for and if loop, 1588 images were deleted.

for the classification matter, we labeled our pics data set into "cats" or "dogs" based on their file(folder) name. then we concatenated it all to gather. the summary of this database is available in a table(1).

in the case of classification problems, frequently used metrics consist of, Accuracy, Log-loss, or F-measure. Based on The metrics from "scikit learn" (our goal model library), If we have a sample data set and want to train a model to predict it, we can use one of these metrics to evaluate how efficient the model is. so in this sample data set, we only evaluated the model once. Assuming we correctly separated the data set into a training set and a test set and fitted the model with the training set while evaluated with the test set, we obtained only a single sample point of evaluation with one test set. How can we be sure it is an accurate evaluation, rather than a value too low or too high by chance? or on comparing two models based on the same sample database?

all in all, The reason we are concerned about this is to avoid surprisingly low accuracy when the model is deployed and used on entirely new data than the one we obtained, in the future. so based on this short view, we split it into a training validation set and test set; also Split the training validation set into a training and validation set.

Images must be transformed from JPG to RGB (or grayscale) pixel values and scaled down. Experiment with different network architectures and training parameters documenting their influence on the final predictive performance. Rescale is a value by which we will multiply the data before any other processing. Our original images consist of RGB coefficients in the 0-255, but such values would be too high for our model to process (given a typical learning rate), so we target values between 0 and 1 instead by scaling with a 1/255.

The question raised is why the re scale is 1/255 and why we need this before training the neural networks. From the above 8 bit grayscale image, every digital image is formed by a pixel having a value in the range 0 to 255. 0 is black and 255 is white. or colorful images, it contains three maps: Red, Green, and Blue, and all the pixels are still in the range of 0 to 255(pixel value ranges according to the storage size, the pixel range is 0 2 bits).

[1] https://unimibox.unimi.it/index.php/s/eNGYGSYmqynNMqF/download

[2] https://colab.research.google.com/drive/1Ant9VA_3xIG9zWUsudieQqxE-dSqa2kP?usp=sharingthis link is the link of the goolecolab of this paper.

Table 1. data description.

	file name	label
count	23410	23410
unique	23410	2
top	/content/CatsDogs/Cats/12186.jpg	cat
freq	1	11741

Source: the outcome of this paper

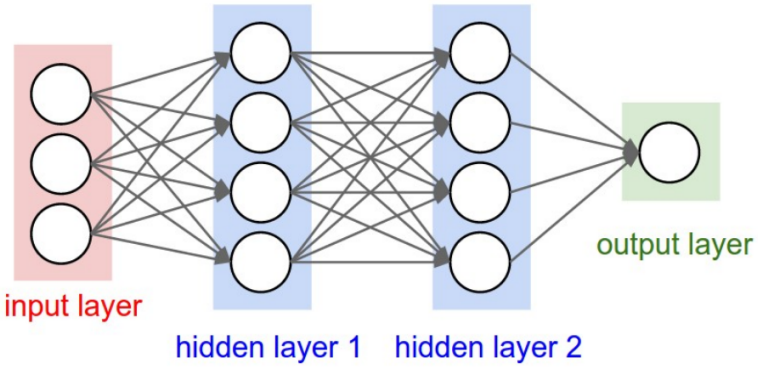

Figure 1. Keras Layers. Source:Simonyan & Zisserman (2014)

Since 255 is the maximum pixel value. Rescale 1/255 is to transform every pixel value from the range [0,255] to [0,1]. the benefits are Treating all images in the same manner and Using a typical learning rate.

some images are a high pixel range, some are a low pixel range. The images are all sharing the same model, weights, and learning rate. so in this case when we reference the learning rate from others' work, we can directly refer to their learning rate if both works do the scaling prepossessing over the images data set. Otherwise, a higher pixel range image results in higher loss and should use a smaller learning rate, a lower pixel range image will need a larger learning rate.

Referencing from the image Keras "ImageDatagenerator" source code, the parameter re-scale is to multiply every pixel in the prepossessing image. after we introduced our "image data generator" model, we add it to all our samples by Flowing From Data-Frame. now our samples are ready to be analyzed by the VGG model which will be introduced in the next part.

3.2. *image recognition with VGG model*

performance of deep learning over traditional machine learning approaches enables new applications in image recognition, computer vision, speech recognition, machine translation, medical imaging, robotics, and many more. "VGG" stands for Visual Geometry Group; also known as "VGGNet", it is a standard deep Convectional Neural Network (CNN) architecture with multiple layers. and was developed to increase the depth of such "CNNs" in order to increase the model performance. The "deep" refers to the number of layers with VGG-3 or VGG-4 consisting of 3 and 4 convectional layers. The "VGG" architecture is the basis of ground-breaking object recognition models and the most popular image recognition architectures.Simonyan & Zisserman (2014).

the VGG model consists of 4 different parts:

1. input: VGG takes in a 180*180 pixel RGB image. For the ImageNet competition, the authors cropped out the center 224x224 patch in each image to keep the input image size consistent.

2. Convolutional Layers: The convectional layers in VGG use a very small receptive field (3x3, the smallest possible size that still captures left/right and up/down). There are also 1x1 convolution filters which act as a linear transfor-

Table 2. VGG model details

Layer (type)	Output Shape	Param
$\text{input}_1(InputLayer)$	[(None, 180, 180, 3)]	0
conv2d (Conv2D)	(None, 180, 180, 32)	896
$\max_p ooling2d(MaxPooling2D)$	(None, 90, 90, 32)	0
$\text{conv2d}_1(Conv2D)$	(None, 90, 90, 64)	18496
$\max_p ooling2d_1(MaxPooling2D)$	(None, 45, 45, 64)	0
flatten (Flatten)	(None, 129600)	0
dense (Dense)	(None, 128)	16588928
$\text{dense}_1(Dense)$	(None, 1)	129

Total params: 16,608,449
Trainable params: 16,608,449
Non-trainable params: 0

Source: the outcome of this paper

mation of the input, which is followed by a ReLU unit. The convolution stride is fixed to 1 pixel so that the spatial resolution is preserved after convolution.

3. Fully-Connected Layers: VGG has three fully-connected layers: the first two have 4096 channels each and the third has 1000 channels, 1 for each class.

4. Hidden Layers: All of VGG's hidden layers use ReLU (a huge innovation from AlexNet that cut training time). VGG does not generally use Local Response Normalization (LRN), as LRN increases memory consumption and training time with no particular increase in accuracy.

this model can be loaded and used in the "Keras" deep learning library. "Keras" provides an Application interface for loading and using pretrained weights provided by the Oxford group and using it as a starting point in our classifying images.

as I mentioned before, in order to write the VGG model we have to write 4 different layers. an input layer, the first block with 32 filters, The second block with 64 filters, Flatten Layer, and the Fully Connected Layers.

in a Keras layer, shapes are tuples representing how many elements an array or tensor has in each dimension. in our model, A tensor with shape (180, 180, 3) is 180 dimensional with the first dimension having 180 elements. Each of these 180 elements has 180 elements, and each of these 180 elements has 3 elements. Thus a total of 180*180*3 = 97200 elements.

in each of the first and second block filters, we have two different layers; the 2D convolution layer (e.g. spatial convolution over images) and the MaxPooling2D layer. The 2D convolution layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

while the Max pooling operation for the 2D layer will Downsample the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool size) for each channel of the input. The window is shifted by strides along each dimension. the difference between these two layers is the difference between filters. in keras, Filters represent the number of filters that should be learned by the convolutional layer. each filter slides over the input image, generating a "feature map" as output. In image processing kernel is a convolution matrix or masks which can be used for blurring, sharpening, embossing, edge detection, and more by doing a convolution between a kernel and an image.

so after writing the model, Before we can start with training, we have to compile the model using the compile method. Compiling configures the model for training. I choose the "Adam" optimizer as this is an excellent default to start with. The loss function is "binary cross entropy" because we are training a classification model with two possible outputs. I want to report the accuracy during training; therefore, I added accuracy as a metric. After I compiled the model, I started training the model by calling the fit method on the model instance. the parameters are as in table(2).

The first parameter is the iterator that we created by calling the flow from the directory of the Image Data Generator for the training data. The second parameter is the iterator from the validation data. The epoch parameter shows how

Figure 2. Measures the Training vices Test Accuracy. Source: the outcome of this paper

Figure 3. Measures the Training vices Test Loss. Source: the outcome of this paper

many times the model will process the entire training set. Here, I want to process all 20.000 training images 50 times. The steps per epoch show how many batches it should process before it finishes the epoch. I established a batch size of 200 previously on the flow from the directory, which gives 200 * 50 = 20,000 (the number of images of our training set). The same goes for the validation steps, 50 * 100 = 5,000, the number of images in our validation set.

3.3. *data analysis result with VGG model*

The task to be tackled is to identify the correct label given an input image using a convolutional neural network architecture for digit image classification, as depicted in Fig. 2. It comprises an input layer that receives samples of digit images, followed by two blocks of convolutional layers with the rectified linear unit as the activation function and max pooling layers.

The compact image feature maps are then flattened to derive a feature array and finally used for classification through a fully connected or dense layer with softmax activation function, which represents the probability distribution over n different classes (Goodfellow et al., 2016).

Table 3. Fitting the VGG model

Epoch	4s step	loss	accuracy	val$_l oss$	val$_a ccuracy$
1/5	1033s	0.7288	0.6333	0.5755	0.6965
2/5	1026s	0.5265	0.7349	0.5184	0.7506
3/5	1028s	0.4460	0.7883	0.5089	0.7545
4/5	1039s	0.3616	0.8409	0.5196	0.7628
5/5	1020s	0.2511	0.8951	0.5851	0.7620

Source: the outcome of this paper

Table 4. summary of search space

name	default	min$_v alue$	max$_v alue$	step
data$_a ugmentation(Float)$	0.01	0.01	0.1	0.01
dropout$_h idden_l ayer(Float)$	0.05	0.05	0.1	0.01
dropout$_f latten_l ayer(Float)$	0.2	0.2	0.5	0.1
convolution$_{1\,f} ilters(Int)$	None	16	64	16
convolution$_{2\,f} ilters(Int)$	None	64	128	32
convolution$_{3\,f} ilters(Int)$	None	128	256	64
convolution$_{4\,f} ilters(Int)$	None	256	512	128
num$_u nits(Int)$	None	64	256	64
num$_u nits_s econd_l ayer(Int)$	None	64	256	64

'sampling': None
Default search space size: 9

Source: the outcome of this paper

Figures (2) and (3) are also shown Clearly the training vice test accuracy and test loss. my model validates with a Test Loss of 0.5849 and a Test Accuracy of 0.7618.

3.4. *Hyperparameters model*

The parameters of the model are the parameters calculated on the given dataset by the model. while Hyperparameters of Models are the parameters that the data model cannot predict. This is used for calculating the parameters of the model, and in deep neural networks is the learning rate. These variables remain constant over the training process and directly impact the performance of your ML program. The process of selecting the right set of hyperparameters for your machine learning (ML) application is called hyperparameter tuning or hyper tuning. The Keras Tuner is a library that helps to pick the optimal set of hyperparameters for the TensorFlow program.

we have 2 types of Hyperparameters; Model hyperparameters and Algorithm hyperparameters. Model hyperparameters influence model selection such as the number and width of hidden layers while Algorithm hyperparameters influence the speed and quality of the learning algorithm such as the learning rate for Stochastic Gradient Descent (SGD) and the number of nearest neighbors for a k Nearest Neighbors (KNN) classifier. for the aim of this paper, I continued with adding Hyperparameters to my model.

The tuning technique is used to estimate the best hyperparameter combination that helps the algorithm to optimize the efficiency of the model. Keras Tuner makes it easy to define a search space and leverage included algorithms to find the best hyperparameter values. Keras Tuner comes with Bayesian Optimization, Hyperband, and Random Search algorithms built-in, and is also designed to be easy for researchers to extend in order to experiment with new search algorithms. According to the Methods of Hyper-Parameter Tuning, bayesian Optimization is The optimization challenge to tune and locate the correct hyperparameters for your model. Adjusting model parameters, to minimize the loss function of our model. In a minimal number of moves, Bayesian optimism helps one find the minimum point.

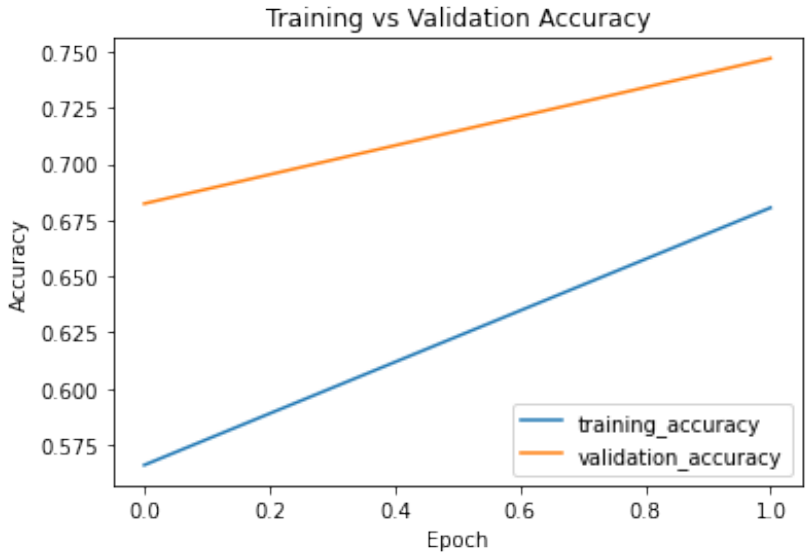

Figure 4. Measures the training vice Validation Accuracy . Source: the outcome of this paper

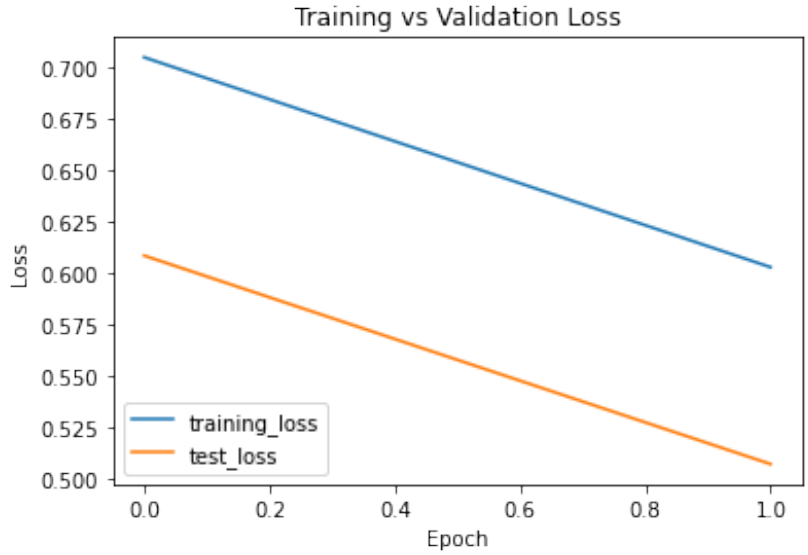

Figure 5. Measures the training vice Validation loss. Source: the outcome of this paper

Bayesian optimization also uses a buying feature that guides the sampling in areas where the best observation is likely to improve on the present one.O'Malley et al. (2019)

I used the "Bayesian Optimization tuning with Gaussian process". I have chosen the Arguments as fallow: first is the Instance of HyperModel class that takes hyperparameters and returns a Model instance. then the objective is in a string or a list of "keras tuner" format. If it is in a string format, the direction of the optimization (min or max) will be inferred. If it is in a list format of "keras tuner", will minimize the sum of all the objectives to minimize subtracting the sum of all the objectives to maximize. this argument will return a single float as the objective to minimize. the third one is "max trials" which is in Integer format and shows the total number of trials (model configurations) to test at most. I put 3 just based on the time consumption for my home computer. all in all, in this part of my modeling, I wrote a function that creates and returns a Keras model. define the hyperparameters during model creation. initialize a tuner with Bayesian Optimization and I used objective to specify the objective to select the best models, and 3 for the number of "trials" to specify the number of different models to try. then, with the use of the search function on

Table 5. Fitting the hyperparameters-VGG model

topic	value
$\text{data}_{a}ugmentation$	0.05
$\text{dropout}_{h}idden_{l}ayer$	0.05
$\text{dropout}_{f}latten_{l}ayer$	0.30000000000000004
$\text{convolution}_{1\,f}ilters$	48
$\text{convolution}_{2\,f}ilters$	64
$\text{convolution}_{3\,f}ilters$	192
$\text{convolution}_{4\,f}ilters$	384
$\text{num}_{u}nits$	256
$\text{num}_{u}nits_{s}econd_{l}ayer$	192

Source: the outcome of this paper

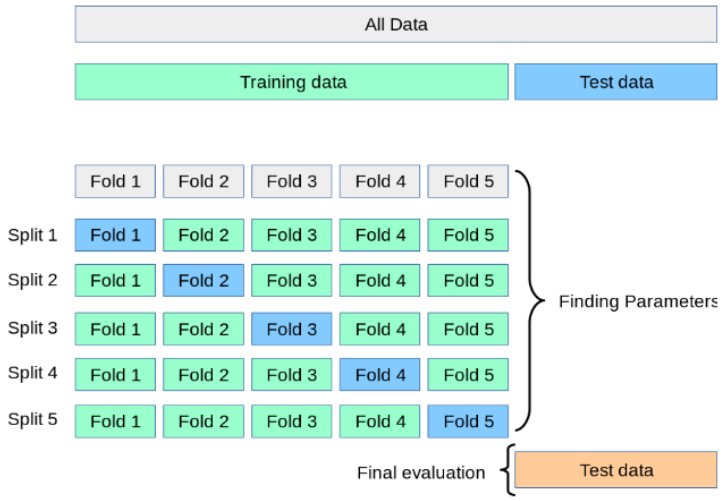

Figure 6. 5 folds cross validation

Source:Res (2010)

the "train generator" data set, I tried to get the best model and validate it with the "valid generator" data set. the summary of search space can be seen in table(4).

after running the terrestrial, the optimal hyperparameter was the third one with the Best accuracy value of 0.6882. another features of this trial has been shown in table(5).

I get the value of the optimal hyperparameter as you can see in table(6).

Figures (4) and (5) also show Clearly the training vice Validation Accuracy and Validation loss. my model validates with a Test Loss of 0.5068 and a Test Accuracy of 0.7472.

3.5. *Cross-Validation*

Cross-validation is a statistical method used to estimate the skill of machine learning models. while in the k-Fold Cross-Validation procedure, k refers to the number of groups that a given data sample is to be split into. The performance measure reported by k-fold cross-validation is the average of the values computed in the loop.

This approach can be computationally expensive but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage in problems such as inverse inference where the number of samples is very small. the 5-fold cross-validation technique can be seen in figure (6).

in this part, I will continue with Using 5-fold cross-validation to compute the risk estimates. While the training loss can be chosen freely, the reported cross-validated estimates will be computed according to the zero-one loss.

Table 6. Fitting the hyperparameters-VGG model("model-2")

Layer (type)	Output Shape	Param
$\text{input}_3(InputLayer)$	(None, 180, 180, 3)]	0
$\text{random}_flip_2(RandomFlip)$	(None, 180, 180, 3)	0
$\text{random}_rotation_2(RandomRotation)$	(None, 180, 180, 3)	0
$\text{random}_zoom_2(RandomZoom)$	(None, 180, 180, 3)	0
$\text{conv2d}_8(Conv2D)$	(None, 180, 180, 48)	1344
$\text{max}_pooling2d_8(MaxPooling2D)$	(None, 90, 90, 48)	0
$\text{dropout}_14(Dropout)$	(None, 90, 90, 48)	0
$\text{conv2d}_9(Conv2D)$	(None, 90, 90, 64)	27712
$\text{max}_pooling2d_9(MaxPooling2D)$	(None, 45, 45, 64)	0
$\text{dropout}_15(Dropout)$	(None, 45, 45, 64)	0
$\text{conv2d}_10(Conv2D)$	(None, 45, 45, 192)	110784
$\text{max}_pooling2d_10(MaxPooling2D)$	(None, 22, 22, 192)	0
$\text{dropout}_16(Dropout)$	(None, 22, 22, 192)	0
$\text{conv2d}_11(Conv2D)$	(None, 22, 22, 384)	663936
$\text{max}_pooling2d_11(MaxPooling2D)$	(None, 11, 11, 384)	0
$\text{dropout}_17(Dropout)$	(None, 11, 11, 384)	0
$\text{flatten}_2(Flatten)$	(None, 46464)	0
$\text{dropout}_18(Dropout)$	(None, 46464)	0
$\text{dense}_6(Dense)$	(None, 256)	11895040
$\text{dropout}_19(Dropout)$	(None, 256)	0
$\text{dense}_7(Dense)$	(None, 192)	49344
$\text{dropout}_20(Dropout)$	(None, 192)	0
$\text{dense}_8(Dense)$	(None, 1)	193

Total params: 12,748,353
Trainable params: 12,748,353
Non-trainable params: 0

Source: the outcome of this paper

The general procedure of 5-fold cross-validation is as follows: 1. Shuffle the data set randomly. 2. Split the data set into 5 groups. 3. For each unique group: Take the group as a holdout or test data set, Take the remaining groups as a training data set, Fit a model on the training set and evaluate it on the test set and Retain the evaluation score and discard the model. 4. Summarize the skill of the model using the sample of model evaluation scores.

I used the "cross-validate" of the "sklearn" package to define my cross-validation model. all n all, i slightly adapt the model in order to add K-fold Cross Validation by strip off some code for the model that we no longer need, firstly. then,longer generate the visualizations, and remove the part generating them.

as you could see in my codes, i generate the data set in "train data generator","valid data generator" by using the train and validation past of the data base that i separated already in the first part of the analysis, and also "test data generator" by using the test part of the separated one. the outcome of the model can be seen in table (7).

4. CONCLUDING REMARKS: COMMENTS AND DISCUSSION ON THE EXPERIMENTAL RESULTS

we have been asked to classified the group of picture belong to the cats and dogs. for the reason of not using libraries directly i choose to use VGG modeling for For my image classification. VGG(Visual Geometry Group)is a standard deep Convolutions Neural Network(CNN) architecture with multiple layers.

There are many instances in which a first pass at model training is insufficient such that the accuracy needs to be increased. as we surely know, Model accuracy is defined as the number of classifications a model correctly predicts divided by the total number of predictions made. It's a way of assessing the performance of a model, but certainly

Table 7. 5 folds cross validation

Fold	Epoch	2s/step	loss	accuracy	val$_loss$	val$_accuracy$
f1	1/2	565s	0.8669	0.6323	0.5666	0.07004
f1	2/2	561s	0.0.5356	0.7425	0.4797	0.0.7780
f2	1/2	558s	0.9915	0.6058	0.5707	0.6800
f2	2/2	557s	0.0.5655	0.7173	0.5160	0.7441
f3	1/2	558s	0.9545	0.6288	0.5605	0.7260
f3	2/2	555s	0.5676	0.7169	0.4960	0.7584
f4	1/2	554s	0.8213	0.6283	0.5584	0.7193
f4	2/2	556s	0.5483	0.7313	0.5224	0.7352
f5	1/2	566s	0.9221	0.6306	0.5538	0.7155
f5	2/2	561s	0.5326	0.7437	0.5150	0.7465

Cross validation split n(1,2,3,4,5)
length of training set :14982
length of validation set : 3746
length of test set : 4682

Source: the outcome of this paper

Table 8. test of VGG with 5 folds cross validation

Fold	loss	accuracy
f1	0.4802	0.7713
f2	0.5247	0.7413
f3	0.5029	0.7441
f4	0.5217	0.7529
f5	0.5127	0.7454

Source: the outcome of this paper

not the only way. In fact, accuracy provides the best perspective on how well a model is performing on a given data set(our test data set).

for checking the performance of the model, i used the 5 fold cross validation. the outcome of the testing VGG model based on the 5 fold cross validation can be seen in table (8). as we could see the the 4th fold with the loos of 0.52 and accuracy of 75 percent is the best accuracy that we get from testing the VGG model.

the most challenging and the most probably weak issue of my work is that i used the least number of "epoc". by using a personal computer and the weight of the image data, it took me so long hours to do the 50 or 100 epoc. so i decided to reduce it in order to make my way at least a little faster and easier.

APPENDIX

A. DECLARATION BY AUTHOR

"I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offenses in the university, and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study."

B. ALL THE AVAILABLE LINKS FOR CODES BY AUTHOR

GitHub:https: https://github.com/elahehesfandi/-neural-network-for-the-binary-classification
or this is the repository name: elahehesfandi/-neural-network-for-the-binary-classification
Google-colab: https://colab.research.google.com/drive/1Ant9VA_3xIG9zWUsudieQqxE-dSqa2kP?usp=sharing

REFERENCES

Goodfellow, I., e. a. 2016, MIT Press, Cambridge, MA. http://www.deeplearningbook.org

O'Malley, T., Bursztein, E., Long, J., et al. 2019. https://github.com/keras-team/keras-tuner

Res, J. M. L. 2010. https://scikitlearn.org/stable/modules/cross_validation.html

Simonyan, K., & Zisserman, A. 2014, Global Survey