The Combination between Two DataSets and Two Models

Po-Chun Lu

Abstract—In this paper, we built a Convolution neural network (CNN) model based on existing models, which was presented by others, and trained this model to classify the data set, which was a combination of two other existing data sets. To understand the feasibility of module inheritance with high similarity problem. Moreover, to understand the influences from two data set collected and created by different agent. We picked up two different CNN models on Kaggle website along with their original problem data set. We found that it will not be too difficult to build a new model from the existing model, The most important thing is to call the parameters correctly and select the most suitable function. Moreover, the understanding about the data set is important as well. Due to the different sources of the dataset, which might contain different attributes, we have to be careful about the number of the samples in each class and the size of each sample. Result showed that our proposed had the ability to classify the data set with the final, average 0.91 F1-score of three classes, and test set accuracy of 94% without overfitting or underfitting problem.

Index Terms—fake face, Image classification, machine learning, neural network.

1 Introduction

In recent years, Machine learning has become a hot topic. It is now wildly used in many fields, from daily used recommendation system to the detector for space exploration. There are four different types of nowadays machine learning, which are Supervised learning, Unsupervised learning, Semi-supervised learning, and the reinforcement learning. The difference between them is the amount of label data in the whole data set. 100% label data is called Supervised learning; 0% of label data is called Unsupervised learning. Semi-Supervised learning is half and half, and the reinforcement learning has less label data.

In the machine learning, the most important element is the data. As the discussion above, we using data data set attribute to distinguish the method. Normally, we do not just feed the row data to the learning model. We have to do some pre-processing, such as parsing the network flow to get the most important feature, or getting rid of the noisy image in the image data. Although it is impossible for us to remove all noise and obtain perfect data, we still have to do our best, because data plays an important role in machine learning.

After pre-processing, we have to separate data set into Training set, Validation set and Testing set. The reason for data splitting is to give each part different purpose. In general, 60% of data for Training, 20% of data for Validation, and 20% of data for Testing, it might be various depend on different problem. The purpose of Training set is to minimize the loss, validation set is for choosing best hyper-parameters, and Testing data is to evaluation the model.

The reason made the machine learning famous could be various. It might because of the trend of the technique or the lower price of high performance system, most people can enter this field more easily than before. However, with the lower restrictions, some issues have come along with this trend. The reality of information has received more attention. Recently, not only fake text, but even biological information containing personal information (such as human faces) may be forged, and are used on the Internet to cheat someone [1].

Generative Adversarial Networks, which designed by Ian Goodfellow and his colleagues in 2014 [2], using two neural network, which are Generator and discriminator, to generate similar data as input, which is as known as "Deep Fake". GAN develop rapidly, on the bright side, it can be used to improve the real-time image resolution [3], or even predict the possible disease image [4]. On the other hand, it also can produce fake images of human faces that cannot be distinguished by humans in just a few second.

Moreover, recently, human activity has been limited by the pandemic. Every one is required to wear the face mask when they go outside, or contact with others. Many corporation or public place also using monitor and machine learning technique to detect whether the visitor follows the criteria. As the result, high accuracy classification is important.

Biological data, (such as human faces, fingerprints) often contains large-scale data size, especially the image quality of cameras is getting higher, which require a certain amount of computing resources and memory, so it is ideal to use convolutional neural network (CNN) model. CNN is generally used at analyzing visual data, such as image or photo. The convolutional layer in CNN can filter out the crucial features from the big image data, then pass those data into the original full connected network. This process can reduce the original workload and focus on the most important elements.

About classification problem, there have been plenty of researches. Many high performance machine learning models have been published. For every dataset, or called

Dr. M. Aledhar and P. Lu are with the College of Computing and Software Engineering, Kennesaw State University, Marietta, GA, 30060 USA. Emails: maledhar@kennesaw.edu and plu5@students.kennesaw.edu

as problem, programmer will try to build a specific and suitable model, for getting high performance and high resource efficiency. Nevertheless, it is very time consuming and difficult for the beginner or non-expert. So we pop up this question: Is that possible to build a new model from the existing model, which might suitable for every similar problem with acceptable performance?

In this research, there is not new method or dataset going to present, but two experiments have been done:

First, what will be the performance when we combine two similar datasets. The first dataset is real human face and fake human face dataset, and the second dataset is real human face with face mask or not.

Second, as the mention before, here is going to combine two datasets that were collected from different resource, the total size of datasets and the image size in the dataset are different. The influence of various data attribute is also tested in this experiment.

The rest of this report is organized as follows: Related work is discussed in Sec II. Sec III provided the detail about existed dataset and CNN-based models. In Sec IV, introducing the model and dataset after combination. Results and discussions is stated in Sec V and the final Sec VI is my conclusion of this paper.

2 Related works

2.1 Object detection

Object detection problem is wildly used in many field. Some tasks require high performance, which means that detection should be completed in a short time. For example, self-driving cars. It does not need 100% accuracy classify the object, but need to know something is there. On the other hand, certain tasks ask for high accuracy, and these tasks do not care about time, such as face-mask checking.

Under normal circumstance, to check whether a person is wearing face-mask or not need complex operations. First of all, under normal circumstances, it should be real-time detection, that is, the program needs to detect the position of the face in the video [5]. After that, since the monitor is usually set at a higher position, the program needs to process facial images from different angles [5].

Moreover, according to (H. H. Nguyen. et, al. 2019) [6], in normal, there are two methods for object detection. One is building more complex algorithm to filter out the target's important feature. The other method is machine learning, it only needs to provide a data set, and if every setting works well, the program will run properly [6].

As the reason mention above, in this paper, we used the normal positional face data set for reduce the error and workload that may occur in the preprocessing stage, and we can focus on the machine learning part. In addition, based on the previous reasons [5], [6], We don't need to label or detect objects, so the "object detection" stage in this experiment can be simplified to the "classification" problem.

2.2 Classification

When talking about the classification problem, in the paper [1], [4]–[7], all pointed out that using CNN model is the ideal method, due to the images, the data size must be quite large. CNN-based model is most ideally and commonly in machine learning.

Plenty of CNN-based models have been published, in paper [8], the author did series of the comparison between some famous CNN models, like, VGG19, Alexnet, etc. The results shown in this article seem to indicate that the latest model may have better performance. However, it still depends on the problem.

For CNN classification problem, the training type could be "Supervised" or "Unsupervised". In paper [7], the proposed that CNN model, which is built on unsupervised learning method, could reduce the training burden, and have better recognition/classification. In their situation, due to using very big dataset, which training time is in unit of day. On the other hand, in my situation, we did not use large data set, so I still used "Supervised learning", which we am familiar and easier to handle.

3 Techniques used in the study

3.1 Dataset collection

In this paper, we used two datasets for Kaggle website. First dataset is "Real and Fake Face Detection" [9]. In this data set, there are 960 fake human faces, and 1081 real human faces. About the fake human faces, they are generated by Generative Adversarial Networks (GAN) model. Mean image size is 600 times 600 pixels. This data is neat, but has smaller data size.

Second dataset is "Face Mask ˜12K Images Dataset" [10], it contains 5000 human faces with face masks and 5000 human faces without mask. Mean image size is 152 times 152 pixels. These dataset is not that neat, but it contains more image data.

3.2 CNN-based machine learning model

In this paper, we also pick up two exist model by others on Kaggle website. First model from M.Michael [11] used CNN – VGG16, with face masks dataset [10]. Second model is from M. Martin [12]. He used CNN – MobileNet with Real and Fake human face dataset [9].

3.2.1 VGG16 [13]

VGG16 is present by K. Simonyan and A. Zisserman. VGG16 contain 12 convolution layer, 5 pooling layer and 3 full connect layer. With smaller kernel size filter, which is 3 by 3, VGG16 can filter out more features from complex input. It has really good learning accuracy, however, due to the small filter, it causes longer training time.

3.2.2 MobilenetV2 [14]

MobilenetV2 is present by a group of researchers from Google at 2018. They presented a new layer module: The inverted residual with linear bottleneck. In short, with this new architecture, the module can pass the feature data in lower dimension, which can reduce the time, and it can filter out the feature data in higher dimension, which can pick out more important features.

3.3 model Implementation

In this paper, we used VGG16 structure. The reasons are below:

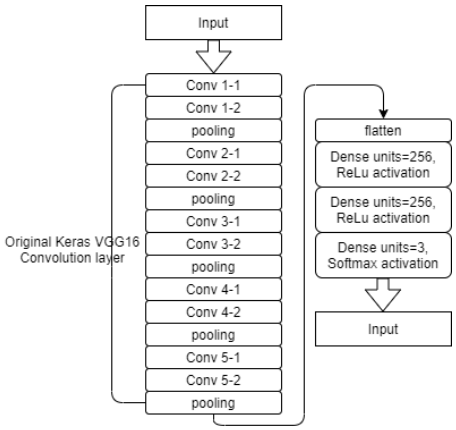

Fig. 1: VGG16 based structure

3.3.1 i) No need to care about the performance, what we need is higher accuracy:

In this paper, it is still in experimental stage, one of the main purpose is to compare the accuracy before and after merger dataset. Moreover, this model does not use real-time data, and does not apply in real environment. Therefore, no need to care about the training time, but as high as possible of the accuracy.

3.3.2 ii) VGG16 model is easier to understand:

VGG16 has simple structure. VGG16 is the most common model, so there is more information, more article, that are useful for me to understand and handle.

As the discussion above, we used the VGG16 model for the experiment, but we did not used the built in dense layer. We only pick up those Convolution layers and pooling layers. Before pass through feature data from convolution layer to dense, we have to flat data by calling flatten layer. Then we added up with 2 dense layers with 256 units and ReLu activation as the fully connected layer. Two drop out layers were added up with 0.5 rate to drop out some features in the layers when we training, this layer can reduce the overfitting problem. For the output layer, we had tree classed in the dataset, so we added the last dense layer with 3 units and Softmax activation.

To summarize, the model used in this paper has 13 convolution layers, 5 max pooling layers, 1 flatten layers, 2 full connected layers, 2 dropout layers and the output layer(Fig. 1).

3.4 Activate function

3.4.1 Rectified Linear Unit activation function [15]

The Rectified Linear Unit (ReLu) function is a machine learning active function, that are commonly used for the image image problem. It default output is between 0 and positive max. That is, if the original output value is negative, the output 0 will be returned; if the original output value is positive, the original value will be returned. With this function, due to the output has a range, it can reduces the cost of computation.

3.4.2 Sigmoid function [15]

The Default output of the Keras Sigmoid function return the output close to 0, if the original output lower than minus five; if the original output value is positive five, it will return the value close to 1. With this output format, Sigmoid function is usually used as the active function of output layer, when the problem need binary classification.

3.4.3 Softmax function [15]

Softmax function can treat as a normalized function, it will return a vector of the classes possibility of output, with the value range between 0 and 1. As the this form of the output, Softmax function is used in this paper for the multiple classification.

4 Experimental Setup

4.1 Environment

Using “Google Colaboratory” [16] as the development environment. The first reason this environment is free. Google provide this free online resources for everyone who has google account. Only need to link the application into personal google drive, and almost every libraries and packages are supported. The second reason is it is powerful. As we mention above, this environment almost contains every library, so no need to install package in the local computer, which might not have enough storage. Beside, Google Colaboratory also provides limited times GPU speed up hardware, which can speed up the training stage.

4.2 Library/Package

The main library used in this paper is Keras [15]. This is a powerful library, it contains many in-build function for machine learning. For example, it has VGG16 structure function, what users need to do is import the library.

Although the user needs to manage many parameters, it also gives a lot of flexibility. All library used: “numpy”,“rarfile”, “os”, “time”, “Keras.layer”, ”Sequential from Keras.models”, ”VGG16 from Keras.applications”, “Adam from Keras.optimizers”, “ImageDataGenerator from Keras”, “matplotlib.pyplot”, “classification_report & confusion_matrix from sklearn.metrics”

4.3 Dataset combination

In these two datasets [9], [10], “Real human face” in first dataset, and “Human face without face mask” are actually the same data. That is, there is actually three categories in these dataset. Therefore, after merging, dataset turned to be, “Real human face” with 6081 images, “Fake human face” with 960 images, and “WithMask” with 5000 images.

Obviously, the dataset is unbalanced, only 960 images for “Fake human face”. We used openCV [17] to do image augmentation. Generate 6000 images from the original 960 images, via “clockwise/anti clockwise rotation”, “horizontal flip”, “adding noise”, and “blurring image”.

As the result, there are two types of dataset (60% training, 30% validation, 10% testing):

i) Unbalanced dataset with three categories:

6081 images in “Real/Without mask”

5000 images in “With mask”

960 images in “Fake face”

ii) Balanced dataset with three categories:

5000 images in “Real/Without mask” (random picked from original 6081 dataset)

5000 images in “With mask”

5000 images in “Fake face” (random picked from generated and original 960 dataset)

4.4 Process Overview

With the model and the data set discussed in the previous section, in this section we will introduce the whole process of our experiment briefly.

4.4.1 File reading

We loaded the data by calling “flow_from_directory” function, and because our data classed change from 2 to 3, we have to change the class mode parameter from “binary” to “categorical”

4.4.2 Training with model

In VGG16 model, we only keep the convolution layers and pooling layer, and added other four layers that are suitable for our purpose. Because the classes of the problem changed for original 2 to 3 categories classification problem, we change the Sigmoid function, which suitable for binary classificaiton, to the Softmax function, that suitable for multiplu classes.

In addition, the loss function has also changed from binary cross-entropy to categorical cross-entropy, also due to the increasing of the output categories. In the original model built by others, because there was only two classes, binary cross-entropy worked well and calculated the cross entropy loss between prediction and real data. When number of classes larger than two, binary method is not suitable, so we used categorical cross-entropy for loss calculation.

4.4.3 Testing model with test set

For a model evaluation, the are so many method. In the original model that I pick from the Kaggle website, they all used the final accuracy of the training process. It is a simple method. However, in this way, we cannot understand over all of the data. The model might have over fitting and under fitting problem, or the model might not perform well with the complete unknown data.

As the result, we not only used the training accuracy to evaluate the model, but also validation set accuracy and the performance report of testing data. We use visual representations to represent the loss and accuracy between training data and validation data. If there are the gap between the result of those two dataset, we can easily notices that there might have some problem, such as over-fitting.

To evaluate model with test set, we import the sklearn library [18] and used the built in metrics functions to calculate precision, recall, and f1-score. F1-score is the harmonic mean of the value of precision and recall. The larger the score, the better the performance of the model.

Precision provides the rate of the correctly retrieved to the all sample with positive retrieve. For instance, in our experiment, the Precision of real face represents the ratio of actually real faces in all samples that are predicted to be real faces. Recall of the test data represent the rate of the data that are successfully retrieved. In our model, the Recall of real face is (real faces) / (real faces + non-real but retrieved as real faces)

5 Results and Discussion

5.1 Original model experiment

We executed the built model and its corresponding data set, to check how the performance of each (Table 1). For the first model from M.Michael [11] with face mask dataset, it has 99% accuracy on training set, and in M. Martin’s MobileNetV2 [12] model with real-and-fake dataset, it only get 49% accuracy on training set. In M. Martin original experiment in Kaggle notebook, it had 99% accuracy on training set.

The reason for the different results might be whether validation data is used. With my limited knowledge about machine learning, training set is for minimizing the loss function, meanwhile, validation set is for choosing the best hyper parameters. As the result, we think validation data is required, so we set 30% from the original data to be validation set.

	Loss	Accuracy	Val_Loss	Val_Accuracy
VGG16	0.0036	0.9987	0.9987	1.0000
MobileNetV2	0.6933	0.4944	0.6932	0.5172

TABLE 1: The original models with their own datasets. MobileNet did not match the original result on Kaggle website.

5.2 Un-Balanced dataset with VGG16

The first experiment using unbalanced dataset, which we have mentioned above, just combination two datasets without any image augmentation. There is a very large different number of samples for each category. The result is shows in figure 1.

From accuracy and loss image (Fig 1a, 1b), we can see that there is a really big gap in training dataset and validation dataset. Training data have almost 100% accuracy in the final, however, the validation result did not have any improve.

Moreover, from heat map and classification report of test data(figure 1, c), it shows the unbalanced data set caused the lower f1-score on the smaller size of the classes, and also we have the lower average accuracy.

Accordingly, we may say this models with the dataset is “Overfitting”, which means that this model is over matched to the training set. This will make validation and test data have poor performance.

5.3 Balanced datasets with VGG16

After image augmentation, three datasets have total 5000 images for each category, with the discussion above, we assigned 60% for training, 30% for validation, 10% for test. The image shows in figure 2, and also contain loss, accuracy, and test result. With the balanced dataset, although the visual representation of loss and accuracy showed the overfitting problem as previous, we have the higher f1-score of all the classes. That is, our model did have some progress.

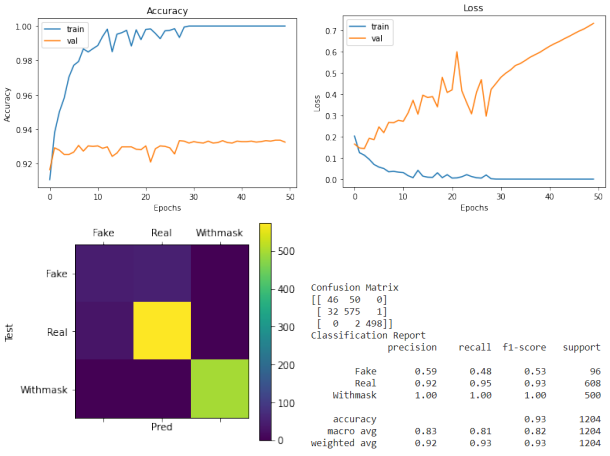

Fig. 2: VGG16 with unbalance dataset a) Accuracy b) Loss c) Heat map of confusion matrix of test c) Classification Report of test

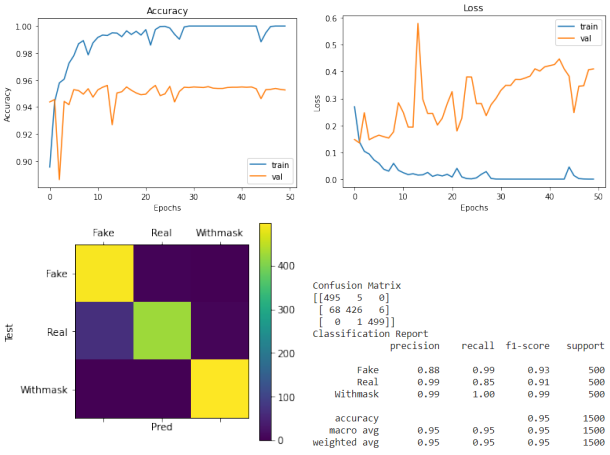

Fig. 3: VGG16 with balance dataset a) Accuracy b) Loss c) Heat map of confusion matrix of test c) Classification Report of test

5.4 Deal with Overfitting problem

There are three common methods to handle overfitting issue:

1) Give more training data For this solution, we modify some parameters in Keras model.fit function. Instead of just using "batch_size" and "epochs". I used "steps_per_epoch=100, epochs=50, batch_size=40" with "ImageDataGenerator" to give more images with augmentation. Total data input turned to be 200,000 (100 50 40) from 9,000.

2) Increase the Regularization We have already used rescale=1./255 in previous experiment, so we did not change anything.

3) Change architecture We focused on VGG16, so we did not change anything

4) Smaller sets of features Add drop out layer before output layer, and the rate set at 0.5

After above change, the result show in figure3, we successfully reduced the "overfitting" effect. There was not large gap between loss and validation loss (figure 3.a, 3.b.) Moreover, we have have the 94% average accuracy and average 0.91 of f1-score.

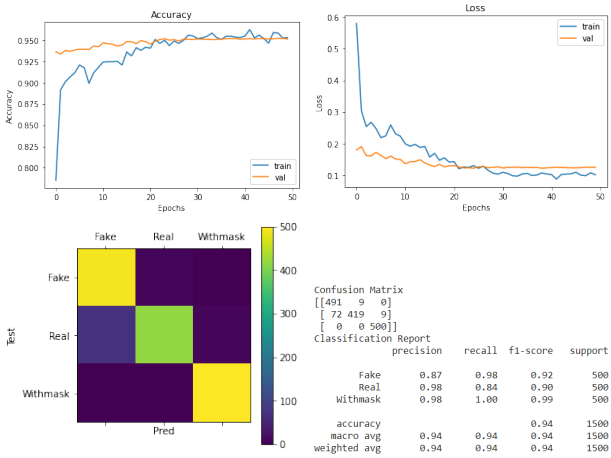

Fig. 4: VGG16 with balance dataset after parameters change.a) Accuracy b) Loss c) Heat map of confusion matrix of test c) Classification Report of test

6 Conclusion and Recommendations for Future Research

In this paper, we did not focus on deeper or newer information, but wider concept of CNN machine learning. About the two main purposes: First, we understand that balanced and unbalanced dataset will influence the test f1-score, due to the difference size of the sample in each classes. Second, for a new question, which is three categorical classification problem in this article, we cannot just built a model exactly the same as the existing model, but still need to change some parameters. After that, we can possible get a model that meets the minimum require, that might give us a brief first sight of a new problem. When we look for the same module, we should not look for similar data sets, but for similar problems. That is, for instance, in this article, if I want to seek for the models suitable for my project, we should find "multiple classification problems", instead of finding model for "face binary classification problem".

To summarize, we did learn a lot of knowledge about machine learning, especially on CNN-based model, and got the answer for those questions, which are the purpose of this article.

For the future research, there are 2 directions. First, try to find the fake face with face mask dataset. Second, for handling overfitting, might can change the model architecture.

References

[1] C.-C. Hsu, C.-Y. Lee, and Y.-X. Zhuang, "Learning to detect fake face images in the wild," in *2018 International Symposium on Computer, Consumer and Control (IS3C)*. IEEE, 2018, pp. 388–391.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[3] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[4] A. Zhavoronkov, Y. A. Ivanenkov, A. Aliper, M. S. Veselov, V. A. Aladinskiy, A. V. Aladinskaya, V. A. Terentiev, D. A. Polykovskiy, M. D. Kuznetsov, A. Asadulaev *et al.*, "Deep learning enables rapid identification of potent ddr1 kinase inhibitors," *Nature biotechnology*, vol. 37, no. 9, pp. 1038–1040, 2019.

[5] G. Deore, R. Bodhula, V. Udpikar, and V. More, "Study of masked face detection approach in video analytics," in *2016 Conference on Advances in Signal Processing (CASP)*. IEEE, 2016, pp. 196–200.

[6] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, "Multi-task learning for detecting and segmenting manipulated facial images and videos," in *2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. IEEE, 2019, pp. 1–8.

[7] K. Nguyen, C. Fookes, and S. Sridharan, "Improving deep convolutional neural networks with unsupervised feature learning," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 2270–2274.

[8] A. Khodabakhsh, R. Ramachandra, K. Raja, P. Wasnik, and C. Busch, "Fake face detection methods: Can they be generalized?" in *2018 international conference of the biometrics special interest group (BIOSIG)*. IEEE, 2018, pp. 1–6.

[9] C. Intelligence and K. Photography Lab, Department of Computer Science Yonsei University, "Real and fake face detection," 2019, https://www.kaggle.com/ciplab/real-and-fake-face-detection/metadata.

[10] S. a. L. K. H. I. Ashish Jangra, "Face mask 12k images dataset," 2020, https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset/metadata.

[11] U. S. Michael Macht New York New York, "mask detection, visualization," 2021, https://www.kaggle.com/michaelcripman/mask-detection-visualization.

[12] M. Matta, "real vs fake face," 2019, https://www.kaggle.com/martin1234567890/real-vs-fake-face.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[15] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[16] E. Bisong, "Google colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 2019, pp. 59–64.

[17] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.