Applying Transfer Learning to Stanford Dog Data Set using Keras Pre-trained Inception-ResNet-V2 and Inception-V3 Model Structures

Mahsa Rezaei Firuzkuhi
Computer Science Department
University of Houston
Houston, USA
mrezaeif@cougarnet.uh.edu

***Abstract*—This paper proposes an Inception-Resnet based deep learning structure to classify the breed from the dog picture in a big dog image dataset. Dog breed categorization is a specific application of the convolutional neural networks because it is considered as a fine-grained image classification problem and on the other hand, we have a very small training example dataset and short computing power. In this paper, we have repurposed two of the best pre-trained image classification models named Inception-ResNet-V2 and Inception-V3 as a fixed feature extraction mechanism in the KERAS tool. We use the transfer learning with image augmentation to get better model generalization and achieved a validation accuracy of around 90% using the Inception-ResNet-V2 and a validation accuracy of around 80% utilizing the Inception-V3 model.**

Keywords—Image Classification, Transfer Learning, Convolutional Neural Network, Inception-V3, Inception-ResNet-V2, Dog Breed

I. INTRODUCTION

The image classification problem is defined as a given set of images that are all labeled with a single category and we are asked to predict these categories for a novel set of test images and measure the accuracy of the predictions. There are a variety of challenges associated with this task, including viewpoint variation, scale variation, intra-class variation, image deformation, image occlusion, illumination conditions, background clutter etc.

Computer Vision introduces a data-driven approach to solve image classification. Instead of trying to specify what every one of the image categories of interest looks like directly in code, they provide the computer with many examples of each image class and then develop learning algorithms that look at these examples and learn about the visual appearance of each class. In other words, they first accumulate a training dataset of labeled images and then feed it to the computer in order for it to get familiar with the data. Fine-grained image classification aims to recognize the subcategories of a certain category. Due to the little difference between subcategories and the large difference within the subcategories, it is more difficult than the general object classification.

Dog image classification is one of the fine-grained image classification problems. In this paper, we demonstrate how to train and test a pre-trained Convolutional Neural Network, based on KERAS[7] (which uses the programming language Python), capable of identifying the breed of a dog in a supplied image. Success will be defined by high validation and test accuracy. This is a supervised learning problem, specifically a multiclass classification problem. For the dog classification problem, dogs of different breeds could share almost all the same visible features, like the same color, same facial characteristics, but they are still classified as different breeds. Dogs of the same breeds, however, could differ significantly depending on the age, position, size and even color. In dog image classification we are facing several challenges.

A. Convolutional Neural Networks

The most popular architecture used for image classification is Convolutional Neural Networks (CNNs)[10]. A typical use case for CNNs is where you feed the network images and the network classifies the data. A typical CNN has two parts. The first part is the convolutional base, which is composed of a stack of convolutional and pooling layers. The main goal of the convolutional base is to generate features from the image[11]. The second part is the classifier, which is usually composed of fully connected layers. The main goal of the classifier is to classify the image based on the detected features. A fully connected layer is a layer whose neurons have full connections to all activation in the previous layer.

B. Transfer Laerning

From a deep learning perspective, the image classification problem can be solved through transfer learning. Actually, several state-of-the-art results in image classification are based on transfer learning solutions [5][6]. When we are repurposing a pre-trained model for our own needs, we start by removing the original classifier, and then we add a new classifier that fits our purposes. For example, freezing the convolutional base, which is an extreme situation of the train/freeze trade-off. The main idea is to keep the convolutional base in its original form and then use its outputs to feed the classifier. We are using the pretrained model as a fixed feature extraction mechanism, which can be useful if the computational power is short, our dataset is small, and/or pre-trained model solves a problem very similar to the one we want to solve.

C. Pre-trained Model

Numerous deep convolutional feature-based algorithms have been proposed, which have advanced the development of fine-grained image classification. From the wide range of pre-trained models that are available, we have selected Inception-V3, a convolutional neural network (CNN) that achieves a new state of the art in terms of accuracy on the ILSVRC image classification benchmark and Inception-ResNet-v2 which is a variation of the earlier Inception V3 model which borrows some ideas from Microsoft's ResNet papers [1][2][3]. Most image classification techniques are trained on ImageNet[9], a dataset with approximately 1.2 million high-resolution training images. Models trained on ImageNet are typically deep CNNs with a number of fully connected output layers that have been trained to categorize

the hidden features exposed by the convolution layers into one of those 1000 categories.

The act of training a neural network, even a relatively simple one, can be extremely computationally expensive. Many companies use server racks of GPUs dedicated to this kind of task. We work on our laptop with corei7 CPU and 16 Megabytes memory on the windows operating system. The basic approach is to construct and train a very simple CNN on the dataset and evaluate its performance. Convolutional neural network (CNN) is by all accounts the best machine learning model for image classification, but in this case, there are not enough training examples to train it. It would not be able to learn generic enough patterns off this dataset to classify different dog breeds. Most likely, it will just overfit to this small amount of training examples so that accuracy on the test set will be low.

There are two possible approaches to mitigate the lack of training examples. One approach is the transfer learning with bottleneck features. In this approach, an existing CNN which has been trained on a massive image library will be adapted to the application in order to transform the input images into bottleneck features. The bottleneck features are the abstract feature representations of the images. In this paper, we use transfer learning with image augmentation. This approach is similar to the bottleneck features approach, but we will attempt to get better model generalization by creating a model which is a stack of a pre-trained bottleneck feature CNN with a custom output layer for our application, and we feed it input images which are randomly augmented by pictographic transformations. We leverage two existing CNNs which have been pre-trained to recognize features of general image data and terminating in a fully connected layer with 120 nodes — one for each class (breed) we are trying to predict.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 3 introduces the proposed method. Section 4 presents the experimental results and evaluations and finally section 5 concludes the paper.

II. RELATED WORK

Deep Learning-powered image recognition is now performing better than human vision on many tasks while using Convolutional Neural Networks. The image classification solution with transfer learning solutions are trained on the ImageNet database have better performance and higher accuracy than the previous solutions in which the accuracy is achieved by training a model from scratch. ImageNet is so much larger than other databases and also contains a considerable amount of dog images which results in better pre-trained models. Image classification methods to tackle the dog breed detection based on the dog image fall into two categories: solutions without leveraging a pre-trained model and methods which utilize a pre-trained model to create the CNN.

A. Solutions without a pre-trained model

The proposed ideas for dog image classification problems which are not using a pre-trained model are mostly released before 2014. This Paper [13] implement a fine grain feature detection and an automatic face detection with localization of eyes and nose relative to the dog's face. They attempt to align the face and extract a scale-invariant feature transform (SIFT). They successfully classifying dog breeds with 67% accuracy. The fact that both of a dog's eyes must be visible in relation to their face is the weakness of their solution.

Another method for achieving high accuracy (52%) on the Stanford Dogs dataset is by using Selective Pooling Vectors [20] which encodes descriptors into vectors and selects only those that are below a certain threshold of quantization error. The classification likelihoods of various classes are determined with respect to a nonlinear function.

B. Solutions utilizing a pre-trained model for CNN

Some of proposed ideas for dog image classification problems which are leveraging a pre-trained model are considered. Zeiler and Fergus in [14] use a CNN model trained on ImageNet as frozen layers and train a new classifier and apply SoftMax function on top of it using the training samples of the target data-set and report high accuracy image classification.

Paper [16] proposes the utilization of transfer learning in successfully identifying dog breeds from large image datasets instead of building a new classification model from scratch. The author leverages the Google's Inception-V3 to correctly classify an initial suite of 275 dog images belonging to a set of 11 unique dog categories. The paper shows the issue with using pre-trained neural networks, in which their flexibility has the potential tendency of overfitting data and introduces the process of augmenting the data by simply rotating the images to reduce the overfitting issue. The paper claims that the smaller dataset size relative to the original dataset size is required to fine-tuning the convolution neural network for the dog image classification and on the other hand, he changed the content similarity of the dataset to the degree that only a linear classifier should be used.

The article [17] demonstrate the process of combining the transfer learning and neural network with leveraging an inception model and how the image after entering the inception model converts to the output information that goes through several fully connected layers and results to a set of probabilities which will be assigned to each class. The paper [18] proposed an effective approach to use the transfer learning and pre-trained models which are used to classify the dog breeds. they worked on two neural networks which are Inception-V3 and VGG16 pre-trained on the Stanford Dog dataset and their accuracy is compared.

In this paper [19], an effective approach proposed to use the transfer learning to pre-trained models which are used to classify the dog breeds. They consider three major Transfer Learning scenarios which depend on the expanse of the new dataset. For example, if we have a dataset that is small and similar to the authentic dataset, then, it isn't advisable to fine-tune the CNN model and if we have a dataset that is large but similar to the original one, then we can fine-tune through the full network. This algorithm is not able to distinguish humans from dogs and they did not apply the image data augmentation to increase the accuracy.

III. PROPOSED METHOD

One way that we can dramatically improve performance is utilizing the transfer learning. In this technique, we can leverage an existing CNN which has been pre-trained to recognize features of general image data and adapt it for our own purposes. Keras has a number of such pre-trained models.

Each is a model that has been trained on a repository of images known as ImageNet which contains millions of images distributed across 1000 categories. We can take one of those pre-trained models and simply replace the output layers with our own fully connected layers, which we can then train to categorize each input image as one of 120 dog breeds.

A. *Inception-ResNet-V2 and Inception-V3*

Numerous deep convolutional feature-based algorithms have been proposed, which have advanced the development of fine-grained image classification. From the wide range of pre-trained models that are available, we have selected Inception-V3, a convolutional neural network (CNN) that achieves a new state of the art in terms of accuracy on the ILSVRC image classification benchmark and Inception-ResNet-v2 which is a variation of the earlier Inception V3 model which borrows some ideas from Microsoft's ResNet papers [1][2][3]. Most image classification techniques are trained on ImageNet[9], a dataset with approximately 1.2 million high-resolution training images. Models trained on ImageNet are typically deep CNNs with a number of fully connected output layers that have been trained to categorize the hidden features exposed by the convolution layers into one of those 1000 categories. Number of convolutional and fully connected layers for Inception-ResNet-V2 is more than the Inception-ResNet-V3. We used Inception-ResNet-V2 method as a reference method for major part of results because the accuracy of this method is higher than the other method.

We freeze the weights and kernels of the convolution layers, which are already trained to recognize abstract features of an image, and only train our own custom output network. This saves an immense amount of time.

B. *Training Dataset*

The training data set used in our research is the Stanford Dogs Dataset[4] with more than 20K images of dogs of 120 breeds. Every image in the dataset is annotated with the breed of a dog displayed on it.

C. *Augmented input model*

we apply a form of bootstrapping the training data called image augmentation. In models trained on image data, during training, we can apply random rotations, zooms, and translations to the training images. This has the effect of artificially increasing the size of the training data by altering the pixels of the training images while maintaining the integrity of the content. For example, if we rotate the image of a Shitzu by 25 degrees and flip it horizontally, the image should still be recognizable as a Shitzu, but the model will never have seen that exact image during training. This technique will both improve model accuracy over a great number of epochs, and also reduces the chance of overfitting. In order to do this, we compile the entire model together for forwarding and backpropagation and since we still do not want to edit the parameters of the model pre-trained on ImageNet, we will freeze those layers before training.

We have added a second dense layer in order that the model will be able to rely a little less on the pre-trained parameters, and we've also augmented both fully dense layers with dropout regularization. Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or dropped out. This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different view of the configured layer[4]. Dropout decreases overfitting by requiring the network to generalize more during training.

Note that in the dense layer we use a SoftMax activation function; the reason for this is that it has a range from 0 to 1, and it forces the sum of all nodes in the output layer to be 1. This allows us to interpret the output of a single node to be the model's predicted probability that the input was of the class corresponding to that node. In other words, if the second node in the layer has an activation value of 0.8 for a particular image, we can say that the model has predicted that the input has an 80% chance of being from the second class. In figure 1, the first column of the table shows the layer type. The second column shows the output shape of each layer, and third column is the number of parameters. Note that we have more than 9 million model parameters in figure 1. The parameters are the weights, biases, and kernels (convolution filters) that our network is going to attempt to optimize. Now it should be evident why this process is intensely computationally demanding.

block8_9_conv (Conv2D)	(None, 3, 3, 2080)	933920	block8_9_mixed[0][0]
block8_9 (Lambda)	(None, 3, 3, 2080)	0	block8_8_ac[0][0] block8_9_conv[0][0]
flatten (Flatten)	(None, 18720)	0	block8_9[0][0]
dense (Dense)	(None, 512)	9585152	flatten[0][0]
dropout (Dropout)	(None, 512)	0	dense[0][0]
dense_1 (Dense)	(None, 40)	20520	dropout[0][0]

Total params: 58,706,632
Trainable params: 9,605,672
Non-trainable params: 49,100,960

Figure 1: Total number of trainable parameters of the final model

Next, we load the ImageNet model, define a custom fully connected output model, and combine them into one sequential model. There are many loss functions and optimizers we could use here, but the current common convention for multiclass image label prediction uses Adam for the optimizer and categorical cross-entropy as the loss function.

D. *Metrics*

In this project, our goal is to predict the breed of our dog image correctly. As such, we use accuracy as the measurement of choice to improve our model after each iteration. Accuracy is simply how many times our model predicts the dog breed correctly. This indicates the percentage of correctly classified instances during the course of classification. It is calculated as

$$\text{Accuracy} = \frac{True\ positive + true\ negative}{Total\ instances} \times 100$$

Misclassification rate (%): The percentage of incorrectly classified instances of the classifier and can be calculated as

$$\text{Misclassification rate} = \frac{False\ positive + False\ negative}{Total\ instances}$$

IV. RESULTS

A. *Effect of Pre-Trained model on Accuracy*

Figure 2 shows the comparison of Inception-ResNet-V2 and Inception-V3 methods for the different number of dog breeds. In this experiment, we used Sigmoid function as the activation function. The two models take a significant amount of time to train simply because each forward pass has to traverse all of the nodes of the pre-trained model. Each epoch takes around 15 minutes to train on our laptops. As shown in this figure, in both methods Inception-ResNet-V2 and Inception-V3, by increasing the number of Breeds the value of the training accuracy and validation accuracy decreased. Also, for each set with a specific number of breeds, the value of training and validation accuracy for Inception-ResNet-V2 is more than Inception-V3. On the other hand, the simulation time of Inception-V3 is less than Inception-ResNet-V2. This is obvious because the number of convolutional and hidden layers in Inception-ResNet-V2 is higher than Inception-V3. As a result, the model with Inception-V3 will reach its maximum accuracy faster while its maximum achieved accuracy is lower.

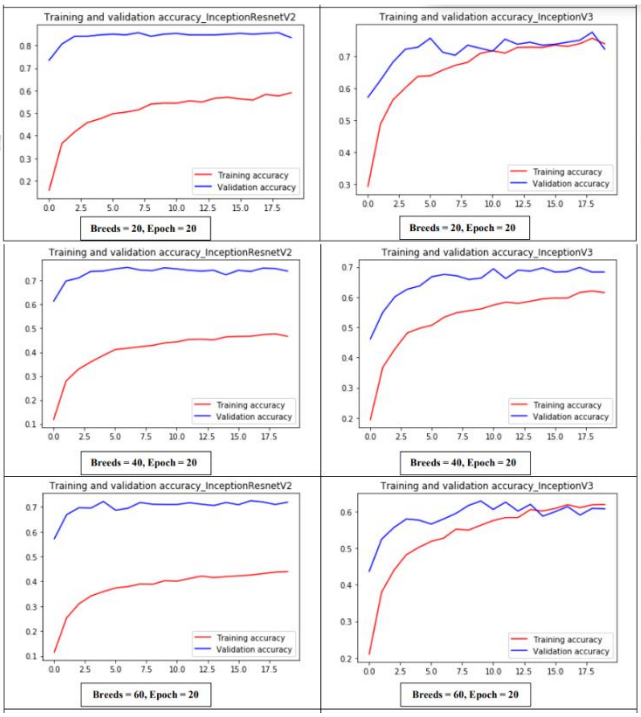

Figure 2: The comparison of Inception-Resnet-V2 and Inception-V3 for 20,40,60 breeds with sigmoid function

Table 1 summarizes the training and test accuracy of the Inception-ResNet-V2 and Inception-V3 methods. This table has two parts. Table 2-a shows the accuracy comparison for training datasets and table 2-b shows the accuracy comparison for validation datasets. As shown in this table, the Training Accuracy of Inception-ResNet-V2 is lower than Inception-V3, but the accuracy of validation dataset is higher. The reason for this result is obvious because when we use neural networks and Dropout, it‘s basically assured that our test accuracy, at its best, will be better than our training accuracy. The details for the effect of using Dropout have been explained later in section 5.

Table 1: The comparison of Inception-ResNet-V2 and Inception-V3 for the different number of breeds. (a) Accuracy on Training Dataset for epoch =20, (b) Accuracy on Validation Dataset for epoch=20

Model	Number of Breeds		
	20	40	60
Inception-V2	0.54	0.47	0.43
Inception-V3	0.75	0.61	0.43

(a)

Model	Number of Breeds		
	20	40	60
Inception-V2	0.85	0.75	0.72
Inception-V3	0.77	0.60	0.54

(b)

B. *The Effect of Activation Functions on Accuracy*

Figure 3 shows the comparison of the accuracy of the model with two different activation functions Relu and Sigmoid. We ran these methods on Inception-ResNet-V2 for 40 breeds and 20 epochs. As shown in this figure, the values of training accuracy and validation accuracy of Sigmoid function are better than the Relu function. In addition, Sigmoid function can work better for the different number of epochs.

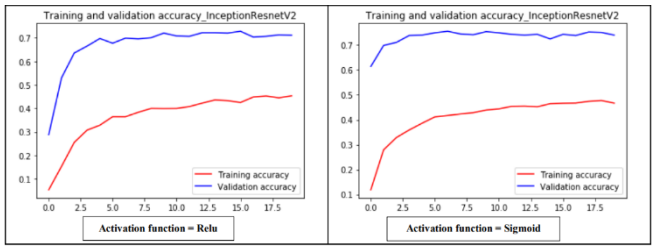

Figure 3: The comparison of two activation functions Sigmoid and Relu for Inception-ResNet-V2 with number of breeds = 40 and epoch=20

C. *The Effect of Dropout rate on Accuracy*

Using dropout is nearly identical to forcing our neural network to become a very large collection of weak classifiers (in other words, an ensemble). As the name implies, an individual weak classifier doesn‘t have much classification accuracy, they only become powerful once you string a bunch of them together. Dropout, during training, slices off some random collection of these classifiers. Thus, training accuracy suffers. Dropout, during testing, turns itself off and allows all of the _weak classifiers‘ in the neural network to be used. Thus, testing accuracy improves. The comparison of different drop-out rate has been shown in Figure 4 for Inception-ResNet-V2, the number of breeds = 40, and epoch=20. As shown in this figure, by increasing drop-out rate, the value of accuracy for training and validation has been increased.

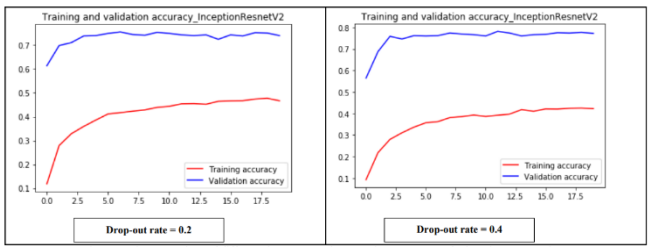

Figure 4: The comparison of different drop-out rate for Inception-ResNet-V2 for number of breeds = 40, epoch 20

D. *Utilizing Confusion Matrix for Accuracy Comparison*

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions [12].

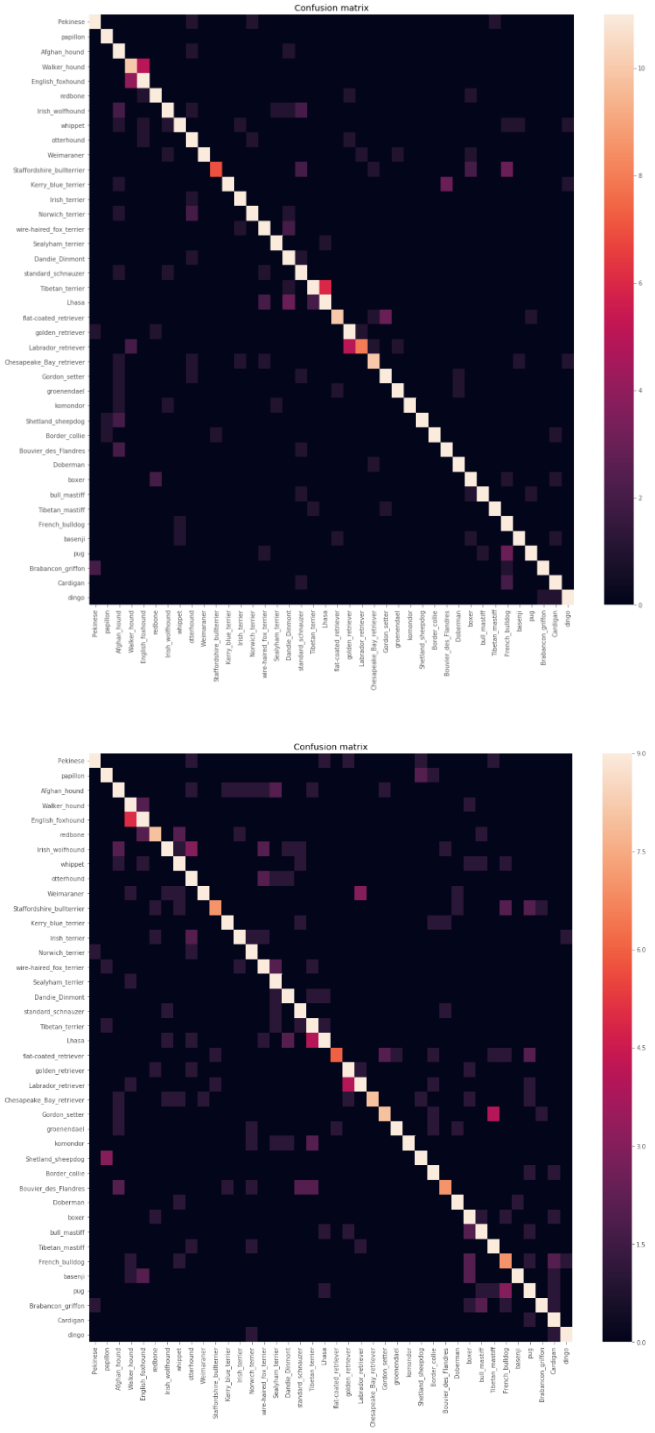

Figure 5: Confusion Matrix for inception-Resnet-V2 (top) and Inception-V3 (bottom)

The colors in the heatmap confusion matrix are a representation of the predictions. Brighter colors are related to higher numbers and darker colors to lower numbers. Brighter colors for off-diagonal elements show misclassification but the diagonal elements indicate correct predictions. Figure 5 shows a comparison of the confusion matrix for Inception-ResNet-V2 and Inception-V3 methods. As shown in this figure, it is obvious that Inception-ResNet-V2 shows better prediction since the color of diagonal elements is brighter than Inception-V3. Also, the rate of misclassification in Inception-V3 is worse than Inception-ResNet-V2 since as you can see the number of bright off-diagonal elements is more than Inception-ResNet-V2. As a result, for the rest of the experiment, we will focus only on the architecture with Inception-ResNet-V2.

E. *Utilizing Confusion Matrix for Misclassified Example*

For example, if we zoom in the confusion matrix for Inception-Resnet-V2 in figure 5 (top matrix), this model can‘t differentiate between a Lhasa Apso and a Tibetan Terrier. The most obvious difference between the two breeds is the size and weight which are the features that are not included in the dog image. In figure 6, the pair Lhasa / Tibetan Terrier is the absolute leader in terms of misclassification which does make sense if we look into how these two breeds of dogs look like.

Figure 6: Sample image of a Tibetan Terrier and Lhasa dog

V. CONCLUSION

As we have observed, it is possible to train a robust image classifier if we have access to pre-trained deep neural networks and modern machine learning libraries like Keras even if we do not have enough training images and/or computing resources. In this project, we are using the transfer learning with image augmentation in order to get better model generalization by creating a model which is a stack of a pre-trained CNN with a custom output layer for our application, and we will feed it input images which are randomly augmented by pictographic transformations. This technique will improve model accuracy over a great number of epochs and reduces the chance of overfitting. As is evident from the accuracy report in figure 2, the Inception-ResNet-V2 achieved a validation accuracy of around 90% while the Inception-V3 achieved a validation accuracy of less than 80% considering 20 breeds from the dog dataset while running for 40 epochs. Since there was also the comparison made between the two models it was done based on all the same parameters i.e., same number of epoch and same number of group size and the training data set was also same. We have also provided the confusion matrix in order to describe the performance of a

classification model for each class and for comparing the classification accuracy of the two classification systems with different pre-trained CNN models. The Inception-Resnet-V2 model achieved better validation accuracy than Inception-V3, while the training accuracy is much lower. The low training accuracy is due to the dropout, because it is never using the full model to evaluate training inputs. We are satisfied that this model (Inception-Resnet-V2) is no longer overfitting to the same degree as the other method. It appears that the validation accuracy and loss have both roughly leveled out. Overall a productive comparison has been made between both the models.

REFERENCES

[1] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016. pp. 770-778.

[2] He He K., Zhang X., Ren S., Sun J. "Identity mappings in deep residual networks". In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9908. Springer, Cham. https://doi.org/10.1007/978-3-319-46493-0_38

[3] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. "Inception-v4, Inception-ResNet and the impact of residual connections on learning". *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[4] Khosla, A., Jayadevaprakash, N., Yao, B., Li, F. F. "Novel dataset for fine-grained image categorization: Stanford dogs". In Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC), 2011.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet classification with deep convolutional neural networks". In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12), USA, 1097-1105.

[6] Simonyan, K., & Zisserman, A. "Very deep convolutional networks for large-scale image recognition". CoRR, abs/1409.1556.

[7] Chollet, F., 2015. "Keras".

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting," 2014. 1929-1958.

[9] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., "ImageNet: A large-scale hierarchical image database". In Computer Vision and Pattern Recognition, CVPR. IEEE Conference on (pp. 248–255). 2009

[10] Voulodimos, A., Doulamis, N., Doulamis, A. and Protopapadakis, E., "Deep learning for computer vision: A brief review". Computational intelligence and neuroscience, 2018.

[11] Chollet, F., "Deep learning with python". Manning Publications Co. 2017.

[12] https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

[13] Liu, Jiongxin & Kanazawa, Angjoo & Jacobs, David & Belhumeur, Peter. "Dog Breed Classification Using Part Localization." (2012). 7572. 172-185. 10.1007/978-3-642-33718-5_13.

[14] Zeiler M.D., Fergus R. "Visualizing and Understanding Convolutional Networks." In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8689. Springer, Cham. https://doi.org/10.1007/978-3-319-10590-1_53

[15] H. A. Shiddieqy, F. I. Hariadi and T. Adiono, "Implementation of deep-learning based image classification on single board computer," *International Symposium on Electronics and Smart Devices (ISESD)*, 2017, pp. 133-137, doi: 10.1109/ISESD.2017.8253319.

[16] Pratik Devikar, "Transfer Learning for Image Classification of various dog breeds." International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 5, Issue 12, 2016. ISSN: 2278 – 1323

[17] Kirill Panarin, "Dog Breed Classification Using Deep Learning: Hands-on Approach." Towards Data Science, https://towardsdatascience.com/dog-breed-classification-hands-on-approach-b5e4f88c333e

[18] A. Varshney, A. Katiyar, A. K. Singh and S. S. Chauhan, "Dog Breed Classification Using Deep Learning," *International Conference on Intelligent Technologies (CONIT)*, 2021, pp. 1-5, doi: 10.1109/CONIT51480.2021.9498338.

[19] M. V. Sai Rishita and T. Ahmed Harris, "Dog Breed Classifier using Convolutional Neural Networks," *International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, 2018, pp. 1-7, doi: 10.1109/ICNEWS.2018.8903980.

[20] Chen G, Yang J, Jin H, Shechtman E, Brandt J, Han TX. "Selective pooling vector for fine-grained recognition", *Proceedings - 2015 Ieee Winter Conference On Applications of Computer Vision, Wacv 2015*. 860-867. DOI: 10.1109/WACV.2015.119