E-ISSN 2581 – 7957
P-ISSN 2277 – 3916

CVR Journal of Science and Technology, Volume 17, December 2019
DOI: 10.32377/cvrjst1711

Dog Breed Identification Using Convolutional Neural Networks on Android

Dr. D. Durga Bhavani[1], Mir Habeebullah Shah Quadri[2], Y. Ram Reddy[3]
[1]Professor, CVR College of Engineering/CSE Department, Hyderabad, India
Email: drddurgabhavani@gmail.com
[2]PG Scholar, CVR College of Engineering/ CSE Department, Hyderabad, India
Email: quadrishah846@gmail.com
[3]Software Engineer, Eximius Design India Pvt. Ltd., Bengaluru, India
Email: ramreddyy@eximiusdesign.com

Abstract: **Identifying the breed of a dog, is a challenging image classification problem. In this paper, we implement an android application that identifies the breed of a dog via image analysis, using a Convolutional Neural Network (CNN) and transfer learning model. The android application lets the user click or upload a picture of a dog. It then pre-processes the image and extracts the features required for testing. Prediction of dog breed is done using CNN and transfer learning. We have used Stanford's standard dog dataset for training the model and achieved an accuracy of 94% on the testing data.**

Index Terms: **Dog Breed Identification, Convolutional Neural Networks, Pre-trained Models, Android**

I. Introduction

Dogs are the most preferred among pets. Potential customers who are looking to buy a dog may do their research by scouting dogs owned by other people. It may not always be possible to approach the owner and inquire about the breed the dog belongs to. Additionally, it is possible that the owners themselves may be unaware of the breed of their dog. In this paper, we have tried to address this market requirement by automating the process of finding a dog breed via an android application that lets you know the breed of the dog, simply by snapping its picture.

Convolutional Neural Networks (CNN) works like human vision. In simple terms we can innately classify things using our vision. CNN provides a software, the ability of image identification and recognition by building a mathematical model and implementing it in the form of an algorithm.

CNN is a class of the feed forward artificial neural networks and deep learning. It is similar to regular neural networks, except that it takes every input as an image. Thus, the assumption allows us to conceal few properties to the architecture thereby making the feed forward function more efficient. CNNs are proven to be effective in analyzing the visual imagery. They use multi-layer perceptron's and usually require less preprocessing in comparison with other image classification algorithms. CNNs have at least one fully connected layer preceded by the desired number of fully convolutional layers as a standard multi-layered network. The input which is given in the structure image format is efficiently utilized by the design of the CNNs, i.e. they can be easily trained. One of the benefits of CNNs is that it is translation invariant, also referred to as shift invariant or space invariant artificial neural networks.

There are three main layers used in the CNNs, namely *convolutional layer, pooling layer and fully connected layer,* (CONV-POOL-FC). Each layer can be repeated a required number of times in order to accomplish the desired output. CNNs divide the images into smaller parts/features and match them individually.

- The input image comprises of raw pixel values represented in a matrix format (m x m x r, m rows, n columns, r channels, for an RGB image r=3).
- CONV layer computes the output of neurons connected to the local region by computing the dot product of the sub image and *k*-filters (size - n x n x q) and finding the average value to obtain *k*-filtered images.
- POOL layer performs down sampling operation along the spatial dimensions i.e., it takes the stack of filtered images and reduces the size of the image matrix to give an optimal output image.
- FC layer will compute the class scores, resulting in volume of size [1x1x a]

Thus, CNNs transform the input image, layer by layer from the original raw pixel values to the final class scores.

Transfer learning is a machine learning technique where an already trained model is used on another related task to attain better results. We have used convolutional neural networks and transfer learning in conjunction, to identify the breed of a dog. Our main aim is to build an android application, using which, a user can identify the breed of a dog. We intend to demonstrate how a hybrid model of pre-trained models for feature extraction gives better results.

II. Related Work

A. Literature review

Research has been done on depth wise separable convolutions in neural computer vision architectures and their effectiveness when used in place of *Inception modules* [1]. An architecture based on this idea was proposed, called Xception which have same parameter count as Inception V3 but are easy to use as regular convolution layers.

In another work, in order to speed up the training process, non-saturating neurons and a very efficient GPU implementation of the convolution operation are used [2]. There is a discussion on "dropout" method which is an effective and recently developed regularization method for reducing over fitting in the fully connected layers. A conclusion was reached by stating that the usage of very large and deep convolutional nets on video sequences where the temporal structure provides very helpful information that is missing or far less obvious in static images, proves to be effective.

Another method proposed, suggested several design principles to scale up convolutional networks which contribute to high performance vision networks having relatively modest computation cost compared to simpler, more monolithic architectures [3]. The study was done in the frame of reference of the Inception architecture and comparison was drawn between the error rates with the others proving the discussed method efficient. This method demonstrated that high quality results can be reached with very low receptive field resolution which might be helpful in detecting small objects in various systems.

In another study, the architectures-Inception-ResNet-v1, Inception-ResNet-v2, Inception v4 have been presented and discussed by the authors [4]. There is also a discussion regarding the variation of training speed with the introduction of residual connections for the Inception architecture.

A study demonstrated flexible and learning scalable, convolutional cells from data transfer to multiple image classification tasks [5]. The main aim in the mentioned approach is to design a search space that decouples the complexity of architecture from the depth of a network. The study concluded that we can use the demonstrated architecture to perform ImageNet classification with reduced computational budgets that outperform streamlined architectures targeted at mobile and embedded platforms.

Depth in visual representations is another methodology discussed for large scale image classification [6]. In this study, an assessment was made for very deep convolutional networks. By evaluating very deep convolutional networks, it was demonstrated that the representation depth is profitable for the classification accuracy, and that we can achieve state-of-the-art performance on the ImageNet challenge dataset using a traditional CNN architecture with considerably increased depth.

In another study, a system for automatically identifying dog breeds through images is described, implemented, and evaluated [7]. It follows a three staged system, in which all three stages will be at least briefly characterized, with more significance given to the implementation and evaluation of the breed identification network. The predictive models used in the above-mentioned stages have ultimately taken the form of convolutional neural networks retain in several important differences.

Work has been done to demonstrate that the difficulty in fine-grained classification problems can be mitigated by the fact that it is possible to establish accurate correspondences between instances from a large family of related classes [8]. The study combines features that can be effectively located using generic feature models with breed specific models of part locations. They also created a publicly obtainable dataset for dog breed identification, integrated with a practical system that attains high accuracy in real-world images.

Another study proposed a deep convolutional neural network architecture (Inception) that attains the new state of the art capacity for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) [9]. Its result yields proof that approximating the anticipated optimal sparse structure by readily present dense building blocks is a practicable method for enhancing neural networks for computer vision. The main advantage of this method is a remarkable quality gain at a modest rise of computational requirements compared to shallower and narrower architectures.

B. Challenges taken into consideration

In dog breed identification, there are a few concerning obstacles namely inter class variance, intra class variance, pose variance. Inter class variance can be explained as distinct similarities in different breeds. Many breeds of canines have similar features like color, fur etc., which are hard to distinguish for an untrained eye. In order to attain high accuracy, a vision algorithm must be able to differentiate the minor details which are distinct to each class. Another issue of dog-breed identification is Intra class variance. An English cocker-spaniel generally is available in black, brown and a few other colors. An animal, though belonging to same breed, may have variable features which makes it hard to distinguish the breed accurately for the convolutional neural networks. Another major issue is the Pose variance. The canines' pictures may not be available in same pose every time. The CNNs depend on their input which is an image. Thus, the pose of the animal and the noise in the background may result in decrease in the accuracy. So, pose normalization can be an important factor to achieve the accurate result.

III. Problem Statement

The pet industry is huge and ever growing. Dogs are the most preferred pets. Currently, the only way to find the breed of a dog, is to inquire about it from either the owner of the dog or the professionals in the industry. This can be a time consuming and confusing experience for a prospective dog owner. Finding the breed of a dog should be as easy as snapping a picture. However, currently there is a lack of an application that addresses this market need. In this paper, we have built an app for the android platform that uses Convolutional Neural Networks (CNN) and Transfer learning to identify the breed of a dog by simply snapping a picture of it.

IV. Proposed Method

We propose an android application that gives its users the ability to find the breed of a dog by either clicking or uploading its picture on the app. The app makes a prediction using Convolutional Neural Networks and pre-trained feature extraction models with the help of Keras and Tensorflow libraries.

A. System Architecture

The android application has 4 major activities, as shown in Fig.1, namely, *Upload, Take Photo, View*, and *Instructions.*

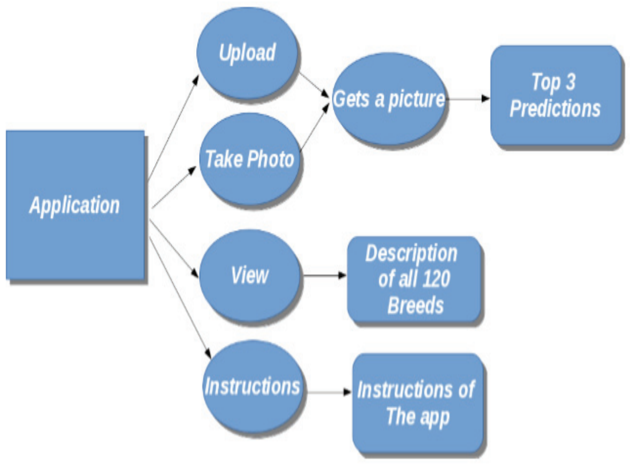

Figure 1. System Architecture

The *View* and *Instructions* activities provide the user with descriptions about different types of dog breeds and how to use this app to find the breed of a dog.

Using the *Upload* and *Take Photo* activities, the user can either upload or directly take a photo using the camera and upload it on the app. The app will then generate the top 3 predictions of the breed of the dog in the given image.

B. Dataset Used

To train the model, we have used Stanford dog's dataset. This dataset has 120 different dog breed images and each dog breed has minimum 60 images. We have divided dataset into train data, validation data and test data using stratified shuffle split where train data has 9199 images, validation data has 2000 images and test data has 9381 images.

C. Feature extraction from images

We have extracted the features of images using pre-trained models, which are trained on image-net and sent to a fully connected layer. We have used different pre-trained models such as Inception-v3, Inception-ResNet-v2, VGG16 and Xception to extract the features of the images. When Inception-v3, Inception-ResNet-v2, VGG16 and Xception are used for extraction of features from an image, accuracy percentages of 89, 94, 81, 93 are obtained respectively on the testing data.

D. Dog breed Identification on Android

To use a Keras model on an android application, we will first, run the Keras model on a server. We then convert the model into a Tensorflow protobuff (pb) file. The following steps are involved in converting a Keras model to a Tensorflow pb file:

1. Run the Keras model on the server
2. Convert the model to a Tensorflow pb file.
3. Save the latest checkpoint.
4. Freeze the graph.
5. Finally, optimize the saved model.

Steps 1 and 2 can be done using Tensorflow whereas steps 3, 4 and 5 are done using bazel. Bazel is a build and test tool. We have used it to freeze the graph and optimize the saved model which is in the format of Tensorflow protobuff.

To add TensorFlow dependencies to android, we compile *'org.tensorflow:tensorflow-android:+'* and load the protobuff file to assets folder. A Tensorflow inference interface has been created to send the uploaded image pixels to the network and get the predictions. Fig. 2 depicts how a prediction is obtained when an image is passed.

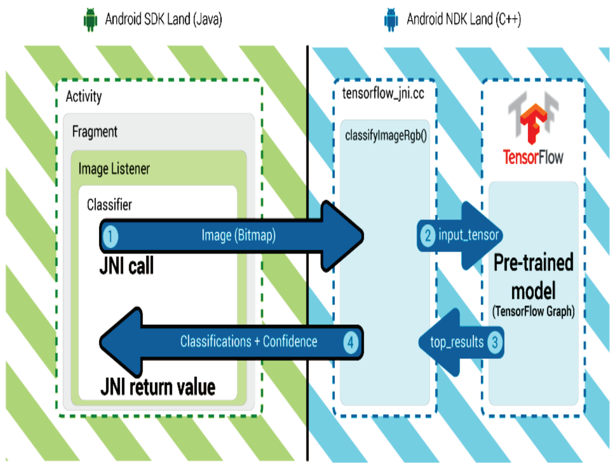

Figure 2. Android app calls from JAVA to TensorFlow

The architecture given in Fig. 2 comprises of two repositories, namely Android Standard Development Kit (android SDK) and Android Native Development Kit (android NDK.)

Android SDK is written in Java. It is designed for interfaces and useful for characteristics like activities, fragments, event listeners etc. On the other hand, android NDK is coded in C++ and is used for working with C++ files. NDK can be useful for working with Tensorflow models, as Tensorflow itself is written in C++.

Initially, the Bit-map input image is sent from SDK to NDK Tensorflow wrapper for Android (written in C++). It takes the given image and resizes it, i.e. to tensor and gives the obtained tensor to a pre-trained model (protobuff file), which is a pre-trained convolutional neural network. The model puts out a tensor (a C++ file) which is returned to the SDK.

E. Procedure

The following is the step wise procedure for deploying the model on android

1. Resize the image to (400, 400, 3).
2. Extract the features of original image from Inception-v3 (9199, 2048), Inception-ResNet-v2 (9199, 2048), Xception (9199, 1536) and

concatenate to form the input (9199, 5632) which will be sent to a fully connected network.

3. Fully connected layer configuration is given in Fig. 3.

Dense (2048)
Activation ('elu')
Dropout (0.5)
Dense (120)
Activation('softmax')

Figure 3. Fully connected layer configuration

4. Train the model with original images and store the model as model-1.
5. Train the model with flipped version of original images and store the model as model-2.
6. Predict the breed of the dog by averaging the results of model-1 and model-2.
7. Convert Keras model to Tensorflow pb file and deploy it on android.

V. Results and discussion

We have created a user-friendly android application that tells you the breed of dogs just by uploading an image of the dog.

Fig. 4 shows the user interface of the home screen of the android application. When the user clicks on the application, the user gets an interface showing the options: upload, Take a photo, view and instructions.

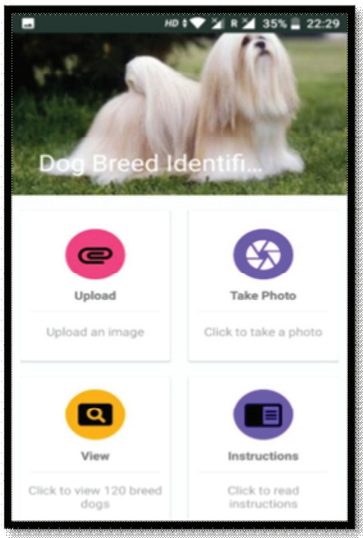

Figure 4. Home screen of the android application

When the user selects *upload*, the user will be provided with the image files on the device which user can upload to identify the breed of the dog. When *Take photo* is selected, the default camera application of the device is opened which enables the user to capture the image of the desired dog. As soon as user inputs the image (either through *Upload* or *Take photo*), the predictions of the breed and their corresponding rounded probabilities are given. As shown in Fig. 5, if the top-1 predicted class has probability above 90 percent, then only top-1 class will be displayed, else top-3 predicted classes will be displayed.

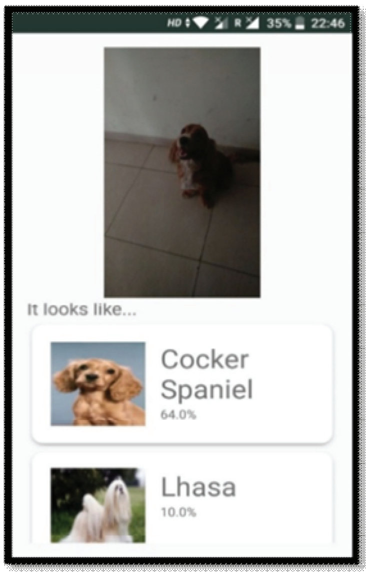

Figure 5. Prediction of uploaded image shown in recycler view

When the user clicks on *view*, 120 dog images of 120 different breeds are displayed; each image, when selected, gives the information regarding that corresponding breed. If the user selects *Instructions,* directions of uploading the image for getting better results are provided.

A. Accuracy

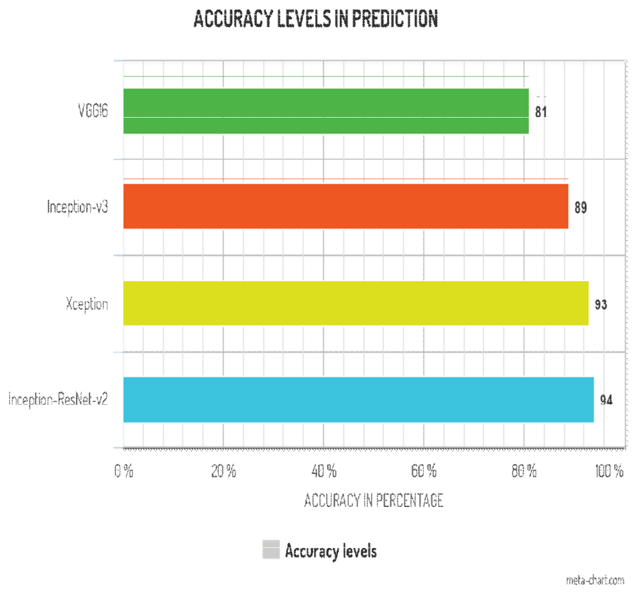

Figure 6. Accuracy levels in prediction of dog breeds with different image feature extraction tools.

Fig. 6 shows the different levels of accuracy achieved, when different pre-trained models for image extractions were used to test the model. Xception and Inception-ResNet-v2 showed the most promising results with an accuracy of 93% and 94% respectively. Inception-v3 and

VGG16 achieved accuracy levels of 89% and 81% respectively. Overall, results from all the four pre-trained models have very promising accuracy levels.

B. Drawbacks and Limitations

A drawback of the application is that it works efficiently with images captured directly with the camera, but it is not efficient with images captured indirectly i.e. image captured from another image. Another drawback is its size, the application takes more memory than that of a user's convenience.

VI. Conclusion and Future Work

The proposed work is designed, implemented and tested successfully. We have created a user-friendly android application that predicts the breed of a dog by uploading or clicking an image. It works without an internet connection and gives the result instantly, thereby resulting in negligible waiting time. In this paper, we have explained how to build a dog-breed identification model using pre-trained models and deploy it on an android device. In the future, improvements can be made for reduction in the size of the application. Also, enhancements can be made to the model which gives a correct prediction for indirectly captured images as well.

References

[1] Francois Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions", arXiv: 610.02357v3, Apr, 2017.
[2] Alex Krizhevsky, IlyaSutskever and GeoffreyE. Hinton, "ImageNet classification with deep convolutional neural networks", In Neural Information Processing Systems, pp.1106-1114, 2012.
[3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, ZbigniewWojna, Rethinking the Inception Architecture for Computer Vision, arXiv: 1512.00567v3
[4] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv: 1602.07261v2, Aug, 2016.
[5] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V.Le, "Learning Transferable Architectures for Scalable Image Recognition", arXiv: 1707.07012v3, Dec, 2017.
[6] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv: 1409.1556v6, Apr, 2015.
[7] Dylan Rhodes, "Automatic Dog Breed Identification".
[8] J. Liu, A. Kanazawa, D. W. Jacobs, and P. N. Belhumeur, "Dog breed classification using part localization", in Proc. European Conference on Computer Vision, 2012.
[9] C Szegedy, W. Liu, Y. Jia. P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions". CoRR abs/1409.4842, 2014.