Breed Identification On Various Classification Models

Vipul Jain[1], Dr. Bhoomi Gupta[2]

[1]*UG student, IT & Maharaja Agrasen Institute Of Technology,Dehi,India*

[2]*Asst. Prof. Department of IT & ,Maharaja Agrasen Institute Of Technology,Dehi,India*

---***---

Abstract - Dog breed classification and identification is one of the fine and interesting topic that makes to look upon and find fascinating. This project is based on the classification of the different class of breed in which the evaluation of each test image takes place and the probabilities of each test image finds the accurate breed of the dog. The use of transfer learning that is what you know in one domain and apply into another domain. Training of the model on the train dataset and then find the outcome and accuracy of the model on test dataset images. There are 120 breeds of dog on which the model made the prediction and 10000+ images for the test, train and validate .The model used in this is MOBILENET –V2 and 4 more model namely VGG-16 ,ResNet50,Inceptionv3,EfficientNet that take under consideration to obtain the accuracy of breed of dog with 94+ % accuracy .Tenserflow and Tenserflow hub is used to implement the model training and 100 epochs and batch size of 32 is used to achieve the accuracy.

Keywords—Dog breed classification,Tenserflow ,image Classification,Transfer Learning,.

1.INTRODUCTION

As considering this project motto, this idea can be benefited for all those people of our society that wanted to change life for all those pet animals that don't get proper hygiene, food, water and other things that are needed the most. Also there are several shelters and NGO that are willing to help these pets which do not have any place to live. By using these technology, one can choose their favourite pet according to their likings by classify that which breed is most suitable for them. Also there would be some awareness in between people that how they can treat their pet dogs such that in times of illness or sickness one can take proper care of them if the natural breed is known to them.

As there are several purpose for these dogs to train them and use them in military so that their abilities are used and organized in the country benefits. So the identification of a random breed and make the best possible use of that kind of gene.

In this project the usage of these models is considered to obtain the accuracy of classification of the breed detection and how the model takes its parameter to learn the properties of that image for the prediction process.

The pre trained models used in this paper are compared and the strength of each model is undertaken .

The rapid development in computer vision and image classification through machine learning with the help of a special tool named Transfer Learning that allows the pre-existing model that are trained on huge dataset for some previous tasks are used to train the images or dataset for the unknown to them.

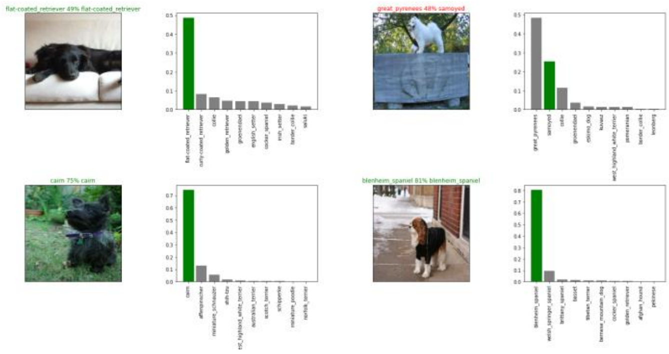

Fig 1. Few predictions of the trained model

2. Body of Paper

I. RELATED WORK

Project Name	**Authors**	**Date Of Publish**	**Methodology**	**Objective**	**Conclusion**
1. Classification of dog barks: a machine learning approach[3]	Csaba Molnár · Frédéric ,Kaplan · Pierre Roy · François ,Pachet · Péter Pongrácz · ,Antal Dóka · Ádám Miklósi	Received: 6 July 2006 / Revised: 23 November 2007 / Accepted: 13 December 2007	Subjects Barks of the Multi breed (a Hungarian sheepdog listed at the 238th Standard of the FC	By only hearing their barks, humans showed only modest accuracy in discriminating between individual dogs	Using new machine learning algorithms analyse the classification paramerets.
2. Face recognition based dog breed classification using coarse-to-fine concept and PCA[5]	Massinee Chanvichitkul, Pinit Kumhom, Kosin, Chamnongthai	Published in: 2007 Asia-Pacific Conference on Communications Date of Conference: 18-20 Oct. 2007 Date Added to IEEE *Xplore*: 21 January 2008	Principle of component is used	. This paper proposes a method to classify dog breed based on the dog face images	It shows the PCA classification method.
3. **Cats and dogs[6]**	Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, C. V. Jawahar	Published in: 2012 IEEE Conference on Computer Vision and Pattern Recognition Date of Conference: 16-21 June 2012 Date Added to IEEE *Xplore*: 26 July 2012	Two classification approaches: a hierarchical one breed, and a flat one, in which the breed is obtained directly.	The fine grained object categorization problem of determining the breed of animal from an image	The task of discriminating the 37 different breeds of pets,

4. Kernelized Structural Classification for 3D Dogs Body Parts Detection	Simone Pistocchi, Simone Calderera, Shanis Barnard, Nicola Ferri, Rita Cucchiara	Date of Conference: 24-28 Aug. 2014 Date Added to IEEE *Xplore*: 08 December 2014	SSVM using 3D and 2D features and parameter of model.	independently from the size and breed ,realize on a kernelized discriminative structural classificator specifically tailored for dogs .	Promising results have emerged during the experimental evaluation carried out at a dog shelter
5. Dog Breed Classification Via landmarks[4]	Xiaolong Wang, Vincent Ly, Scott Sorensen, Chandra Kambhamettu	Date of Conference: 27-30 Oct. 2014 Date Added to IEEE *Xplore*: 29 January 2015	Model the facial geometry of dog breeds based on 2-D landmarks.	geometry, one can obtain significant performance improvements in dog breeds categorization	The Grassmann manifold is applied to describe the geometry of a given breed structure.
6. Comprehensive Study of Multiple CNNs Fusion for Fine-Grained Dog Breed Categorization[7]	Minori Uno, Xian-Hua Han, Yen-Wei Chen	Date of Conference: 10-12 Dec. 2018 Date Added to IEEE *Xplore*: 07 January 2019	Explore transfer Learning algorithms for CNN fusion.	Fusion of multiple CNN architecture and increase the performance of the model.	Study of the fusion of different layers with fusion architecture.
7. An Efficient Framework for Animal Breeds Classification Using Semi-Supervised Learning and Multi-Part Convolutional Neural Network (MP-CNN)[8]	S. Divya Meena, L. Agilandeeswari	Date of Publication: 21 October 2019	Fine-grained classification where we utilize the Multi-Part Convolutional Neural Network (MP-CNN).	The influence of the number of images and their role in testing accuracy.	For a fine-grained animal breed classification,this experiment utilize the MP-CNN,

II. DATASET AND PROCESSING

Problem: Problem statement is that to identify the breed of dog by giving a random image.

Data: The data used here is taken from various sources and the raw data is also used from scratch and also from google images.

The labels.csv file that contain the labels of all the dog breeds is the source used from the internet by searching about the breeds all over the world.

Evaluation: The evaluation is the file with prediction probabilities of each dog breed of each test image.

Features: some info about the data:

- We are dealing with images (unstructured data) so it is probably best use of deep learning/transfer learning.
- There are 120 breeds of dogs (this means there are 120 different classes)

There are around 10000+ images in test(with no labels, because we'll want to predict them)and training(with labels)sets.

We prepare data by applying feature-wise and samplewise standardization on raw images. Further, we apply data augmentation methods such as random rotation, horizontal and vertical shifts, shears, zooms, flips and filling points outside boundaries by nearest points in image. We use rotation range of 40 degrees while horizontal, vertical, shear and zoom ranges are all equal to 0.2 to generate augmented images. We also generate images by flipping them horizontally. We fill the points outside the boundaries according to the nearest mode. We then fine-tune the pre-trained Inception model from ImageNet using this augmented data. Data augmentation helps in increasing the training data size and hence we can aim for a better validation accuracy.

III. METHODOLOGY

This kind of problem is called multi-class image classification. It's multi-class because we're trying to classify mutliple different breeds of dog. If we were only trying to classify dogs versus cats, it would be called binary classification (one thing versus another). Multi-class image classification is an important problem because it's the same kind of technology Tesla uses in their self-driving cars or Airbnb uses in automatically adding information to their listings.

There are the most important pre trained model that are used for image classification :

- VGG-16:Very Deep Convolution Network to beat even today,developed at visual graphics group in 2014 at the university of oxford.

  The layers of the model:

  - Convolutional Layers = 13
  - Pooling Layers = 5
  - Dense Layers = 3
  - number of parameters is **138 Billion**
- ResNet50: Residual Network, main reason behind this model was to increase accuracy as the model went on to become deeper
  - ResNet50 is among the most popular that achieved a top error rate of 5%.
- Inceptionv3: A bit smaller than VGG with lower error rate,but module just performs convolutions with different filter sizes on the input, performs Max Pooling.
  - number of layers in Inceptionv1 is 22
  - reduced the error rate to only 4.2%.
- EfficientNet: Comprised the new scaling method called compound Scaling,got fixed amount of layers and time which achieve better performance.

A. Accessing the data

The data files we're working with are available on our Google Drive, we can start to check it out.
Let's start with `labels.csv` which contains all of the image ID's and their assosciated dog breed (our data and labels)

B. Getting images and their labels

Since we've got the image ID's and their labels in a DataFrame (`labels_csv`), we'll use it to create:

- A list a filepaths to training images
- An array of all labels
- An array of all unique labels

We'll only create a list of filepaths to images rather than importing them all to begin with. This is because working with filepaths (strings) is much efficient than working with images.
If it all worked, we should have the same amount of images and labels.
Finally, since a machine learning model can't take strings as input (what `labels` currently is), we'll have to convert our labels to numbers.To begin with, we'll find all of the unique dog breed names.
Then we'll go through the list of `labels` and compare them to unique breeds and create a list of booleans indicating which one is the real label (`True`) and which ones aren't (`False`)

C. Creating our own validation set

Since the dataset from Kaggle doesn't come with a validation set (a split of the data we can test our model on before making final predicitons on the test set), let's make one.We could use Scikit-Learn's train_test_split function or we could simply make manual splits of the data.
For accessibility later, let's save our filenames variable to `X` (data) and our labels to `y`.
Since we're working with 10,000+ images, it's a good idea to work with a portion of them to make sure things are working before training on them all.This is because computing with 10,000+ images could take a fairly long time. And our goal when working through machine learning projects is to reduce the time between experiments.

D. Preprocessing images (turning images into Tensors)

Our labels are in numeric format but our images are still just file paths.Since we're using TensorFlow, our data has to be in the form of Tensors.A Tensor is a way to represent information in numbers. If you're familar with NumPy arrays (you should be), a Tensor can be thought of as a combination of NumPy arrays, except with the special ability to be used on a GPU. Because of how TensorFlow stores information (in Tensors), it allows machine learning and deep learning models to be run on GPUs (generally faster at numerical computing).

To preprocess our images into Tensors we're going to write a function which does a few things:

1. Takes an image filename as input.
2. Uses TensorFlow to read the file and save it to a variable, image.
3. Turn our image (a jpeg file) into Tensors.
4. Resize the image to be of shape (224, 224).
5. Return the modified image.

It might be wondering why (224, 224), which is (heigh, width). It's because this is the size of input our model (we'll see this soon) takes, an image which is (224, 224, 3).

E. Creating data batches

Now we've got a function to convert our images into Tensors, we'll now build one to turn our data into batches (more specifically, a TensorFlow `BatchDataset`).

What's a batch? A batch (also called mini-batch) is a small portion of your data, say 32 (32 is generally the default batch size) images and their labels. In deep learning, instead of finding patterns in an entire dataset at the same time, you often find them one batch at a time.Let's say you're dealing with 10,000+ images (which we are). Together, these files may take up more memory than your GPU has. Trying to compute on them all would result in an error.Instead, it's more efficient to create smaller batches of your data and compute on one batch at a time.
TensorFlow is very efficient when your data is in batches of (image, label) Tensors. So we'll build a function to do create those first. We'll take advantage of of `process_image` function at the same time.

F. Building the model

We use `model.build()` whenever we're using a layer from TensorFlow Hub to tell our model what input shape it can expect.In this case, the input shape is `[None, IMG_SIZE, IMG_SIZE, 3]` or `[None, 224, 224, 3]` or `[batch_size, img_height, img_width, color_channels]`.Batch size is left as `None` as this is inferred from the data we pass the model. In our case, it'll be 32 since that's what we've set up our data batches

G. Saving and reloading a model

After training a model, it's a good idea to save it. Saving it means you can share it with colleagues, put it in an application and more importantly, won't have to go through the potentially expensive step of retraining it.

H. Training a model (on the full data)

After completion of the training of the subset of data on the model ,the whole data of 120 breeds i.e 10222 images have to be trained and the accuracy is to be found as best as possible by hyper tuning of the model.

I. Making predictions on custom images

It's great being able to make predictions on a test dataset .To do so, we'll:

- Get the filepaths of our own images.
- Turn the filepaths into data batches using create_data_batches(). And since our custom images won't have labels, we set the test_data parameter to True.
- Pass the custom image data batch to our model's predict() method.
- Convert the prediction output probabilities to prediction labels.
- Compare the predicted labels to the custom images.

IV. FINAL RESULT

The model that is trained on the 10 k + images is to be tested on the custom images of the dog to predict the actual accuracy of the model and the project that define the classification of the dog breeds.

The below snippet state the training of the model on full dataset with the best accuracy after the hyper-parameter tuning.

```
Epoch 1/10
100/100 [==============================] - 57s 566ms/step - loss: 0.3263 - acc: 0.9115 - val_loss: 1.0140 - val_acc: 0.6790
Epoch 2/10
100/100 [==============================] - 57s 569ms/step - loss: 0.2455 - acc: 0.9280 - val_loss: 0.9344 - val_acc: 0.7670
Epoch 3/10
100/100 [==============================] - 56s 562ms/step - loss: 0.2805 - acc: 0.9255 - val_loss: 0.4332 - val_acc: 0.8910
Epoch 4/10
100/100 [==============================] - 56s 564ms/step - loss: 0.2556 - acc: 0.9280 - val_loss: 0.2783 - val_acc: 0.9330
Epoch 5/10
100/100 [==============================] - 56s 564ms/step - loss: 0.2200 - acc: 0.9335 - val_loss: 0.1693 - val_acc: 0.9530
Epoch 6/10
100/100 [==============================] - 56s 564ms/step - loss: 0.2522 - acc: 0.9335 - val_loss: 0.1427 - val_acc: 0.9640
Epoch 7/10
100/100 [==============================] - 56s 563ms/step - loss: 0.2287 - acc: 0.9400 - val_loss: 0.1284 - val_acc: 0.9640
Epoch 8/10
100/100 [==============================] - 56s 564ms/step - loss: 0.2013 - acc: 0.9400 - val_loss: 0.1183 - val_acc: 0.9700
Epoch 9/10
100/100 [==============================] - 56s 562ms/step - loss: 0.2253 - acc: 0.9370 - val_loss: 0.1147 - val_acc: 0.9700
Epoch 10/10
100/100 [==============================] - 56s 563ms/step - loss: 0.2056 - acc: 0.9440 - val_loss: 0.1418 - val_acc: 0.9640
```

Fig .3 Training of data on the model ResNet50

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 06 | June - 2021 ISSN: 2582-3930

```
Epoch 1/10
100/100 [==============================] - 31s 309ms/step - loss: 0.6972 - acc: 0.6905 - val_loss: 0.3897 - val_acc: 0.8130
Epoch 2/10
100/100 [==============================] - 31s 309ms/step - loss: 0.4857 - acc: 0.7760 - val_loss: 0.2598 - val_acc: 0.8920
Epoch 3/10
100/100 [==============================] - 31s 313ms/step - loss: 0.4189 - acc: 0.8090 - val_loss: 0.2257 - val_acc: 0.9100
Epoch 4/10
100/100 [==============================] - 31s 308ms/step - loss: 0.4042 - acc: 0.8200 - val_loss: 0.2459 - val_acc: 0.8940
Epoch 5/10
100/100 [==============================] - 31s 309ms/step - loss: 0.3738 - acc: 0.8335 - val_loss: 0.2105 - val_acc: 0.9210
Epoch 6/10
100/100 [==============================] - 31s 309ms/step - loss: 0.3740 - acc: 0.8375 - val_loss: 0.2304 - val_acc: 0.9090
Epoch 7/10
100/100 [==============================] - 31s 309ms/step - loss: 0.3483 - acc: 0.8470 - val_loss: 0.2114 - val_acc: 0.9240
Epoch 8/10
100/100 [==============================] - 31s 309ms/step - loss: 0.3518 - acc: 0.8490 - val_loss: 0.2306 - val_acc: 0.9080
Epoch 9/10
100/100 [==============================] - 31s 309ms/step - loss: 0.3528 - acc: 0.8515 - val_loss: 0.2361 - val_acc: 0.9000
Epoch 10/10
100/100 [==============================] - 31s 309ms/step - loss: 0.3339 - acc: 0.8595 - val_loss: 0.1837 - val_acc: 0.9320
```

Fig .4 Training of data on the model VGG-16

```
Epoch 1/10
100/100 [==============================] - 8s 84ms/step - loss: 0.2508 - acc: 0.9640 - val_loss: 0.1576 - val_acc: 0.9820
Epoch 2/10
100/100 [==============================] - 8s 84ms/step - loss: 0.3697 - acc: 0.9460 - val_loss: 0.0740 - val_acc: 0.9860
Epoch 3/10
100/100 [==============================] - 8s 84ms/step - loss: 0.2880 - acc: 0.9550 - val_loss: 0.1545 - val_acc: 0.9720
Epoch 4/10
100/100 [==============================] - 8s 84ms/step - loss: 0.3564 - acc: 0.9560 - val_loss: 0.2403 - val_acc: 0.9680
Epoch 5/10
100/100 [==============================] - 8s 84ms/step - loss: 0.3452 - acc: 0.9580 - val_loss: 0.2760 - val_acc: 0.9600
Epoch 6/10
100/100 [==============================] - 8s 83ms/step - loss: 0.3198 - acc: 0.9620 - val_loss: 0.1099 - val_acc: 0.9840
Epoch 7/10
100/100 [==============================] - 8s 84ms/step - loss: 0.2845 - acc: 0.9510 - val_loss: 0.3676 - val_acc: 0.9460
Epoch 8/10
100/100 [==============================] - 8s 83ms/step - loss: 0.3051 - acc: 0.9460 - val_loss: 0.4805 - val_acc: 0.9500
Epoch 9/10
100/100 [==============================] - 8s 83ms/step - loss: 0.2433 - acc: 0.9660 - val_loss: 0.5949 - val_acc: 0.9060
Epoch 10/10
100/100 [==============================] - 8s 84ms/step - loss: 0.3141 - acc: 0.9550 - val_loss: 0.1923 - val_acc: 0.9620
```

Fig .5 Training of data on the model InceptionV3

```
Epoch 1/10
100/100 [==============================] - 32s 324ms/step - loss: 0.3151 - accuracy: 0.9055 - val_loss: 0.0549 - val_accuracy: 0.9670
Epoch 2/10
100/100 [==============================] - 30s 304ms/step - loss: 0.3398 - accuracy: 0.9160 - val_loss: 0.0211 - val_accuracy: 0.9670
Epoch 3/10
100/100 [==============================] - 30s 303ms/step - loss: 0.4132 - accuracy: 0.9070 - val_loss: 0.0252 - val_accuracy: 0.9780
Epoch 4/10
100/100 [==============================] - 30s 303ms/step - loss: 0.3304 - accuracy: 0.9170 - val_loss: 0.5167 - val_accuracy: 0.9810
Epoch 5/10
100/100 [==============================] - 31s 309ms/step - loss: 0.3719 - accuracy: 0.9195 - val_loss: 0.1717 - val_accuracy: 0.9780
Epoch 6/10
100/100 [==============================] - 30s 299ms/step - loss: 0.3617 - accuracy: 0.9145 - val_loss: 3.9468e-06 - val_accuracy: 0.9760
Epoch 7/10
100/100 [==============================] - 31s 307ms/step - loss: 0.3225 - accuracy: 0.9245 - val_loss: 0.0853 - val_accuracy: 0.9710
Epoch 8/10
100/100 [==============================] - 30s 299ms/step - loss: 0.3143 - accuracy: 0.9265 - val_loss: 1.0649e-05 - val_accuracy: 0.9780
Epoch 9/10
100/100 [==============================] - 30s 297ms/step - loss: 0.3283 - accuracy: 0.9225 - val_loss: 0.3379 - val_accuracy: 0.9570
Epoch 10/10
100/100 [==============================] - 30s 298ms/step - loss: 0.3275 - accuracy: 0.9220 - val_loss: 0.4117 - val_accuracy: 0.9790
```

Fig .6 Training of data on the model EfficientNet

Testing of the model on the custom images that are picked randomly and finding the results.

```
# Check custom image predictions
plt.figure(figsize=(10,10))
for i,image in enumerate(custom_images):
  plt.subplot(1,4,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.title(custom_pred_labels[i])
  plt.imshow(image)
```

Fig 7 Prediction of the dog images

The algorithm has an accuracy of 83.7% by using Tenserflow(using transfer learning). At first the model trains on 100 epochs with a batch size of 32 and then it is done with 15 epochs to obtain this accuracy.

Model	Year	Number of Parameters	Top-1 Accuracy
VGG-16	2014	138 Million	74.5%
ResNet-50	2015	25 Million	77.15%
Inception V3	2015	24 Million	78.8%
EfficientNetB0	2019	5.3 Million	76.3%
EfficientNetB7	2019	66 Million	84.4%

3. CONCLUSIONS

This project provides an overview of the TensorFlow and TensorFlow hub that how these model works. The deep knowledge of Data Augmentation is very important to manipulate the images or distort them, to create even more training data for the model to learn from. Fine tuning of the model predictions and find the patterns inside the test dataset and applied it to our own model. Also it is important that higher the number of dataset or images to be trained on, more will be the accurate and precise results until the overfitting reached.

As considering this project motto, this idea can be benefited for all those people of our society that wanted to change life for all those pet animals that don't get proper hygiene, food, water and other things that are needed the most. Also there are several shelters and NGO that are willing to help these pets which do not have any place to live. By using these technology, one can choose their favourite pet according to their likings by classify that which breed is most suitable for them. Also there would be some awareness in between people that how they can treat their pet dogs such that in times of illness or sickness one can take proper care of them if the natural breed is known to them.

The main objective of this project is to find out the breed of the dog for the consumer or the user so that it is easier to choose which kind of dog is suitable for the user as a good companion is the best friend. As someone sees a dog and wants to adopt one but does not know which kind of pup was that. This system is very well versed in this scenario.For future references the objective is to find the facial expression of the dog, the nature and behavior.

REFERENCES

1 [1] J. Breckling, Ed., The Analysis of Directional Time Series: Applications to Wind Speed and Direction, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[2] towardsdatascience.com/review-mobilenetv2

[3] Classifcation of dog barks: a machine learning approach Csaba Molnár · Frédéric Kaplan · Pierre Roy · François Pachet · Péter Pongrácz · Antal Dóka · Ádám Miklós.

[4] DOG BREED CLASSIFICATION VIA LANDMARKS Xiaolong Wang, Vincent Ly, Scott Sorensen and Chandra Kambhamettu

[5] Face recognition based dog breed classification using coarse-to-fine concept and PCA Massinee ChanvichitkulDepartment of Electronics and Telecommunication Engineering, Faculty of Engineering, King Mongkut¿s University of Technology Thonburi (KMUTT), Pracha-uthit Road, Tung-kru, Bangkok 10140, Thailand

[6] Cats and dogs Omkar M Parkhi Department of Engineering Science, University of Oxford, United Kingdom Andrea Vedaldi

[7] Comprehensive Study of Multiple CNNs Fusion for Fine-Grained Dog Breed Categorization Minori Uno Fac. of Sci., Yamaguchi Univ., Yamaguchi, JapanXian-Hua HanGrad. Sch. of Sci. & Technol. for Innovation, Yamaguchi Univ., Yamaguchi, Japan Yen-Wei Chen

[8] An Efficient Framework for Animal Breeds Classification Using Semi-Supervised Learning and Multi-Part Convolutional Neural Network (MP-CNN)

[9] Tejeswar Sadanandan et al, International Journal of Computer Science and Mobile Computing, Vol.9 Issue.4, April- 2020, pg. 76-82