ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 12, December 2016

Transfer Learning for Image Classification of various dog breeds

Pratik Devikar

[1]

***Abstract*— Dog Breed Categorisation is a very specific application of Convolutional Neural Networks. The classification of Images by Convolutional Neural Network has proven to be highly efficient, but it has some drawbacks. This methodology requires a significant amount of images as training data and substantial time for training and achieving higher accuracy on the classification. Transfer Learning can be used to overcome this problem. Through Transfer learning, a pre-trained model can be fine-tuned to perform classification on image datasets that may be outside the domain of the pre-trained model. Due to limitation of image datasets available for non-popular dog breeds, we propose to use transfer learning to perform Image Classification of 11 different non-popular dogs.**

We have used Google's Inception-v3 model trained on 100,000 images covering 1000 different categories. These categories of images originally do not include dog breeds like Belgian Malinois, Cavalier King Charles Spaniel, Havanese and 8 more types of breed. We retrained the Inception model to classify the dog images, using the Tensorflow Library and achieved an overall accuracy of 96% on the images taken from Google..

***Index Terms*—Image Classification, Transfer Learning, Convolutional Neural Network, Inception-v3, Dog Breed.**

I. Introduction

Since the introduction of Deep Learning, there have been rapid advances in the field of Image Classification. Convolutional Neural Networks (CNN) have been used to a great effect in applications such as object classification, scene recognition, and other applications mainly due to its high accuracy. [5] Many machine learning methods work well only under a common assumption: the training and test data are drawn from the same feature space and the same distribution. When the distribution changes, most statistical models need to be rebuilt from scratch using newly collected training data. In many real world applications, it is expensive or impossible to re-collect the needed training data and rebuild the models. It would be nice to reduce the need and effort to re-collect the training data. In such cases, knowledge transfer or transfer learning between task domains would be desirable. In this paper, we demonstrate the use of Transfer Learning to classify images of 11 different non-popular dog breed [1]. We used Google's pre-trained CNN model known as Inception-v3 on the ImageNet dataset comprising of 100,000 images of about 1,000 classes. Transfer learning refers to the process of applying learning from a previous training session to a new training session. The original Inception model is trained by feeding an image to the input, and at each layer of the model it will perform some computations on the data until it outputs a label and classification accuracy. Transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task. In our case we retrained that last layer on the features of various dog breeds so that we can add their representations to the Inception model's repository of knowledge.

The paper firstly explains the underlying concepts in Convolutional Neural Network and how it carries out the task of image classification. Secondly the Inception model is described. Then the dataset used for the research is elaborated and the process of Image Classification is illustrated. Finally, the methodology to implement Transfer learning to fine-tune the Inception-Model and retrain the last layers of the CNN model to classify the dog image data is described.

II. Previous works and background

Deep Learning-powered image recognition is now performing better than human vision on many tasks thanks to Convolutional Neural Networks. At present, one of the best learning models known as Inception-v4 has achieved a 3.08% top error rate on the ImageNet dataset. Inception-v4 model has 75 trainable layers. It will take around a couple of weeks to train such an immense and a complex model, coupled with substantial computational requirements.

The great variety in dog breed poses a significant problem to those who would be interested in acquiring a new canine companion.. Walking down the street or sitting in a coffee shop, one might see a friendly, attractive dog and wonder at its pedigree. In many situations, it is impossible to ask an owner about the breed, and in many cases, the owner themselves will be either unsure or incorrect in their assessment. Unless the dog falls into one of a few very widely known and distinctive breeds such as the golden retriever, Siberian husky, or German Shepherd to name a few, it might prove difficult to identify one's ideal companion without a great deal of research or experience[8]. The data corresponding to non-popular dog breed, however, is not available in larger scales. This hinders the implementation of convolutional neural networks, due to its requirements of large amount of image data for training. Transfer learning thus becomes a workable option to tackle the issue of dog image classification, where the generalized features from a pre-trained model can be used to test classification on related datasets.

[1]***Manuscript received Dec, 2016.***

***Pratik Devikar**, Computer Science, Shri Ramdeobaba College of Engineering and Management, Nagpur, India, +919096098612.*

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 12, December 2016

III. Problem Setup

A. *Convolutional Neural Networks*

Convolutional neural networks (CNNs) consist of multiple layers of receptive fields. These are small neuron collections which process portions of the input image. The outputs of these collections are then tiled so that their input regions overlap, to obtain a better representation of the original image; this is repeated for every such layer. One major advantage of convolutional networks is the use of shared weight in convolutional layers, which means that the same filter (weights bank) is used for each pixel in the layer; this both reduces memory footprint and improves performance. Different layers in CNN include Convolutional , Rectified Linear Unit, Pooling and Fully Connected layers.

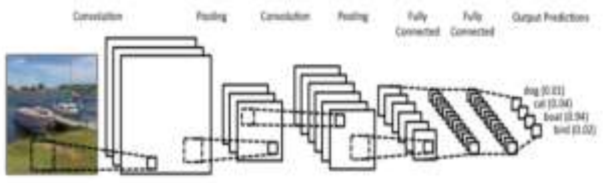

Fig 1: Convolutional Neural Network Architecture

Observable from Figure 1, A *convolutional neural network* consists of model layers that apply local filters and are stacked in a certain order.

Convolution: The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Filters act as Feature detectors. The value of the filter, is in fact, not manually provided but the machine chooses the suitable value by training and changing its weights.

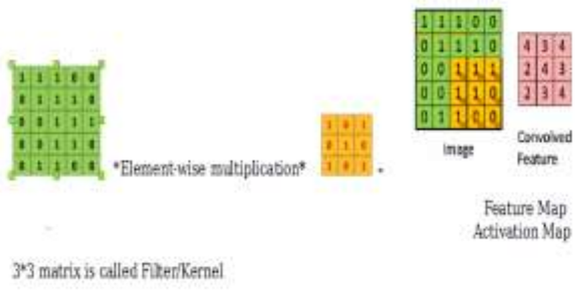
Element-wise multiplication
Image
Convolved Feature
Feature Map
Activation Map
3*3 matrix is called Filter/Kernel

Fig 2: Convolution

Non-Linear (Rectified Linear Unit(ReLU)): ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU). Ex: Output = Max (zero, input)

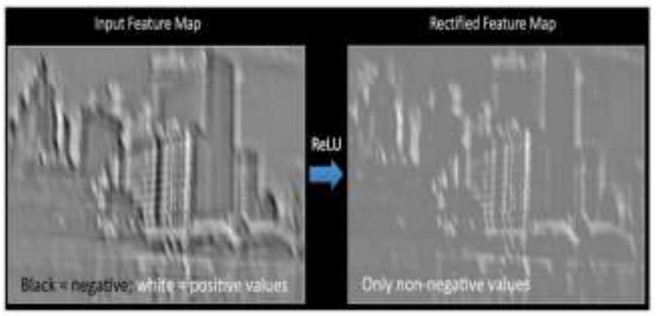

Fig 3: Rectified Linear Unit

Pooling or Sub-Sampling: Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc

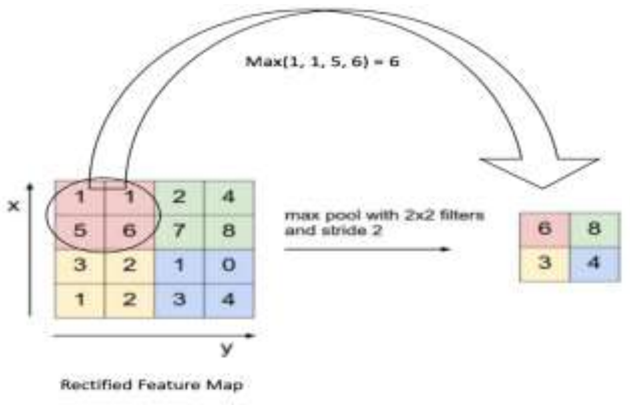

Fig 4: Pooling

Fully-Connected: Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. A fully connected layer takes all neurons in the previous layer (be it fully connected, pooling, or convolutional) and connects it to every single neuron it has.

B. *Inception-v3 model*

In this paper,we are using a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. This model consists of a network of 22 layers. The exact structure of the extra network on the side, including the auxiliary classifier, is as follows[18]:
The exact structure of the extra network on the side, including the auxiliary classifier, is as follows:

- An average pooling layer with 5×5 filter size and stride 3, resulting in an 4×4×512 output for the (4a), and 4×4×528 for the (4d) stage.
- A 1×1 convolution with 128 filters for dimension reduction and rectified linear activation.
- A fully connected layer with 1024 units and rectified linear activation.
- A dropout layer with 70% ratio of dropped outputs.
- A linear layer with softmax loss as the classifier (predicting the same 1000 classes as the main classifier, but removed at inference time)

A schematic view of the resulting network is depicted in Figure 6.

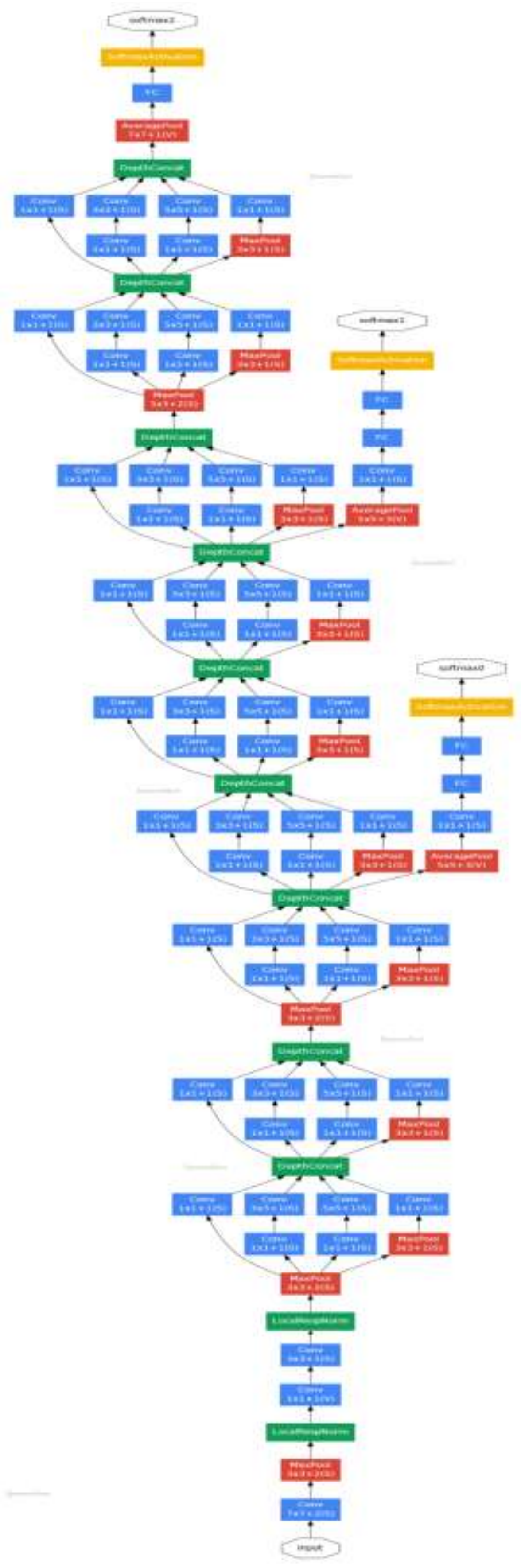

Fig 6: GoogLeNet Network

C. Combining Transfer Learning and Convolutional Neural Network

The low-level and high-level features learned by a CNN on a source domain can often be transferred to augment learning in an alternate but related target domain. For target problems with abundant data, we can transfer low-level features, such as edges and corners, and learn new high-level features specific to the target problem. To extract the features, we retrieve the next-to-last layer of the Inception-v3 as a feature vector for each image. Indeed, the last layer of the convolutional neural network corresponds to the classification step: as it has been trained for the ImageNet dataset, the categories that it will be output will not correspond to the categories in the Product Image Classification dataset we are interested in. The output of the next-to-last layer, however, corresponds to features that are used for the classification in Inception-v3. The hypothesis here is that these features can be useful for training another classification model, so we extract the output of this layer.Nonetheless, if the source and target domain are adequately similar, the feature representation learned by the CNN on the source task can likewise be utilized for the target problem. Deep features extracted from CNNs trained on large annotated datasets of images have been used as generic features viably for an extensive variety of computer vision tasks.

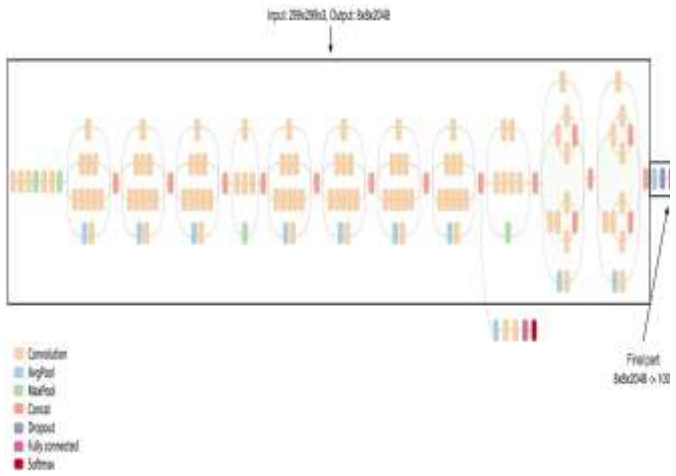

Fig 5: Inception-v3 model

IV. Dog Breed Dataset

We used 11 image datasets: one dataset for each type of breed. One dataset consists of 25 slightly different images of the same type of dog. All these images were of the resolution 100x100 pixels and were taken directly from Google Images. Some of the training data is shown below:

Fig 7: Australian Shepherd

Fig 8: Belgian Malinois

Fig 9: Bichon Frise

Fig 10: Bulldog

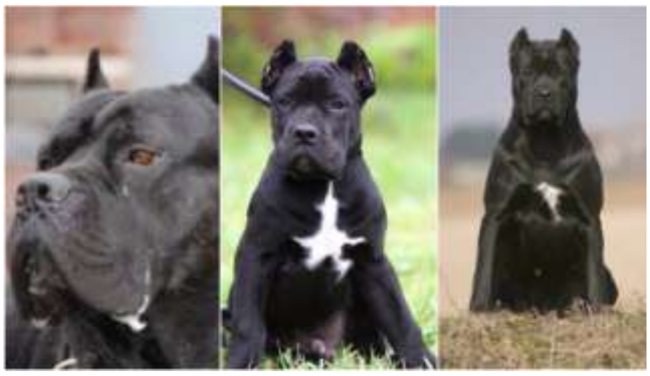
Fig 11: Cane Corso

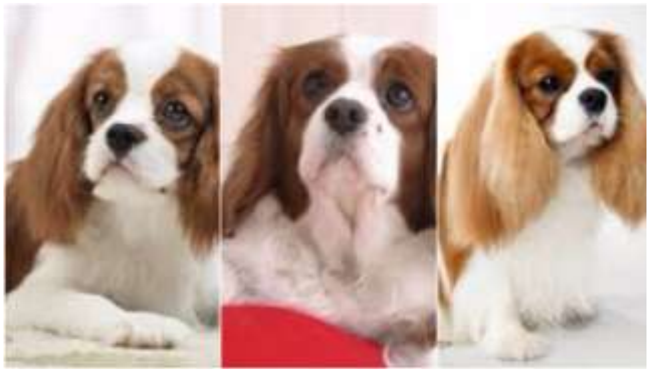
Fig 12: Cavalier King Charles Spaniel

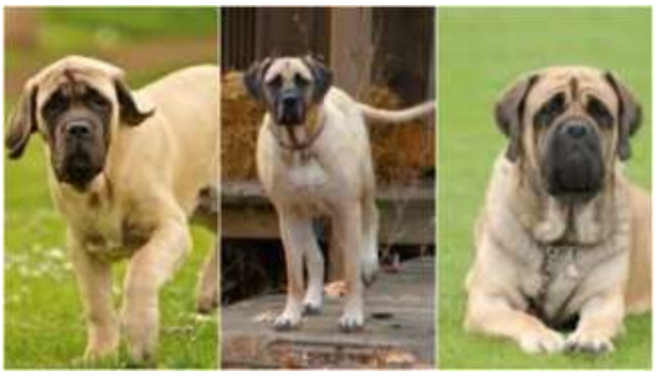
Fig 13: English Mastiff

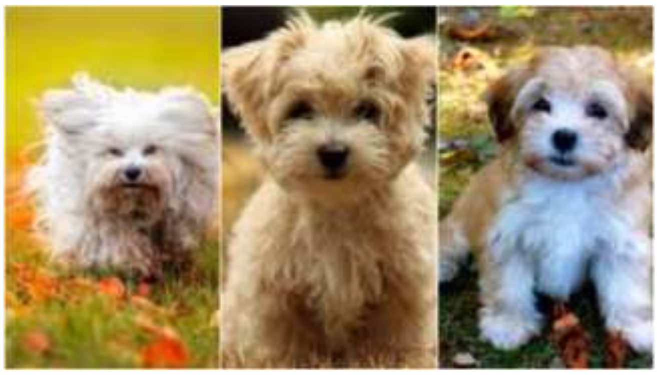
Fig 14: Havanese

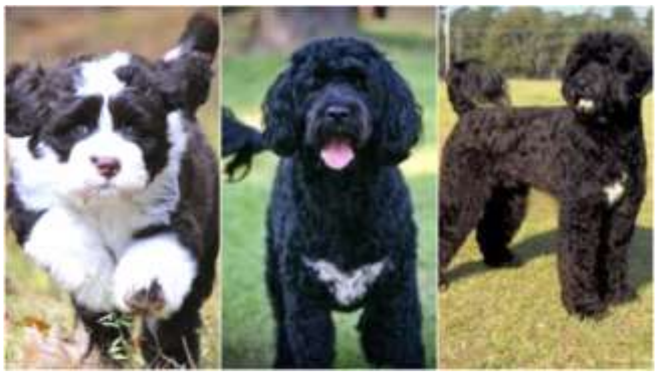
Fig 15: Portuguese Water Dog

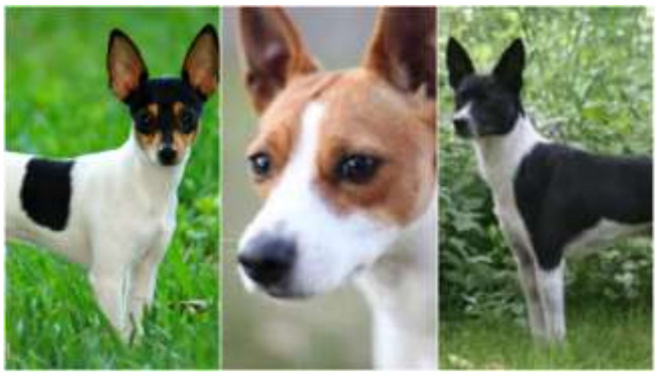
Fig 16: Rat Terrier

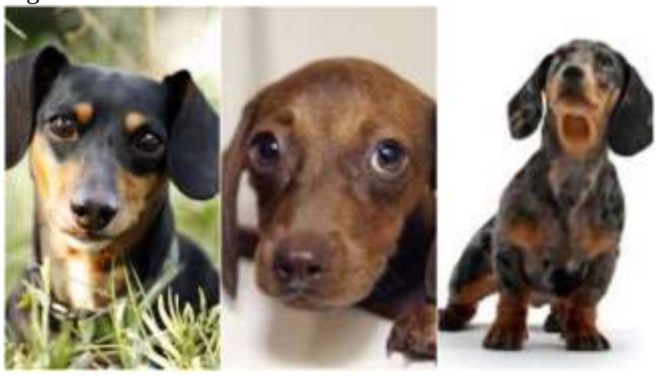
Fig 17: Daschund

The Inception model that accomplished the cutting edge in object detection and image classification was trained on 100,000 images; while the data what we had for dog breed classification was only in the triple figures. Hence fine tuning the CNN model and retraining it in the dog dataset was required.

V. Transfer Learning and Fine-Tuning Convolutional Neural Networks

In practice, we don't usually train an entire Deep Convolutional Neural Network from scratch with random initialization. This is on the grounds that it is relatively rare to have a dataset of abundant size that is required for the depth of network required. Rather, it is common to pre-train a DCNN on a very large dataset and then use the trained DCNN weights either as an initialization or a fixed feature extractor for the task of interest.

Fine-Tuning: Transfer learning methodologies rely upon different elements; however the two most essential ones are the size of the new dataset and its similarity to the original dataset. Knowing that DCNN features are more generic in early layers and more dataset-specific in later layers, there are four major scenarios:

1. New dataset is smaller in size and comparatively similar in content to the original dataset: If the data is small, overfitting concerns reduce the effectiveness of fine-tuning the DCNN. Since the data is similar to the original data, we expect higher-level features in the DCNN to be relevant to this dataset as well. Hence, the best idea would be to train a linear classifier on the CNN-features.
2. New dataset is relatively large in size and similar in content compared to the original dataset: Since we have more data, we can have more confidence that we would not over fit, if we were to try to fine-tune through the full network.
3. New dataset is smaller in size but very different in content compared to the original dataset: Since the data is small, it is likely best to only train a linear classifier. Since the dataset is altogether different, training the classifier from the top of the network, which contains more dataset-specific features, would not necessarily efficient. Rather, training a classifier from activations somewhere prior in the network would work better.
4. New dataset is relatively large in size and contrasted in content compared to the original dataset: Since the dataset is very large, we may expect that we can afford to train a DCNN from scratch. However, in practice it is very often still beneficial to initialize with weights from a pre-trained model. In this case, we would have enough data and confidence to fine-tune through the entire network.

Fine-tuning DCNNs: For this Dog Breed Classification problem, we fall under scenario 3. We fine-tuned higher-level layers of the network. This is propelled by the perception that the prior features of a DCNN contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the DCNN becomes progressively more specific to the subtle elements of the classes contained in the dog dataset.

Transfer learning constraints: As we decided to use a pre-trained network, slight constraints in terms of the model architecture needed to be handled. For example, we couldn't arbitrarily take out convolutional layers from the pre-trained network. However, due to parameter sharing, we could without much of a stretch, run a pre-trained network on images of different spatial size. This is obviously apparent in the case of Convolutional and Pool layers because their forward function is independent of the input volume spatial size. In case of Fully Connected (FC) layers, this still remains constant in light of the fact that FC layers can be changed over to a Convolutional Layer.

Learning rates: We used a smaller learning rate for DCNN weights that were being fine-tuned under the assumption that the pre-trained DCNN weights are effective. In order to keep the learning rate and learning rate decay small, we didn't distort the weights too quickly or too much.

Data Augmentation: One of the drawbacks of non-regularized neural networks is that they are extremely flexible: they learn both features and noise equally well, increasing the potential for overfitting. In our model, we applied L2 regularization to avoid overfitting. But even after that, we observed a large gap in model performance on the training and validation dog images, indicating that the fine tuning process is overfitting to the training set. To combat this overfitting, we leveraged data augmentation for the dog image dataset.

There are many ways to do data augmentation, such as the popular horizontally flipping, random crops and color jittering. As the color information in these images is very important, we only rotated the images at different angles – at 0, 90, 180, and 270 degrees.

Fine-tuning Inception model: We fine-tuned all layers, except for the top 2 pre-trained layers which contain more generic data-independent weights. The original classification layer "loss3/classifier" outputs predictions for 1000 classes. We replaced it with a new binary layer to classify the three types of dogs in the dataset.

VI. *Approach*

A. *Tensorflow setup*

TensorFlow is an open source software library for machine learning in various kinds of perceptual and language understanding tasks, released by Google Inc. To use transfer learning for classifying dog images, we used Tensorflow library [19] to load the Inception-v3 model on our local machine, retrain it on the dog image dataset and then classify new images to be one of the eleven categories - 'Australian Shepherd', 'Belgian Malinois', 'Bichon Frise', 'Bulldog', 'Cane Corso', 'Cavalier King Charles Spaniel', 'English Mastiff','Havanese', 'Portuguese Water Dog', 'Rat Terrier', 'Dachshund'. We used Linux's 'pip install' to install and run the Tensorflow.

```
# Ubuntu/Linux 64-bit, CPU only, Python 2.7
$exportTF_BINARY_URL=https://storage.googleapis.com/
tensorflow/linux/cpu/tensorflow-0.12.0rc1-cp27-none-
linux_x86_64.whl
```

```
# Python 2
$ sudo pip install --upgrade $TF_BINARY_URL
```

#Code to install the Tensorflow image.

We stored our dataset in the directory' Images'. This directory contained 275 images of 11 different breeds of dogs.. Inception is a huge image classification model with millions of parameters that can differentiate a large number of kinds of images. We only trained the final layer of that network, so training ended in a reasonable amount of time about 10 minutes.

```
$ python tensorflow/examples/image_retraining/Dog.py

--Training_data_directory = /images\
--model_dir = /ImageNet\
--output_graph= /ImageNet/classify_image_graph_def.pb\
--output_features = /features\
--output_labels = /labels\
--Saved_softmax_function = /SVC_Classifier.sav\
--
Saved_Probabilistic_softmax_function=/SVC_Classifier_Pr
obabilities.sav
--Testing_images = /Testing_images\

#Code to load the pre-trained Inception v3 model, remove
the #old final layer, and train a new one on the plant images
and test the trained model.
```

ImageNet was not trained on any of these dog breeds, originally. However, the kinds of information that make it possible for ImageNet to differentiate among 1,000 classes are also useful for distinguishing other objects. By using this pre-trained network, we used that information as input to the final classification layer that distinguished our plant classes.

B. *Bottlenecks*

The Inception v3 model comprises of many layers stacked on top of each other (see Fig 5). These layers are pre-trained and are already very valuable at finding and summarizing information that will help classify most images. We intended on training only the last layer; the previous layers retain their already-trained state. The first phase analyzes all the images on disk and calculates the bottleneck values for each of them. The layer just before the final output layer that actually does the classification is informally termed as 'Bottleneck'. This penultimate layer has been trained to output a set of values that's good enough for the classifier to use to distinguish between all the classes it's been asked to recognize. That means it has to be a meaningful and compact summary of the images, since it has to contain enough information for the classifier to make a good choice in a very small set of values. The reason our final layer retraining can work on new classes is that it turns out the kind of information needed to distinguish between all the 1,000 classes in ImageNet is often also useful to distinguish between new kinds of objects.

Because every image is reused multiple times during training and calculating each bottleneck takes a significant amount of time, it speeds things up to cache these bottleneck values on disk so they don't have to be repeatedly recalculated. By default they're stored in the /tmp/bottleneck directory. If we rerun the script, they'll be reused, so we didn’t have to wait for this part again.

C. *Classifying with the Retrained Model*

The retraining script wrote out the feature values in ‘features’ file, labels in ‘labels’ file and the outputs of the two softmax funtions in ‘SVC_Classifier.sav’ and ‘SVC_Classifier_Probabilities.sav’. We used a Python script to load the retrained graph and the text labels using the Tensorflow library. We then inputted the new image to the graph to get first prediction.

$$x'_i = \frac{1}{\{1 + e^{\{-\frac{\{x_i - \mu_i\}}{\sigma_i}\}}\}}$$

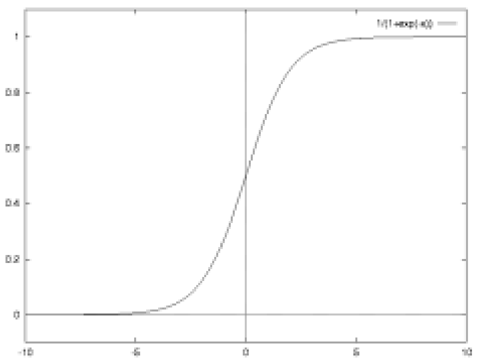

Fig 9: Softmax function

Finally, we tested new and fresh images on the retrained model to check its accuracy.

VII. Results

Once we extracted all the layers except the last one from the Inception-v3 model, the actual training of the top layer of the network began. We assessed the whole process using training accuracy, testing accuracy and cross validation. The training accuracy shows what percent of the images used in the current training batch were labeled with the correct class. The key difference is that the training accuracy is based on images that the network has been able to learn from so the network can overfit to the noise in the training data. A true measure of the performance of the network is to measure its performance on a data set not contained in the training data. The testing accuracy represents how well the trained model is classifying completely new images. The validation accuracy is used during training process and is mainly carried out to detect and avoid overfitting in the model. If the train accuracy is high but the cross validation accuracy remains low, that means the network is overfitting and memorizing particular features in the training images that aren't helpful more generally. The training's objective was to make the loss as small as possible, so we could tell if the learning was working by watching out for whether the loss continued inclining downwards, disregarding the short-term noise.

During every epoch of training, the model picked up one image, found their features, updated its filter value and feed them into the final layer to get prediction. Those predictions were then compared against the actual labels to update the final layer's weights through the back-propagation process. As the process continued, the reported accuracy improved, and after all the epochs, a final test accuracy evaluation was run on a set of images kept separate from the training and cross validation pictures. This test evaluation is the best estimate of how the trained model will perform on the classification task. We achieved an accuracy value of 96%. This number is based on the percent of the images in the test set that are given the correct label after the model is fully trained.

VIII. CONCLUSION

This paper proposes a method to classify dog breeds from various dog images using transfer learning. We demonstrated an application of transfer learing in the context of dog breed classification and introduce the concept of retraining the Inception model released by Google.Inc to classify dog image data. In this experiment, 275 dog images belonging to 11 different dog categories are used to retrained a model. We achieved a remarkable 96% accuracy in correctly classifying a new data. In transfer learning, we are mostly concerned with training the fully connected layers because the convolutional layers act as feature extractors; however there have been advantages in also training the final convolutional layer, so this would be an avenue worth exploring.

In future work, we hope to explore some more classification models and make them from scratch. We also wish to do the image processing parts like filter values, feature extraction, edge detection ourselves instead of using a pre-trained model and then compare the results. One of the most exciting things that can be done is Ensemble Learning i.e combining the results of various classification models and observing the results. In addition, we hope to introduce approaches that can take heterogeneous features and temporal dimensions into account with the techniques mentioned.

ACKNOWLEDGMENT

Pratik Devikar would like to thank Google. Inc for making Inception-v3 model and Tensorflow library open source to help researchers like us to progress in this fascinating field of Machine Learning and Artificial Intelligence. He would also like to thank all the other staff of Computer Science and Engineering Department of Shri Ramdeobaba College of Engineering and Management, Nagpur, for their enormous support and help. Finally, he would like to thank Google Images for letting him use images for training purpose.

REFERENCES

[1] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur, “Dog Breed Classification Using Part Localization”, Computer Vision–ECCV 2012. Springer Berlin Heidelberg, 2012. 172-185.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton. “Imagenet classification with deep convolutional neural networks”, Advances in neural information processing systems. 2012.

[3] Wright, J.C. and Nesselrote, M.S., 1987. Classification of behavior problems in dogs: distributions of age, breed, sex and reproductive status. Appl. Anita. Behav. Sci., 19: 169-178.

[4] Kang, Tim, "Using Neural Networks for Image Classification" (2015). Master's Projects. Paper 395.

[5] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.

[6] Rajat Raina , Alexis Battle , Honglak Lee , Benjamin Packer , Andrew Y. Ng, Self-taught learning: transfer learning from unlabeled data, Proceedings of the 24th international conference on Machine learning, p.759-766, June 20-24, 2007, Corvalis, Oregon [doi>10.1145/1273496.1273592]

[7] P. Prasong and K. Chamnongthai, "Face-recognition-based dog-breed classification using size and position of each local part, and PCA," *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, Phetchaburi, 2012, pp. 1-5. doi: 10.1109/ECTICon.2012.6254212

[8] O. M. Parkhi, A. Vedaldi, A. Zisserman and C. V. Jawahar, "Cats and dogs," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, 2012, pp. 3498-3505. doi: 10.1109/CVPR.2012.6248092

[9] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In Computer Vision, 2009 IEEE 12th International Conference on, pages 606–613. IEEE, 2009.

[10] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[11] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy), June 2010. Oral Presentation.

[12] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In Computer Vision–ECCV 2010, pages 438–451. Springer, 2010.

[13] Y. Zhang *et al.*, "Weakly Supervised Fine-Grained Categorization With Part-Based Image Representation," in *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1713-1725, April 2016. doi: 10.1109/TIP.2016.2531289

[14] D. Wang, Z. Shen, J. Shao, W. Zhang, X. Xue and Z. Zhang, "Multiple Granularity Descriptors for Fine-Grained Categorization," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, 2015, pp. 2399-2406. doi: 10.1109/ICCV.2015.276

[15] L. Zhang, Y. Yang, M. Wang, R. Hong, L. Nie and X. Li, "Detecting Densely Distributed Graph Patterns for Fine-Grained Image Categorization," in *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 553-565, Feb. 2016. doi: 10.1109/TIP.2015.2502147

[16] Guotian Xie, Kuiyuan Yang, Yalong Bai, Min Shang, Yong Rui and Jianhuang Lai, "Improve dog recognition by mining more information from both click-through logs and pre-trained models," *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Seattle, WA, 2016, pp. 1-4. doi: 10.1109/ICMEW.2016.7574665

[17] Wenbo Li and Chuan Ke, "Ensemble deep neural networks for domain-specific Image Recognition," *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Seattle, WA, 2016, pp. 1-4. doi: 10.1109/ICMEW.2016.757466

[18] C. Szegedy *et al.*, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 1-9. doi: 10.1109/CVPR.2015.7298594

[1] Abadi, Martín. "TensorFlow: Learning Functions at Scale." Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming - ICFP 2016 (2016): n. pag. Web.

Pratik Devikar is pursuing Bachelor of Engineering in Computer Science & Engineering from Shri Ramdeobaba College of Engineering and Management, Nagpur, India. His research interests include Machine Learning, Natural Language Processing, Computer Vision and Artificial Intelligence.