**Dog Identification App**

**Project Overview/Domain Background**

The task of assigning breed to dogs from images is considered exceptionally challenging. Even humanwould have trouble distinguishing between a Brittany and a Welsh Springer Spaniel. In this project, I created a dog identification application capable of classifying more than 100 different kinds of dog breeds.

**Problem Statement**

To build an app that takes user-supplied images as input and provides an estimate of the canine’s breed of dog is detected in the image. If human is detected in the image, it will provide an estimate of the dog breed that is most resembling.

**Metrics**

Accuracy is one of the common metrics for classifiers. The model is evaluated based on accuracy as described below.

Accuracy = (TN + TP)/(TN+TP+FN+FP) = (Number of correct assessments)/Number of all assessments)

Where

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

**Data Exploration**

There are 13233 human images in the human dataset and 8351 dog images in the dog dataset. Data is imbalanced as there are more human images compared to dog images. Also the number of dog images for each breed are not very similar as can be seen from the below.

The number of images in train, valid, test are: 'train': 6680, 'valid': 835, 'test': 836

root@942bf67450f6:/data/dog\_images/train# du -a | cut -d/ -f2 | sort | uniq -c | sort -nr

78 005.Alaskan\_malamute

75 029.Border\_collie

74 015.Basset\_hound

72 057.Dalmatian

70 041.Bullmastiff

70 039.Bull\_terrier

70 014.Basenji

68 046.Cavalier\_king\_charles\_spaniel

67 087.Irish\_terrier

67 012.Australian\_shepherd

67 011.Australian\_cattle\_dog

67 008.American\_staffordshire\_terrier

66 056.Dachshund

66 036.Briard

66 032.Boston\_terrier

66 023.Bernese\_mountain\_dog

65 076.Golden\_retriever

65 044.Cane\_corso

65 034.Boxer

65 027.Bloodhound

65 021.Belgian\_sheepdog

65 006.American\_eskimo\_dog

65 001.Affenpinscher

64 115.Papillon

64 068.Flat-coated\_retriever

64 042.Cairn\_terrier

64 004.Akita

63 071.German\_shepherd\_dog

63 051.Chow\_chow

63 024.Bichon\_frise

63 020.Belgian\_malinois

63 017.Bearded\_collie

62 082.Havanese

62 061.English\_cocker\_spaniel

61 060.Dogue\_de\_bordeaux

60 079.Great\_pyrenees

60 016.Beagle

59 103.Mastiff

59 090.Italian\_greyhound

59 002.Afghan\_hound

58 091.Japanese\_chin

58 054.Collie

58 038.Brussels\_griffon

57 081.Greyhound

57 031.Borzoi

55 112.Nova\_scotia\_duck\_tolling\_retriever

55 048.Chihuahua

55 047.Chesapeake\_bay\_retriever

54 118.Pembroke\_welsh\_corgi

54 089.Irish\_wolfhound

54 086.Irish\_setter

54 063.English\_springer\_spaniel

54 062.English\_setter

54 045.Cardigan\_welsh\_corgi

54 040.Bulldog

54 035.Boykin\_spaniel

53 030.Border\_terrier

53 003.Airedale\_terrier

52 088.Irish\_water\_spaniel

52 069.French\_bulldog

51 124.Poodle

51 106.Newfoundland

51 097.Lakeland\_terrier

51 084.Icelandic\_sheepdog

51 058.Dandie\_dinmont\_terrier

51 055.Curly-coated\_retriever

51 050.Chinese\_shar-pei

51 049.Chinese\_crested

51 043.Canaan\_dog

51 037.Brittany

51 018.Beauceron

51 010.Anatolian\_shepherd\_dog

51 007.American\_foxhound

50 095.Kuvasz

50 052.Clumber\_spaniel

49 129.Tibetan\_mastiff

49 117.Pekingese

49 101.Maltese

49 072.German\_shorthaired\_pointer

49 019.Bedlington\_terrier

48 070.German\_pinscher

48 059.Doberman\_pinscher

48 053.Cocker\_spaniel

48 022.Belgian\_tervuren

47 107.Norfolk\_terrier

47 098.Leonberger

47 083.Ibizan\_hound

47 080.Greater\_swiss\_mountain\_dog

47 013.Australian\_terrier

46 109.Norwegian\_elkhound

46 033.Bouvier\_des\_flandres

45 130.Welsh\_springer\_spaniel

45 123.Pomeranian

45 111.Norwich\_terrier

45 094.Komondor

45 092.Keeshond

45 075.Glen\_of\_imaal\_terrier

44 096.Labrador\_retriever

44 077.Gordon\_setter

43 104.Miniature\_schnauzer

43 099.Lhasa\_apso

43 073.German\_wirehaired\_pointer

43 065.Entlebucher\_mountain\_dog

42 127.Silky\_terrier

42 074.Giant\_schnauzer

42 026.Black\_russian\_terrier

41 078.Great\_dane

40 120.Pharaoh\_hound

40 113.Old\_english\_sheepdog

40 064.English\_toy\_spaniel

38 085.Irish\_red\_and\_white\_setter

38 025.Black\_and\_tan\_coonhound

36 114.Otterhound

36 093.Kerry\_blue\_terrier

36 028.Bluetick\_coonhound

35 125.Portuguese\_water\_dog

35 100.Lowchen

35 067.Finnish\_spitz

35 009.American\_water\_spaniel

34 110.Norwegian\_lundehund

34 066.Field\_spaniel

33 122.Pointer

32 119.Petit\_basset\_griffon\_vendeen

32 105.Neapolitan\_mastiff

31 133.Yorkshire\_terrier

31 131.Wirehaired\_pointing\_griffon

31 128.Smooth\_fox\_terrier

31 126.Saint\_bernard

31 116.Parson\_russell\_terrier

30 102.Manchester\_terrier

29 121.Plott

27 132.Xoloitzcuintli

27 108.Norwegian\_buhund

**Exploratory visualization**

**Dog Images**

**A dog looking at the camera

Description automatically generated**

**A dog looking at the camera

Description automatically generated**

**A dog looking at the camera

Description automatically generated**

**Human images**

**A screen shot of a person

Description automatically generated**

**Algorithms and Techniques**

1. CNN’s were used.
2. Initially a custom CNN architecture with 4 CNN layers and 2 FC layers was used.
   1. Drop out was applied between FC layers to avoid overfitting.
3. Then transfer learning technique was used.
   1. Pretrained resnet 50 is taken as network.

**Benchmark**

1. Custom CNN is trained with 4 layers and got an accuracy of 14%.

**Data Preprocessing**

**Train data:**

1. Random resized crop of 224x244 is chosen.
2. Horizontal flip is done.
3. Images are normalized with mean and standard deviation values used for pretrained models as mentioned in PyTorch documentation.

**Valid & Test data:**

1. Images are resized to 256x256.
2. Center crop of 224x224 is taken.
3. Images are normalized with mean and standard deviation values used for pretrained models as mentioned in PyTorch documentation.

**Implementation**

CNN have outperformed all classic computer vision techniques and have become one of the widely used methods to solve computer vision problems especially classification, object detection and segmentation tasks.

Created a custom CNN from scratch.

1. Has 4 conv layers with filters of 16, 32, 64, 128 respectively.
2. 2 fully connected layers with 1000, 133 nodes respectively to classify dog breeds.
3. Last fc layer has 133 nodes since we have 133 classes to distinguish from Drop out was applied between fc layers to avoid overfitting.

**Refinement**

Used transfer learning technique and trained a resnet50 model. Resnet is one of the classic architectures from Microsoft and have shown to do well for classification tasks.

1. Since resnet50 is trained on ImageNet dataset that has many different dog images, we can be confident that the parameters learned for resnet on ImageNet dataset would generalize well for our dataset too.

**Model evaluation and validation**

Custom CNN

1. Obtained test accuracy of 17% on test set after training for 20 epochs.
2. Cross entropy is used as loss function and SGD is used as optimizer.

Transfer learning resnet

1. Obtained test accuracy of 83% on test set after training for 6 epochs.
2. Cross entropy is used as loss function and Adam is used as optimizer.
3. Since resnet is of much bigger architecture, it can represent and encode complex patterns and performs well.

**Justification**

The transferr learned resnet obtains an accuracy of 83% which is not bad given the variation in dogs (133 classes). Also its much higher than the accuracy of 17% that we got initially.