Udacity Machine Learning Nanodegree Chelsea Friday

Capstone Proposal September 2020

Dog Breed Classifier Using Convolutional Neural Networks

Domain Background

Have you ever been on a walk and spotted a dog that you were not quite sure of the breed? Well I have! I live across from a dog park and am always spotting dogs which I can't quite distinguish if they are this breed or that? Think, a Boston Terrier vs. a French Bulldog. Or, an Alaskan Malamute vs. Siberian Husky.

This project not only gives me an opportunity to solve a silly personal problem but also to work with Convolutional Neural Networks (CNN) that I then could manipulate and apply to other image classification problems. Problems like plant species detection or skin mark detection. As we can see from the table below taken from the paper "Deep learning and transfer learning approaches for image classification"[4] CNN can be applied to many different types of datasets with relatively high accuracy making them fun and effective models to work with.

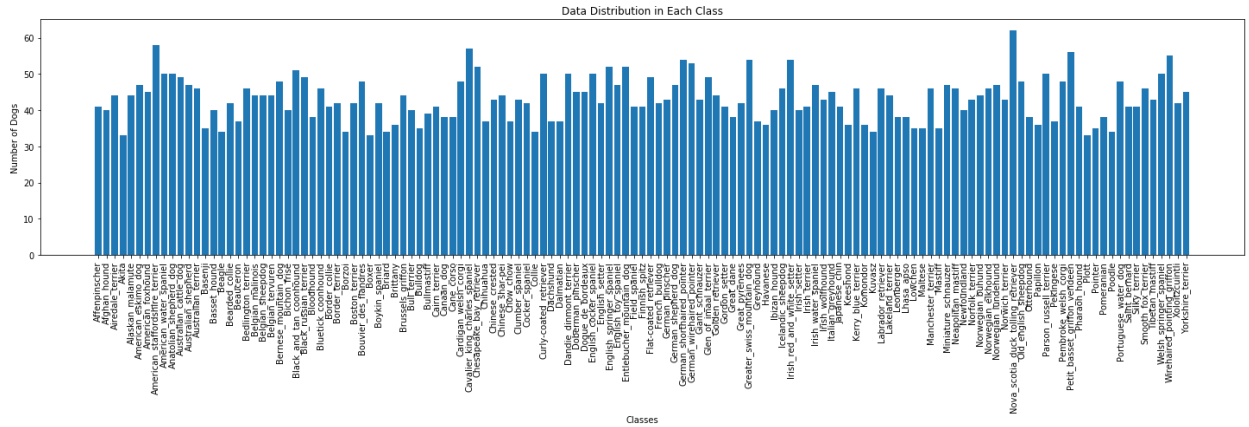
No.	Methods used	Data sets used	Training (Images/%)	Testing (Images/%)	Accuracy
1	Transfer learning with AlexNet, GoogleNet,ResNet50	CIFAR10	50000	10000	GoogleNet-68.95%, ResNet50-52.55%, AlexNet-13%
		CIFAR100	50000	10000	
2	Proposed a network contains 5 convolutional and 3 fully connected layers	ImageNet Fall 2011, 15M images, 22K categories	7.5M	7.5M	Error rates top-5 : 37.5%, top-1 : 17.0%.
3	Transfer learning, web data augmentation technique with Alex,vgg16,res net-152	Flowers102	8189		92.5%
		dogs	20580		79.8%
		Caltech101	9146		89.3%
		event8	1579		95.1%
		15scene	4485		90.6%
		67 Indore scene	15620		72.1%
4	Transfer Learning with Inceptiov3 model	CIFAR10	50000	10000	70.1%
		Caltech Face	12150		65.7%
					500 epochs-91%
					4000 epochs-96.5%
5	CNN deep learning	Caltech 101	9146		96%
6	Compare NN models	DR	77%	23%	LeNet-72%, VGGNet-76%
		CT	72%	28%	LeNet-71%, VGGNet-78%
7	An autonomous learning algorithm automatically generate Genetic DCNN architecture	MNIST, CIF AR10, CIFAR100	90%	10%	99.72% on MNIST, 89.23% on CIFAR10, 66.70% on CIFAR100
8	Google's Inception-v3	Dogs			96%
9	CNN + AdaBoost	CIFAR-10	80%	20%	88.4%

Problem Statement

The main objective of this project will be to build a machine learning model that can be used within a web app to process real-world, user-supplied images. Given an image of a dog, the algorithm will return a predicted dog breed. Given an image of a human, the algorithm will identify which dog breed the human face most closely resembles

Datasets and Inputs

There are two datasets for this project provided by Udacity. The dog images dataset has 8,351 total images. This dataset is sorted into 6,680 train images, 836 test images, and 835 validation images. Each has 133 folders corresponding to 133 dog breeds. The data is relatively balanced. As we can see from the bar graph below. The human images dataset contains 13,233 total images. The images will be used as input data to create a feature map to feed into the CNN.



Solution Statement

For this classification problem a CNN will be used. A CNN is a deep learning algorithm which can take in an input, an image, and assign it a class, a dog breed. We will first create the feature detectors. We have the image inputs - the human and dog image datasets. We will use the human dataset and existing Haar feature based Cascade Classifier to create a human face detector. We will use the dog image dataset and a pre-trained VGG16 model to create a dog face detector. Once this is done, we can take in an image and apply the feature to create a feature map. From this we can create a CNN that after being trained through forward and back propagation can identify and classify a dog breed from an image.

Benchmark Model

We will be using the CNN model created from scratch as a benchmark model to compare to the CNN model created using Transfer Learning. The CNN model created from scratch must attain a test accuracy

of at least 10%. This can confirm that the model is working because a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

Evaluation Metrics

Since we are dealing with a multi-classification problem that is relatively well balanced we will use accuracy as our evaluation metric.

Project Design

Step 0: Import datasets and libraries.

Step 1: Use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.

Step 2: Create a dog detector using the pre-trained VGG-16 model along with weights that have been trained on ImageNet.

Step 3: Create a CNN to classify dog breeds from scratch. Train, validate, and test model.

Step 4: Create a CNN to classify dog breeds using Transfer Learning. Train, validate, and test model.

Step 5: Write and test an algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither. Then,

- If a dog is detected in the image, return the predicted breed.
- If a human is detected in the image, return the resembling dog breed.
- If neither a dog nor human detected in the image, provide output that indicates an error.

References

[1] Brownlee, J. (2020) How to Choose Loss Functions When Training Deep Learning Neural Networks. *Machine Learning Mastery* [Online]. Available: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[2] Kaur, T., Gandhi, T.K. (2020) Deep convolutional neural networks with transfer learning for automated brain image classification. *Machine Vision and Applications* 31, 20 [Online]. Available: https://doi.org/10.1007/s00138-020-01069-2

[3] Krishna, S., Kalluri, H. (2019) Deep learning and transfer learning approaches for image classification. International Journal of Recent Technology and Engineering Vol. 7, Issue. 5S4 [Online]. Available: https://www.ijrte.org/wp-content/uploads/papers/v7i5s4/E10900275S419.pdf

[4] Han, D., Liu, Q., Fan, W. (2017) A New Image Classification Method Using CNN transfer learning and Web Data Augmentation. *Expert Systems with Applications. Elsevier* Vol. 95, pp. 43-56 [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0957417417307844

[5] OpenCV (2013) Tutorial Cascade Classifier [Online]. Available: https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

[6] PyTorch (2020) Torchvision Models. *torchvision.models - PyTorch 1.6.0 documentation* [Online]. Available: https://pytorch.org/docs/stable/torchvision/models.html

[7] Udacity (2019) project-dog-classification [Source code]. Available: https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification