Capstone Project Proposal Ankit Kumar Saini

Machine Learning Engineering Nanodegree July 10th, 2021

Dog Breed Classifier Model Using CNN

1. **Domain Background**
   Image classification is a standard computer vision task in the Deep Learning domain. The use of convolutional neural networks (CNNs) have revolutionized the image classification, object detection and segmentation tasks. Dog Breed identification is a well-known multi class image classification task in deep learning. The task is to implement an end-to-end deep learning pipeline that can be used within a web or mobile app to process real-world, user-supplied images. The pipeline will accept any user-supplied image as input and will predict whether a dog or human is present in the image. If a dog is detected in the image, it will provide an estimate of the dog's breed. If a human is detected, it will provide an estimate of the dog breed that is most resembling. The model will be deployed as a web app using a flask server. I chose this as my capstone project because it allows me to build and deploy a CNN model.

2. **Problem Statement**
   The objective of this capstone project is to build and train a deep learning model that can be deployed as a web or mobile app for real-world use cases. There are two important tasks that need to be done as part of this project.

   - **Dog Detector:** This model will identify whether a dog is present in the image or not.
   - **Human Face Detector:** This model will identify whether a human face is present in the image or not.

   Once the content of the image is identified (dog/human face), the dog breed classifier model will predict the dog breed in case of a dog image and the most resembling dog breed in case of a human image.

3. **Datasets and Inputs**
   For this capstone project, the input data is in the form of RGB images. Two different datasets were provided by Udacity for this project.
   - **Human images dataset:** The first dataset is Labeled Faces in the Wild containing 13233 human images. All images are 250x250 pixels in height and width. Most of the images in the dataset contain a single human face.

- **Dog breed dataset:** The second dataset is the dog breed dataset containing 8351 dog images with 133 dog breeds. It is divided into training (6680 images), validation (835 images) and test (836 images) sets. The images are of different sizes with different backgrounds. The dataset is not balanced. There are few classes that have less than 30 images while there are some classes that have more than 70 images.

4. Solution Statement

The task in this capstone project will be solved using convolutional neural networks (CNNs). CNNs use kernels/filters to automate the task of feature extraction from the images in an incremental manner. Earlier layers in a deep neural network get specialized in detecting lower level features such as edges, texture, corners, high frequencies, etc. and deeper layers detect object level features such as faces and other complex geometrical shapes by combining the lower level features to classify, detect and segment objects in images.

- I will be using three models in this project. First model will be a human face detector. I will be using a pre-trained Haar cascade face detector model from the OpenCV library to determine if a human face is present in the image or not.

- Second model will be a dog detector. I will be using a pre-trained ResNet50 model for dog detection. This model has been trained on ImageNet, a very large and popular dataset for image classification and other vision tasks.

- Third model will be a dog breed classifier. I will be using a pre-trained EfficientNetB4 model for feature extraction. A GlobalAveragePooing layer followed by a Dense layer with softmax activation will be added on top of this pre-trained model. This model will predict the breed of the dog.

5. Benchmark Model

- The pre-trained CNN model with ImageNet weights trained using Transfer Learning should achieve an accuracy of 70% or above.
- The CNN model built and trained from scratch should have an accuracy of 6% or above. This will ensure that the model is learning from the data and is not guessing randomly.

6. Evaluation Metrics

To evaluate the performance of the dog breed classifier model, accuracy, recall, precision, f1-score and confusion matrix were used. Accuracy is a good indicator of model performance in case of a balanced dataset. But it is not sufficient to tell in which cases the model is performing good and where it needs improvement. The precision of a class defines how trustable the result is when the model classifies that an image belongs to a particular class. The recall of a class is defined as how well the model is able to detect that class. The F1-score of a class is given by the harmonic mean of precision and recall. It combines precision and recall of a class in one metric. Confusion matrix displays the correct and incorrect classifications predicted by the model.

7. Project Design

The general workflow for this project will be as follows:
Step 0: Import necessary modules and define some global variables (such as paths of datasets, image size, batch size, etc.)
Step 1: Load and preprocess the datasets (image resizing, normalization)
Step 2: Perform exploratory data analysis (display few random images as well)
Step 3: Define data augmentation functions
Step 4: Load pre-trained models (human face-detector and dog-detector)
Step 5: Built, train and evaluate CNN model from scratch
Step 6: Built and train EfficientNetB4 model to classify dog breeds
Step 7: Evaluate model performance using metrics defined above
Step 8: Write an algorithm to combine the three models (human face detector, dog detector and dog breed classifier) to perform the given task

- If dog is present in the input image, return the predicted dog breed
- If human is present in the input image, return the most resembling dog breed
- If none is detected, display an error message (only human or dog images)

Step 9: Deploy this algorithm on a web app using flask

8. References

1. EfficientNet paper: arXiv:1905.11946
2. OpenCV cascade classifier: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
3. ResNet50: https://github.com/keras-team/keras-applications/releases/download/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5'
4. TensorFlow documentation: https://www.tensorflow.org/api_docs