Machine Learning Engineer Capstone Proposal

CNN Project: Dog Breed Classifier

Tingting Zhou

November 14, 2020

Project Background

Nowadays, Machine Learning could be one of the most popular words, people may hear it through various scenarios, such as commercials, finance news and some advanced medical fields. However, we may still think Machine Learning application is still a bit far from real life, although we've already used Alexander to play music for us. Mainly it's because most of people don't have the access to machine learning skills or have the opportunity to build a machine learning application from scratch, and most of the related topics are full of notations and we can not implement the algorithm into real life. Therefore, it interests me that if I can use deep learning algorithm to build applications that close to our life, for example, take a picture on the road when we come across a dog, and by sending the photo to an APP, it will tell us the breed of the dog.

It's easy to tell if an image is picturing human being or dogs for human being. However, it's always a challenging job for computer. Thanks to the access to big data and the development in machine learning algorithm, image classification is one of the areas where deep learning models are very successful and popular, among the various complicated models, the Convolutional neural networks (CNN) model is a very effective class of neural networks that is highly effective at the task of image classifying, object detection and other computer vison problems.[1] With the development of large data age, (CNN) with more hidden layers have more complex network structure and more powerful feature learning and feature expression abilities than traditional machine learning methods. [2] Compared to standard feedforward neural networks with similarly-sized layers, CNN have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse. Despite the attractive qualities of CNN, and despite the relative efficiency of their local architecture, they have still been prohibitively expensive to apply in large scale to high-resolution images. Luckily, current GPUs, paired with a highly-optimized implementation of 2D convolution, are powerful enough to facilitate the training of interestingly-large CNN, and recent datasets such as ImageNet contain enough labeled examples to train such models without severe overfitting.[3]

In practice, very few people train an entire CNN from scratch, because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pre-train a ConvNet on a very large dataset(e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest.[4]

Therefore, in this project, I will not only build a simple classifier model from scratch using CNN but also build another transfer learning CNN model from some pre-trained models in Pytorch, which should be able to attain at least 60% accuracy on the test set.

Problem Formulation

The application aims to help people identify dog breeds, it will have two main functions, after feeding it with a image, the application should tell users whether it's a photo of human being or dog, and it will return the breed of the dog.

Among the above three functionalities, identifying dog breeds is exceptionally challenging, the same breed may look different due to dog age, color and subspecies, on the contrary, different breeds may look similar , such as young golden and Labrador Retriever. I will build a CNN breeds classifier from scratch, also build a transform learning model in order to attain a higher accuracy.

This project will be implemented in Jupyter Notebook dog_app, provided by Udacity Machine Learning engineer nanodegree team, which already embedded instructions, template codes, referred materials and test functions, I will follow the guidelines, implement additional functionality and improve the model performance.

Datasets

In this project, the CNN model needs to be trained with a large dataset, here, the dataset includes 13,233 human images and 8,351 labeled dog images from 133 breeds, which is provided by the Udacity Machine Learning engineer nanodegree team, the human being images will be only used for human detector, and they're uniformly sized with 256 by 256 pixels, while the dog dataset will be split into training set with 6,680 images, validation set with 835 images and 836 images in the test set. Those dog images are RBG with size ranging from hundreds by hundreds to thousands by thousands of pixels, which will need to be transformed to uniform size to feed the model.

Besides, I will also provide 6 more local images among which are two human images and four dog images for testing the application performance.

Solution Statement

As mentioned in the problem formulation, in the project, I need to build a human face detector, dog face detector and a dog classifier to make those functions work. The first two detectors will be used for identifying the existence of human faces and dog faces, Udacity team has provided some sample code:

- Human detector: Use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.
- Dog detector: I will use a pre-trained VGG-16 model, this model will take the path to an image as input and return the index corresponding to the ImageNet different categories, since there are 1000 categories in the ImageNet dataset, the output of this model is expected to be a integer between 0 and 999, inclusive. However, only an uninterrupted sequence includes all categories for dog breeds, so for a dog face detector, we only need to check if the output is in 151 and 268(inclusive).

Once the detector confirms there is a dog face presented in the input image, the image path will be passed to the breed classifier to identify its breed. Otherwise, the application will return that a human face is detected or require another image if neither human face nor dog face has been determined from the input image.

Dog breed classifier is the essential function of this project, I will firstly build a CNN model with relatively simple architecture. However, in order to attain a higher prediction accuracy score, I will build another classifier that transformed from pre-trained models, such as AlexNet, ResNet50, ResNet101 and VGG16, the final model will be determined by the performance of each candidate model, the evaluation will include test accuracy score and test loss value.

Evaluation Metrics

Since the evaluation will be depend on accuracy score and test loss, I need to address the definition of the evaluation metrics here:

- Accuracy: our model is a classifier, the accuracy will depend on the correct predictions that the model determines, the accuracy score will be calculated as:

$$\text{Accuracy}=\frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- Loss function: For this image classification problem, I will use Cross-entropy loss function, which measures the performance of classification model whose output is a probability value between 0 and 1, and the loss penalizes both types of errors, but especially those predictions that are confident and wrong. The formula is described by the Pytorch documentation as below:

$$\text{loss}(x,\text{class}) = -\log\left(\frac{\exp(x[\text{class}])}{\sum_j \exp(x[j])}\right) = -x[\text{class}] + \log\left(\sum_j \exp(x[j])\right)$$

  or in the case of weight argument being specified:

$$\text{loss}(x,\text{class}) = \text{weight}[\text{class}]\left(-x[\text{class}] + \log\left(\sum_j \exp(x[j])\right)\right)$$

  The loss value will need to be calculated for training set, validation set and test set, by comparing the loss value between each dataset, I can tell whether the model under-fits or over-fits the training dataset, if the training loss is much lower than the validation loss, the model might overfit, on the contrary, if both the validation loss and training loss are high, the model may be under-fitting.

Benchmark

There are two classification models in this project, the simple-structure CNN model will be developed from scratch, and the accuracy score is expected be over 10% as it's very difficult to make enough right predictions with a simple architecture model, and the random guess will provide a correct answer with a accuracy of less than 1%.

For the transfer learning model, the accuracy is expected to be over 60% as the model will be built from much more complicated pre-trained model.

Project Design

Step 0: Import Datasets

The datasets are provided by Udacity Machine Learning Engineer team, the image paths will be saved as Numpy arrays, there are 13,233 total human images and 8,351 total dog images, during this process, I will implement some data visualization to explore the attributes of the dataset, also I can determine how to prepare the data for following steps.

Step 1: Detect Humans

In this section, the human detector function is already provided by Udacity team, the function is implemented with Haar feature-based cascade classifiers in OpenCV, which provides many pre-trained face detectors. This function will take a file path as input and return with a Boolean to indicate whether a human face is detected. I will write a assess function to evaluate the human detector's performance with 100 dog images and 100 human images.

Step 2: Detect Dogs

A pre-trained VGG-16 model will be used in this section, which includes 13 convolutional layers and 3 linear layers with weights, and it's considered to be one of the excellent vision model architecture till date. All the pre_trained models from Pytorch expect inputs images normalized in the same way: mini-batches of 3-channel RGB images of shape (3 x H x W), where H and W are expected to be at least 224. The images have to be loaded in to a range of [0, 1] and then normalized using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225], so I will define a data processor function to resize the input images, and the function is expected to return an index through 151 to 269 indicating the dog categories classification. Afterwards, I will assemble the dog detector function into another assessor and evaluate its performance

Step 3: Create a CNN to Classify Dog Breeds (from Scratch)

First, I will write data loaders for my training, validation and test datasets, in this part, I will resize and crop the input images, normalize the data and converter them to tensor, in order to augment my training set, I will implement random horizontal flip and rotations. Once the data is prepared, I will start to build the model with a simple 3 convolution layers and a linear layer that sample code provides, then I will try different architecture and hyper-parameters depending on the validation and test performance that evaluated by the metrics. I plan to follow part of the architecture of the VGG-16 model if the model still doesn't generate good results after several trials.

Step 4: Create a CNN to Classify Dog Breeds (using Transfer Learning)

Same as mentioned in the Step 3, in this section, I will also firstly build data loaders for training, validation and test sets. Here, I will copy the data loaders in the Step 3 to make the results more comparable. As I mentioned in the Solution Statement, the candidate transfer learning models will include AlexNet, ResNet50, ResNet18, ResNet34, VGG16 and ResNet101, [5] the benchmark accuracy for a not bad classier is at least 60%. Since the pre-trained models have extracted lots of the low level features, I will simple train the model with different parameters in their last layer, so we can reduce lots of training efforts and still attain a good performance.

Step 5: Wrap up Algorithm

After I finished all the above steps, I will warp up human detector, dog detector and dog breeds classifier into a run_app algorithm, this function will the final application for this project.

Step 6: Test Algorithm using local images

I will firstly test the final application with images in the loaded dataset from Udacity team, then I will upload some local images to further evaluate the performance of the application.

References

1. M. T. Islam, B. M. N. Karim Siddique, S. Rahman and T. Jabid, "Image Recognition with Deep Learning," *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, Bangkok, 2018, pp. 106-110, doi: 10.1109/ICIIBMS.2018.8550021.

2. A. M. Al-Saffar, H. Tao and M. A. Talab, "Review of deep convolution neural network in image classification," *2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, Jakarta, 2017, pp. 26-31, doi: 10.1109/ICRAMET.2017.8253139.

3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60*(6), 84-90. doi: 10.1145 / 3065386

4. Chilamkurthy, S. (2017). Transfer Learning Tutorial. In PyTorch. Retrieved from https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

5. Talo, Muhammed & yıldırım, Özal & Baloglu, Ulas & Aydin, Galip & Acharya, U Rajendra. (2019). Convolutional neural networks for multi-class brain disease detection using MRI images. Computerized Medical Imaging and Graphics. 78. 101673. 10.1016/j.compmedimag.2019.101673.