# Capstone Proposal

Chu Gong

2020-08-09

I choose Dog Breed Classifier(CNN) as my capstone project.

1. Domain Background

Computer Vision is a popular field that have derived many interesting academic researches and industry applications. CV related techniques are aimed to tackle problems such as human face recognition and object detection for automatic cars. As the develpoment of artificial intelligence and deep learning algorithms within these years, people can solve these problems with higher accuracy and efficiency. For example, from the simple CNN architectures like LeNet-5 and AlexNet, to deeper model such as VGG and GoogLeNet, these models help people achieve better performance on image classification tasks[1]. In order to get a closer look at how CNN works and build my own interesting image classification app, I choose this capstone project and hope to explore more advanced techniques of deep learning.

2. Problem Statement

The problem of this project is to build an app and relevant algorithms to detect dogs in the image and identify the breed of the dogs. For a given image that user inputs, my model are expected to output the most possible answer of the dog's breed. And if the image is not a dog but a human, my model can detect the human and give a resembling breed as output[2]. The basic model I plan to use is CNN(convolutional neural network) but the detailed architecture of cnn model need to be explored and specified. In order to measure and decide which model is more suitable for our dog recignition problem, I plan to compare the accuracy of predictions on test set.

3. Datasets and Inputs

The dataset we use is *dog dataset and human dataset* that udacity provided on Github notebook[2]. We need human dataset to train our model how to detect human and also need dog dataset that our model can learn dog breeds recognition knowledges.

All datasets we use is image type, a specical type of data that can be understood as standard inputs like features as well. Firstly the original image is RGB image that each pixel is determined by three parameters(red, green and blue). This type of data is more complex to handle directly, so we transform images to gray scale and each pixel is a feature and has value range from 0 to 255. The following cnn model can use these features as input directly.

4. Solution Statement

We need several solutions for different steps of this project. First task is to detect people and dogs in the image. In this step, we use OpenCV's pre-trained face detector and VGG-16 network to solve this problem. These pre-trained models can be implemented by extracting xml files directly from github or other open source platform. The second task is to classify dog breeds and similar dog breeds for people. We can design our own cnn model architecture from scratch or use transfer learning based on some pre-trained architecture such as *Inception or ResNet*. After specified the model architecture, the following pipeline is same for each machine learning model, from optimizer definition, model training to validation on test set. Finally I will compare the test set classification error and decide the best performance model to use.

5. Benchmark Model

When we try to tackle image classification problem with cnn model, there are two approaches we can choose, design cnn model from scratch or use pre-trained models. The prior one can help us understand the structure of convolutional network better, but it may take too much time if we want to try out a deeper structure. So working with pre-trained models can be more efficiency. Usually pre-trained models are more complex and usually have hundreds of layers. But most parameters are already trained and we can tune some high-level layers to make this model flexible enough for our specific problem and dataset. So in this project, I will use prior approach as benchmark model to get a sense of how cnn works, and then implements several pre-trained models to achieve a better performance on dog breeds classification.

I would like to choose relatively "simple" model, *LeNet-5 and AlexNet* to build from scratch as use as benchmark model. The architecture of those benchmark model are as follow[1]:

*Table 13-1. LeNet-5 architecture*

| Layer | Type | Maps | Size | Kernel size | Stride | Activation |
|---|---|---|---|---|---|---|
| Out | Fully Connected | – | 10 | – | – | RBF |
| F6 | Fully Connected | – | 84 | – | – | tanh |
| C5 | Convolution | 120 | $1 \times 1$ | $5 \times 5$ | 1 | tanh |
| S4 | Avg Pooling | 16 | $5 \times 5$ | $2 \times 2$ | 2 | tanh |
| C3 | Convolution | 16 | $10 \times 10$ | $5 \times 5$ | 1 | tanh |
| S2 | Avg Pooling | 6 | $14 \times 14$ | $2 \times 2$ | 2 | tanh |
| C1 | Convolution | 6 | $28 \times 28$ | $5 \times 5$ | 1 | tanh |
| In | Input | 1 | $32 \times 32$ | – | – | – |

*Table 13-2. AlexNet architecture*

| Layer | Type | Maps | Size | Kernel size | Stride | Padding | Activation |
|---|---|---|---|---|---|---|---|
| Out | Fully Connected | – | 1,000 | – | – | – | Softmax |
| F9 | Fully Connected | – | 4,096 | – | – | – | ReLU |
| F8 | Fully Connected | – | 4,096 | – | – | – | ReLU |
| C7 | Convolution | 256 | $13 \times 13$ | $3 \times 3$ | 1 | SAME | ReLU |
| C6 | Convolution | 384 | $13 \times 13$ | $3 \times 3$ | 1 | SAME | ReLU |
| C5 | Convolution | 384 | $13 \times 13$ | $3 \times 3$ | 1 | SAME | ReLU |
| S4 | Max Pooling | 256 | $13 \times 13$ | $3 \times 3$ | 2 | VALID | – |
| C3 | Convolution | 256 | $27 \times 27$ | $5 \times 5$ | 1 | SAME | ReLU |
| S2 | Max Pooling | 96 | $27 \times 27$ | $3 \times 3$ | 2 | VALID | – |
| C1 | Convolution | 96 | $55 \times 55$ | $11 \times 11$ | 4 | SAME | ReLU |
| In | Input | 3 (RGB) | $224 \times 224$ | – | – | – | – |

6. Evaluation Metrics

I choose *accuracy* as the evaluation metric of this project. In face detection section, we will test our detector model on 100 images of human files and 100 images of dogs files. So the accuracy can be defined as *the number of faces that model detected / 100* for each detector. In the breeds classification section, *accuracy = number of correct classification / total number of test set* .

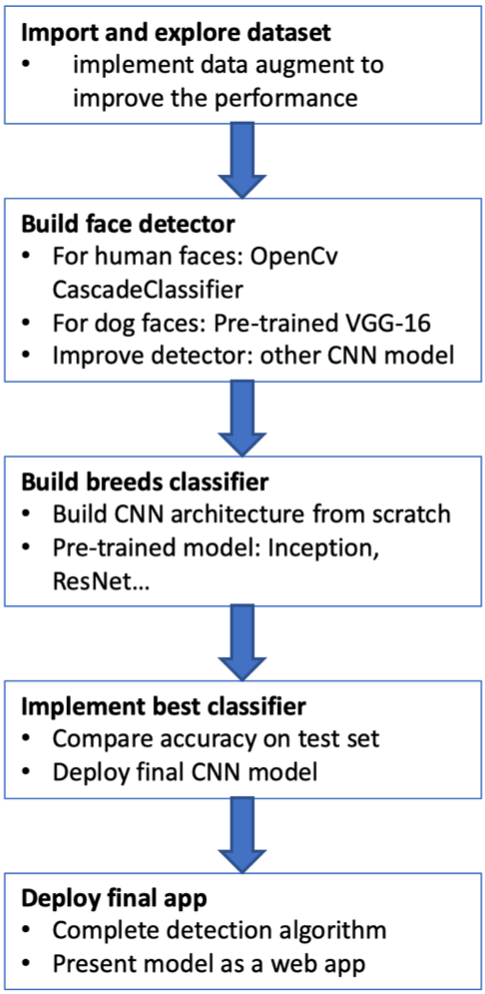
7. Project Design

project workflow:

# References


[1] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chapter 13.* O'Reilly Media, Inc.

[2] `https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification`. Udacity.


Import and explore dataset
• implement data augment to improve the performance
Build face detector
• For human faces: OpenCv CascadeClassifier
• For dog faces: Pre-trained VGG-16
• Improve detector: other CNN model
Build breeds classifier
• Build CNN architecture from scratch
• Pre-trained model: Inception, ResNet...
Implement best classifier
• Compare accuracy on test set
• Deploy final CNN model
Deploy final app
• Complete detection algorithm
• Present model as a web app