Using Mobile Neural Network For Pets Classification

Wen Jie 1155092211

Supervised by Prof. Michael R. LYU

Table of Contents

INTRODUCTION

Introduction

1. Overview

In recent years, we human beings have witnessed an expeditious development of ability of Artificial Intelligence (AI). In March 2016, AlphaGo, which is developed by Google DeepMind beat Sedol Lee, a South Korean professional Go player of 9 dan rank, in playing the board game. In May 2017, AlphaGo further challenged Chinese Go player Jie Ke who ranked number one in the world. In October 2017, Google DeepMind launched AlphaGo zero, the newest version of AlphaGo created without using data from human games. Using AI to play Go actually requires a huge amount of calculations. Compared to other games such as chess, computer usually will have a much smaller chance to beat human player in playing Go since the traditional AI algorithms such as exhaustion, Alpha–beta pruning and heuristic will not work for such amount of calculations. Under such condition, the success of AlphaGo turned deep learning into a popular topic not only in the field of computer science but also in the general public.

INTRODUCTION

Additionally, more and more governments, universities as well as industry leaders are promoting research in AI. It has suddenly become the most popular program in computer science. Number of applicants who are applying for top computer science programs in U.S. have grown rapidly, and AI track is the most popular one among all of the tracks.

Number of published articles about AI also have exploded. Giant IT companies such as Google has incorporated AI into the company's value. Universities all over the world, including CUHK, have also started new degree programs related to AI.

Take Computer Vision as an example, basically all of the top universities have programs or tracks related to this. Carnegie Mellon University even has separate Master and PhD program focus on computer vision. CUHK also has a strong reputation in the computer vision field. The reason for the popularity of computer vision techniques is because the wide range of the usages, including healthcare, robot, automobile, security, entertainment and many others.

INTRODUCTION

Many companies also provide service related to computer vision. Take Amazon as an example, they provide Amazon Rekognition to the public users which allows its users to analysis images and videos through Amazon Rekognition API. This fancy API can help users with no understanding about AI, machine learning and computer vision to do face recognition and deep analysis from the given images or videos. Amazon Rekognition has also cooperated with the local law enforcement to help them catch criminals from the crowd by comparing faces captured by the police's camera and the criminal face dataset.

While in mobile devices, computer vision techniques have also been implemented in many places. When we are taking photos using our mobile phone, we can see the face recognition function built in the mobile phone are helping us detecting faces as well as adjusting other parameters to improve the quality of photo with the focus on faces shown in the photo. For iPhone users, the FaceID feature was also adopted by Apple to replace fingerprint as the only biometric method for new iPhone devices, which proves the accuracy as well as safety for computer vision techniques.

2. Motivation

As the fast development for the server-end neural network models, neural networks for mobile devices also show great potential to apply techniques to real-world problems in a way that general public can also take advantages from the rapid development of AI. The task of my project is to come up with a mobile neural network model which can efficiently classify pets, more specifically, cats and dogs.

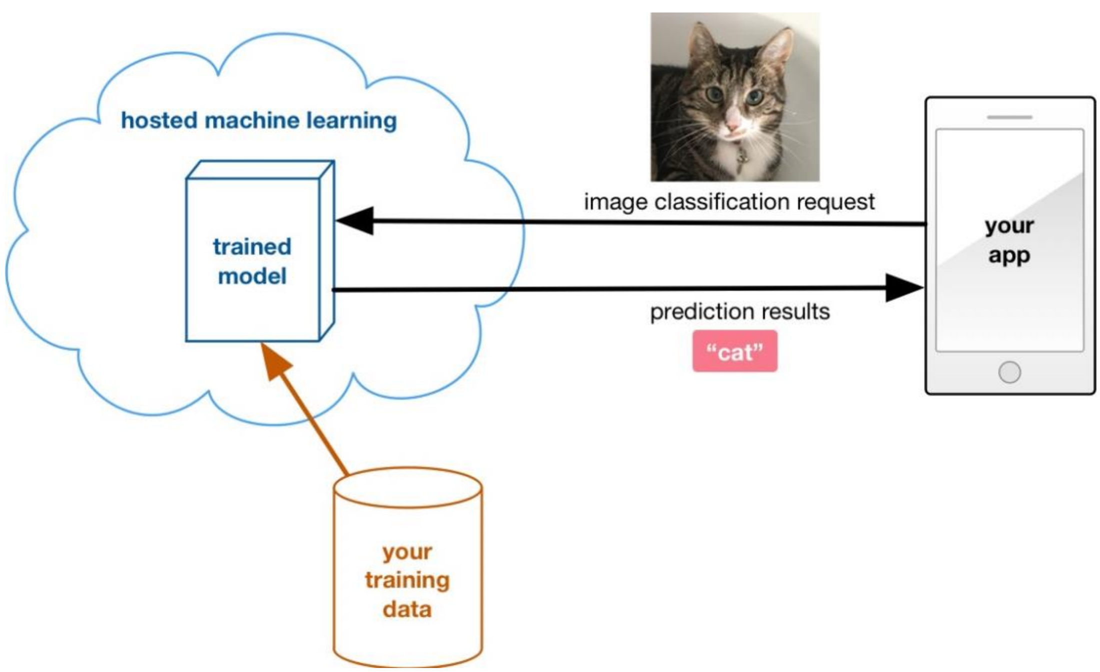


Fig 1 Example of could-based mobile image classification

INTRODUCTION

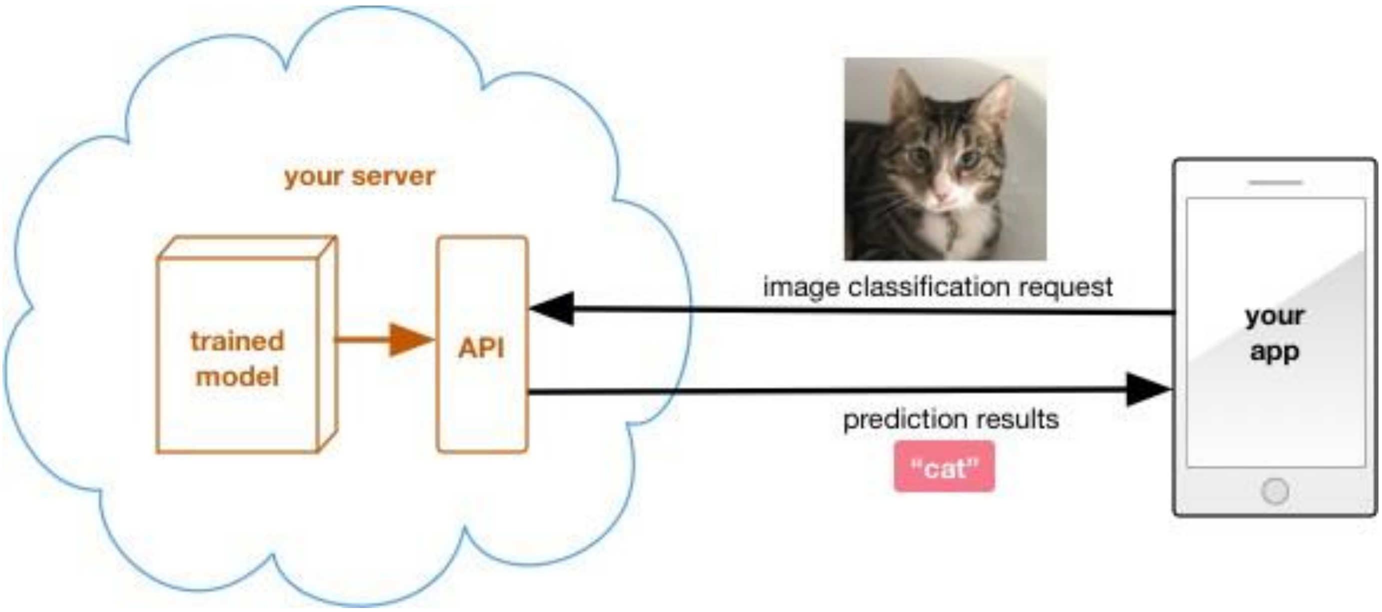


Fig 2 Example of could-based mobile image classification with API

One way to implement mobile image classification is to use method shown in Fig 1: upload the images to the cloud server and get the result from the server. Some big IT companies, such as Google and Amazon, even provide API to the public so that users without any AI experience can also utilize the powerful machine learning models. Could server can allow complicated machine learning models to “run” on mobile devices, however, with the vapid development of mobile devices’ capability as well as the machine learning algorithms and models, to run a machine learning model offline in a mobile device is not hard any more. Additionally, we can save the time and money to deploy a could server and to run the tasks anywhere without the need of Internet access. For some tasks which need to be conducted in the place where Internet is not available such as forest or open see, or in some situations where Internet is expansive to get, a model which can run offline would be even more important.

In my project, I chose to further investigate the capability of current mobile devices and the neural networks. To make this project with a more meaningful outcome, it is essential for me to focus on one specific area to work on. Noticing that in our daily life, there are many cases where we are attracted by some pets, most likely dogs and cats, but we do not know exactly what the names of these pets are. I decided to focus on classifying pets on mobile devices and building a mobile neural network model which can be used by anyone without any other costs.

Background

1. Neural Network

1.1 Introduction to neural network

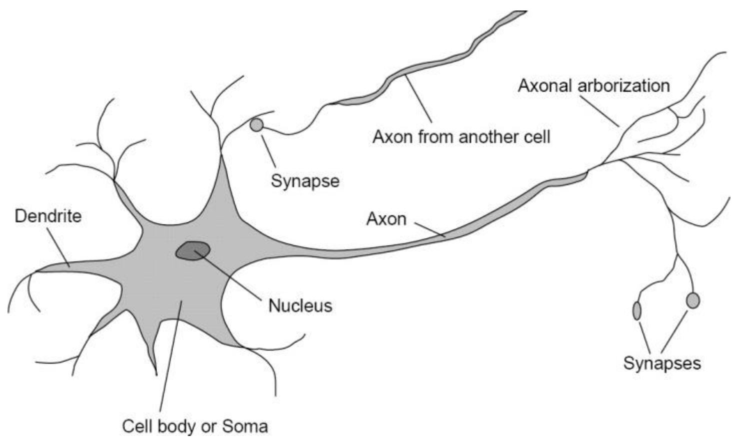


Fig 3 Example of Synapse

Twenty or thirty years ago, when people were talking about neural network, we might classify this problem as a biology problem and think about the nerve cells and systems exist in our brain. Nowadays, however, more and more people would know think about computer science when they heard about neural networks. Indeed, the idea of neural network in computer science is to simulate the structure of neural network in our brains and to make computer think and react like human beings.

BACKGROUND

In biology, the neuron, or nerve cells, is the fundamental functional unit of all nervous system tissue, including the brain. See Fig 3, the connecting junction is called a synapse and each neuron can form synapses with dozens or even hundred thousand other neurons.

Signals are propagated from neuron to neuron by complicated electrochemical reaction. Chemical transmitter substances are released from the synapses and enter the dendrite, raising or lowering the electrical potential of the cell body. These mechanisms are thought to form the basis for learning in the brain where we can get thought, action, feelings and consciousness from simple cells.

However, our brains are composed of 90 billion nerve cells, there would be no way for us represent all of them. In order to achieve our goal, we have to generate rules based on our nerve systems and to transfer the rules into structures and operations so that a computer can recognize.

1.2 Basic Neural Network

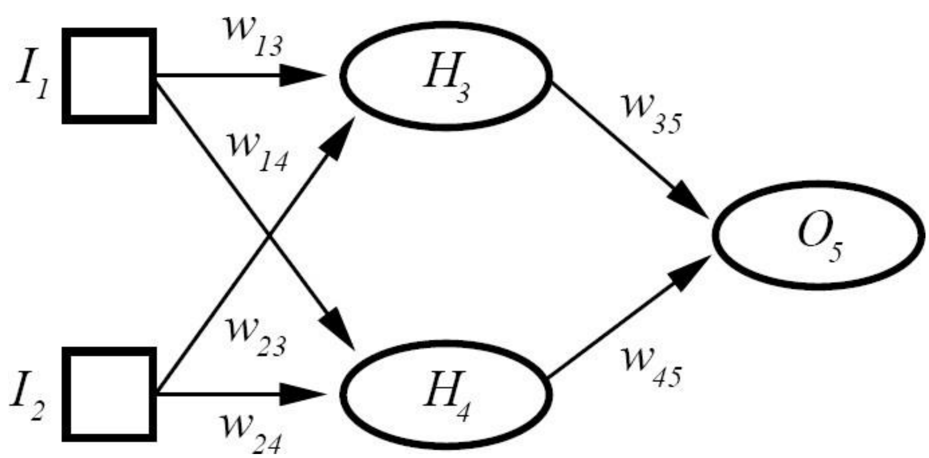


Fig 4 Basic neural network

A basic neural network is composed of a number of nodes (units) which are connected by directed arrows. Some of them are input nodes and some of them are output nodes. Each directed arrow is associated with a number which is the numeric weight of it. In such case, the learning process can be represented as the process of updating weights.

Each unit has some output links which connects to other units as well as some input links which connects some other nodes to itself. It should also have a current activation level and a standard to compute the new activation level in the next step with its inputs and weights.

To design a neural network to perform some tasks, we need to carefully choose the number and type of the nodes, layers as well as the connections. Then try to train the weights based on these.

1.3 Simple Computing lements

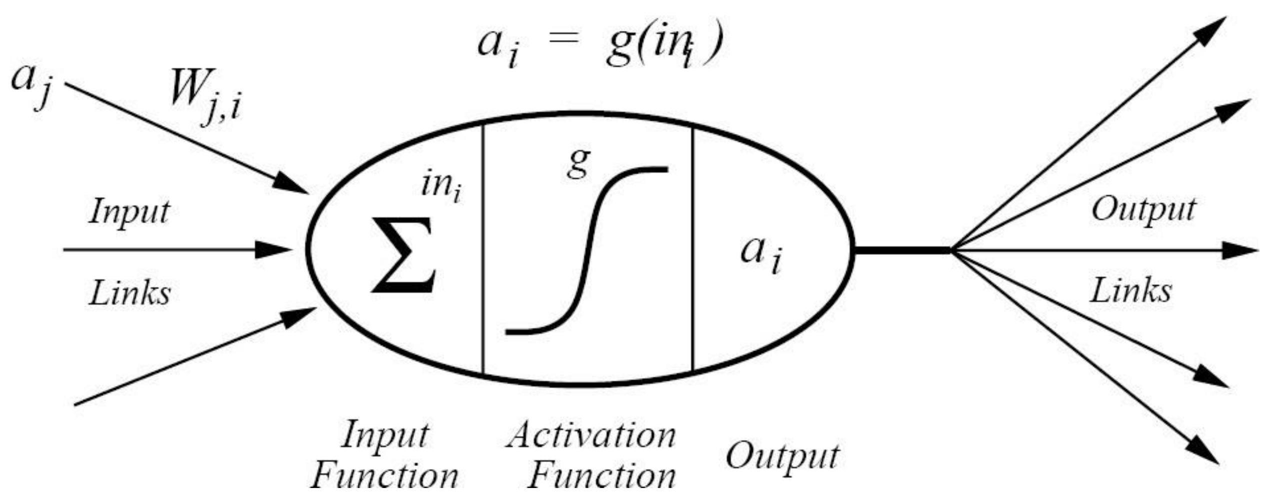


Fig 5 Simple computing elements

Fig 5 shows the structure of a simple computing element. The input function, in_i, is a linear component which can calculate the weighted sum of the input values of the unit. g is the activation function which is a non-linear part of the computing element which can get the final value a_i, served as the activation value of the unit, based on the transformation of the weighted sum.

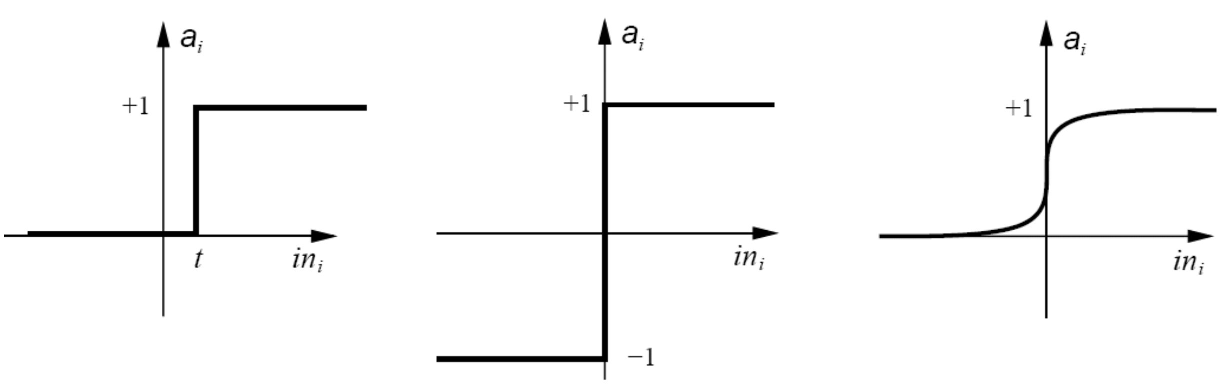
The total weighted input is the weights of the input activation multiples their corresponding weights:

$$in_i = \sum_j W_{j,i} a_j = \mathbf{W}_i \cdot \mathbf{a}_i$$

Then we can apply the activation function g to the result of the input function and get:

$$a_i \leftarrow g(in_i) \quad g\left(\sum_j W_{j,i} a_j\right)$$

There are different models of g to choose, the most common three are step, sign and sigmoid functions.



(a) Step function (b) Sign function (c) Sigmoid function

The mathematical formulas for each model are:

$$step_t(x) = \begin{cases} 1, & \text{if } x \geq t \\ 0, & \text{if } x < t \end{cases} \qquad sign(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \qquad sigmoid(x) = \frac{1}{1 + e^{-x}}$$

1.4 Neural Network Structures

There are two main types of neural network structure: feed-forward networks and recurrent networks.

In feed-forward networks, links are unidirectional which will not form any cycle, while in recurrent networks links can form any topologies without restriction. Fig 4 is a typical layered feed-forward network. Each unit only links to units in the next year. Links between same-layer units, backward to previous-layer units or pointed to cross-layer units do not exist.

Networks can be classified as perceptrons or multilayer feedforward networks based on the number of hidden units. For a mixed structure g, learning is process of tuning all the parameters for the given dataset in the training set.

For recurrent network, our brain could be a best example. It has a short-term memory as well as an internal state. Recurrent network can store in the activation levels of the units and are able to deal with more complicated agents with the internal state. However, the learning process will also be much more difficult and unstable.

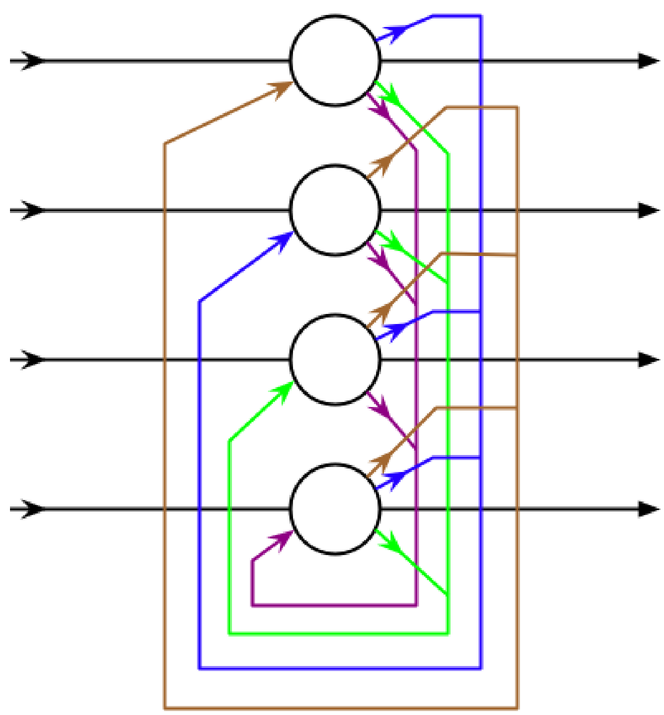


Fig 6 Hopfield Network

In Fig 6, we can use Hopfield Network as an example for recurrent network. We can see that all units are both input and output units. After training, there will be an associative memory which is an activation pattern correspond to the training dataset that most closely resembles the new stimulus generated after training.

1.5 Perceptron and Multilayer Feed-Forward

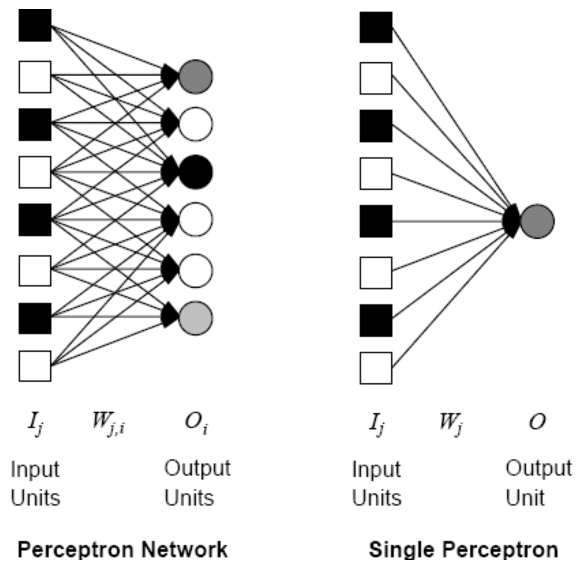


Fig 7 Perceptron

Fig 7 shows an example of perceptron. It is a single-layer and feed forward training without hidden layer. Weight from input unit j to output unit O is W_j. The activation of input unit is I_j and for the output unit is:

$$O = Step_0\left(\sum_j W_j I_j\right) = Step_0(\mathbf{W} \cdot \mathbf{I})$$

Where weight W_0 provides a threshold for the step function with $I_0 = -1$.

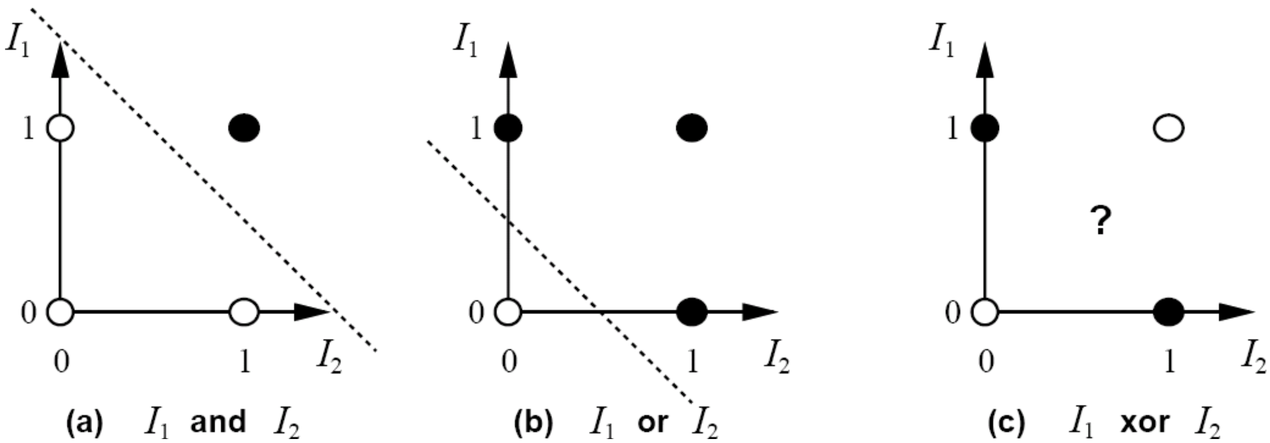


Fig 8 Linearly separable problem for perceptron

However, perceptron can only handle linearly separable functions. As we can see from Fig 8, if we have a problem like (c), we will not be able to use perceptron to find a solution. In such case, we will also introduce the multilayer Feed-Forward structure.

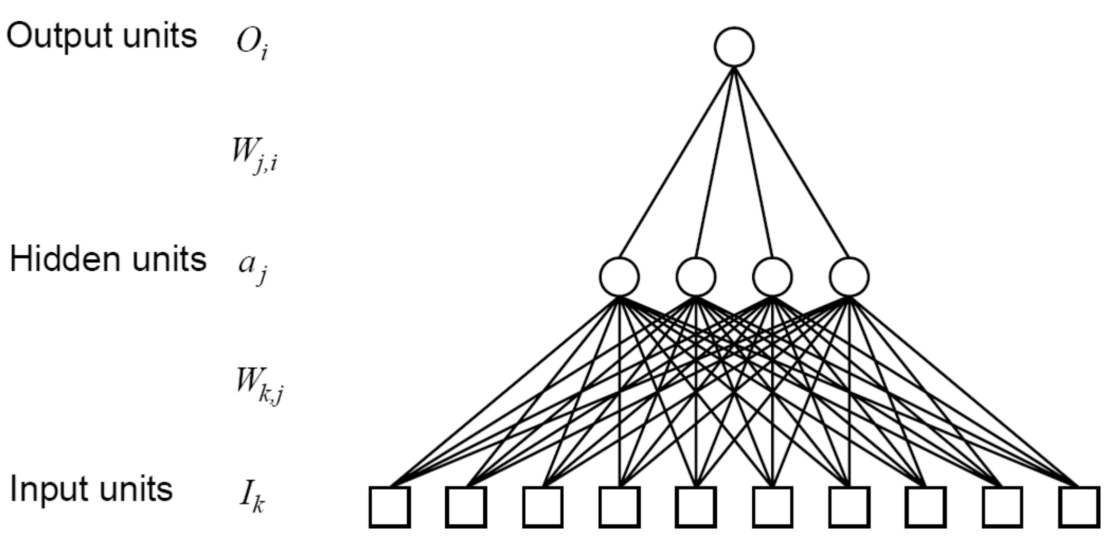


Fig 9 A 2-layer Feed-Forward Network

Though the learning process is similar to perceptron, when dealing with multilayer feed-forward structure, we will meet a problem which is to choose the right number of hidden units.

In such a model, we will evaluate the error and divide it among the contributing weights. The weight update rule at the output layer is to use the activation of the hidden unit's a_j as the input values and the gradient of the activation function.

When we are updating the W between input and hidden layers, we need to search for an equivalent error of the output nodes, which is also called the error back-propagation.

Some part of error in each of the output nodes is caused by the hidden node. The error values are divided according to the strength and propagated back to hidden layer.

1.6 Artificial Neural Network (ANN)

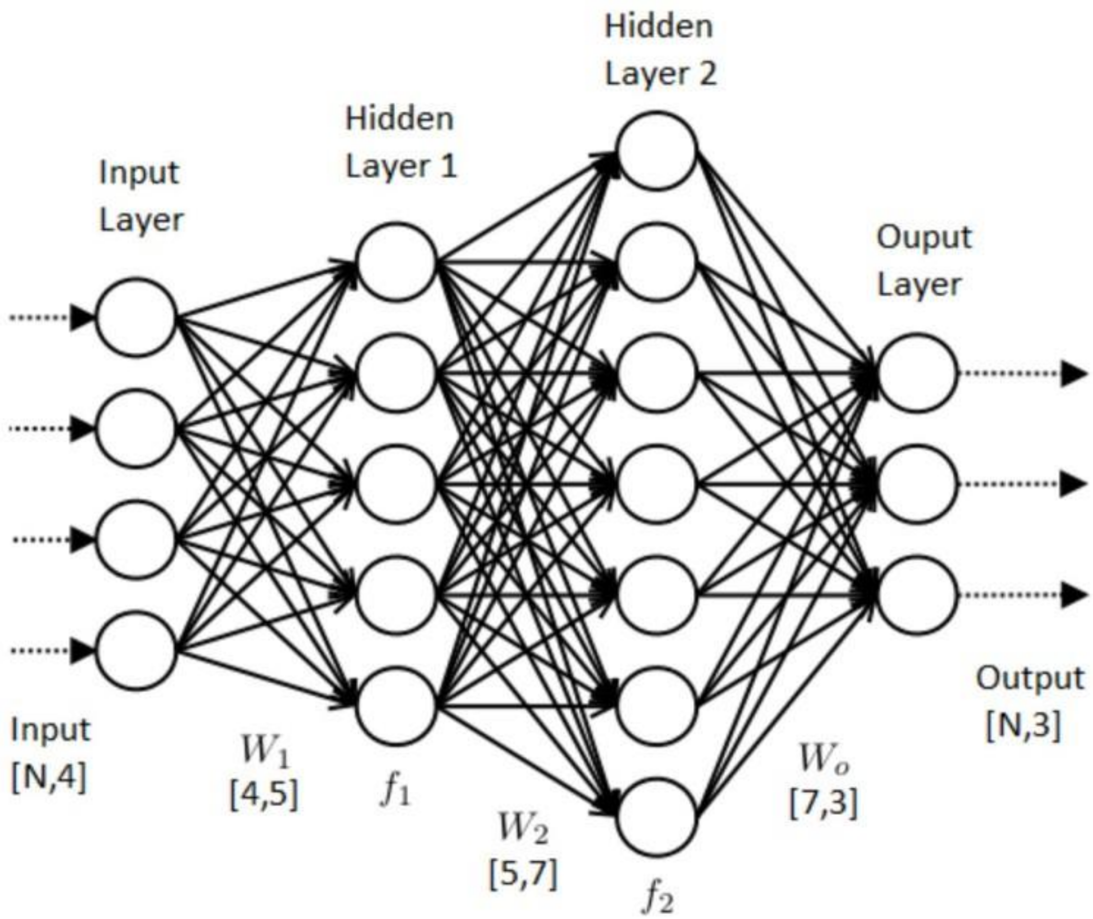


Fig 9 Example of ANN

With all the knowledge we have mentioned before, now we can take a look at a simple version of Artificial Neural Network (ANN).As we have mentioned before, ANN is inspired by the biological structure of human brain. Our biological neural networks enable us to do complex tasks and the “CPU” for our brain is the neuron. Similar to human brains, ANN also has many processors called as neurons which can provide some basic operations such as summing function. When we are training an ANN, information will be stored as the weights between layers.

There are three major areas that researchers are exploring with ANN:

A. Using ANN to simulate the biological human brains’ networks to gain biological understanding of human brain.
B. Using ANN to gain understanding of how to solve difficult problems that traditional AI methods can hardly be useful, such as the example of AlphaGo we have mentioned before.
C. Using ANN to solve real-world problems from different angles.

2. TensorFlow

2.1 Introduction to TensorFlow

Fig 10 TensorFlow

In my project, I have used a famous machine learning platform: TensorFlow. It is an open source project which is developed by Google Brain. TensorFlow is one of the biggest machine learning platforms which supports a great number of neural networks as well as other machine learning APIs and has been used by many machine learning developers. With its various APIs, developer can choose to build a neural network step by step or even with the help of published well-performed models.

2.2 Introduction to TensorFlow Lite

Similar to TensorFlow, TensorFlow Lite is a lightweight version of TensorFlow which is designed to be used in mobile or embedded devices. With TensorFlow Lite, we can run machine learning models on mobile devices with low latency, which is suitable for many tasks including regression, classification or any other tasks according to our own wiliness.

TensorFlow Lite supports both Android and iOS platform through C++ API and also provides a Java Wrapper for Android. Furthermore, TensorFlow Lite also supports Android Neural Networks API for hardware acceleration on those Android Devices which support this API. For those devices which do not have such API, TensorFlow Lite will use CPU for execution by default.

TensorFlow Lite is made up of a runtime on which neural network models can be ran, and a series of tools which help us prepare for the model to be used on mobile or embedded devices.

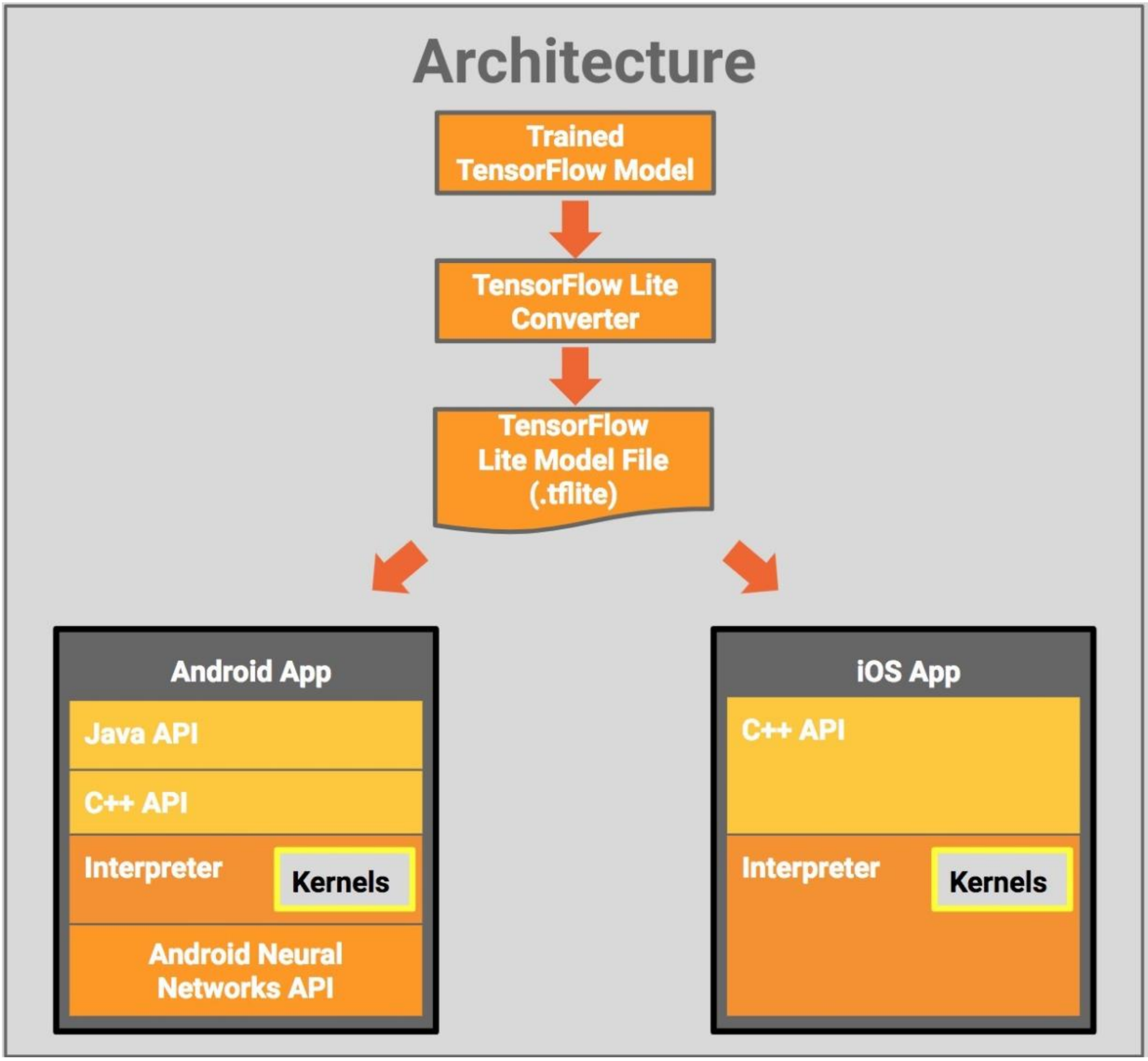


Fig 11 Architecture of TensorFlow Lite

Through the architecture of TensorFlow Lite, we can see that out previous model can also be deployed to mobile devices. However, this might not always work since some essential functions of your neural network model might ont be supported by mobile devices or TensorFlow Lite.

3. Image Processing

3.1 Introduction to Image Processing

In computer vision, image processing represents an implementation to perform some operations on images which are intended to enhance the images or to search for some meaningful information from the images. In some extent, image processing is also one kind of signal processing where we use images as input and some results or middle-layer weights as output. Primarily, image processing will have the following three procedures:

A. Import the images with the help of some tools.

B. Preprocess and analyze the images.

C. Output the designated result according to the evaluation for the images from B.

When we are talking about image processing, actually there are two types of image processing: analogue and digital. However, with the rapid development of smart devices, nowadays most of the people are using digital images in their daily life. In this report, we will mainly focus on digital image processing as well.

Digital image processing can help us modify the digital images based on our own criteria by using computers. Since raw images might contain deficiencies, in order to get the original information from those images, we have to go through the following phases of processing.

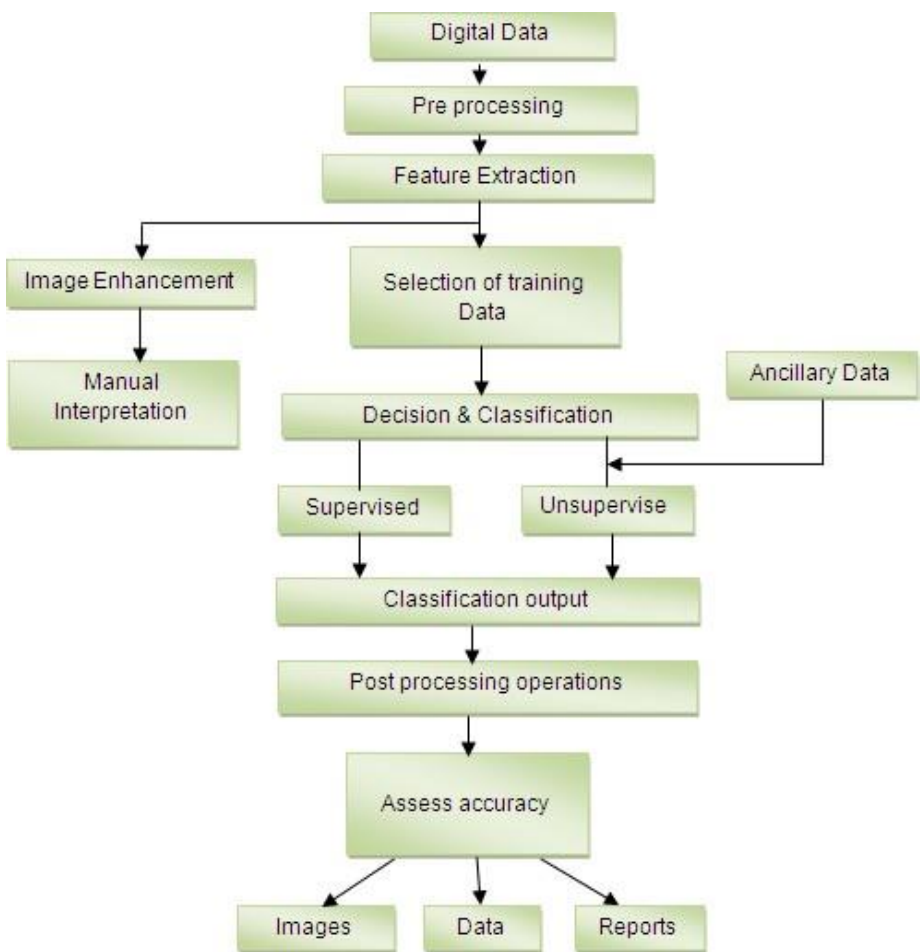


Fig 12 Flow Chart of Digital Image Processing

BACKGROUND

As we can see from Fig 12, the basic three phases that all kinds of images have to undergo are Pre-processing, enhancement and information extraction. The purpose of image processing can also be divided into 5 groups:

A. Visualization, which is to detect objects that were not visible inside the images.

B. Image restoration, which is to improve the quality of images.

C. Image retrieval, which is to search for some specific images.

D. Pattern measurement, which is to measure objects in images.

E. Image recognition, which is to recognize objects in images.

3.2 Feature Detection

In computer vision, feature detection could be used to represent the methods for calculating abstractions of information from the given image as well as the process of making decisions at every point of the given image to see whether there is a designated image feature or not.

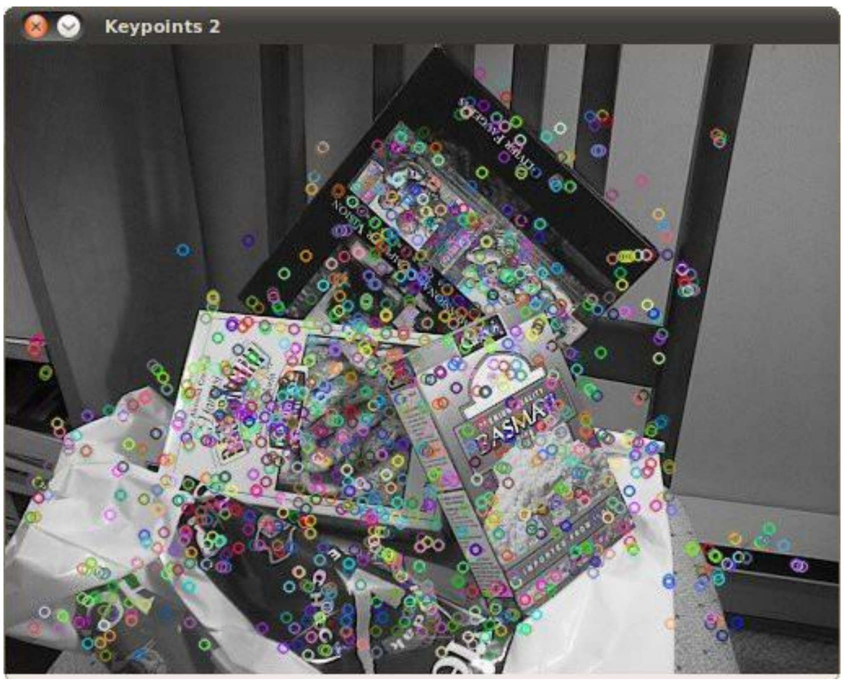


Fig 13 Example of feature detection

However, a ubiquitous and accurate definition of the so-called “feature” would be hard to define since the exact definition is highly related to the specific problem that people are trying to solve. We could think feature as some parts of an image which interest us the most. In many cases, since features are used since the very beginning of the whole experiment by evaluating images pixel by pixel, a bad representation of features will make it impossible for us to get a good result. Besides, repeatability should also be one of the requirements of a feature detector: same features should be detected from different images.

Since many different image processing algorithms are relying on the result of feature detection to continue, a vast number of feature detection methods have been explored. In such case, the types of features as well as the complexity are varied from each other.

As for the main types of image features, the following four features should be the most famous and popular kinds: Edges, Corners, Blobs and Ridges.

- ***Edges***

Edges represent for points of the boundary or edge between two regions. Theoretically, an edge can actually be of arbitrary shape even with junctions. Practically, edges are usually defined as group of points which show a strong gradient magnitude in an image. Some popular algorithms

Will then link points with high gradient together so that a more intuitive version of edge can be formed. Usually, some restrictions such as gradient value, shape and smoothness will also be adopted by these algorithms.

Fig 14 Example of Edge Detection

- ***Corners***

Another common feature is corner. Corner (or interest points) refer to features in an image which are point-like with a local two-dimensional structure. Firstly, the algorithms were seeking for edge in the images and then evaluated the edges to find a point where the direction sudden changed. Later, with the evolvement of the algorithms, the edge detection part was no longer needed. Instead, the algorithms were trying to search for high levels of curvature in the image gradient. Since then, we were able to detect corners in an image where we would not be able to find a corner in the traditional sense.

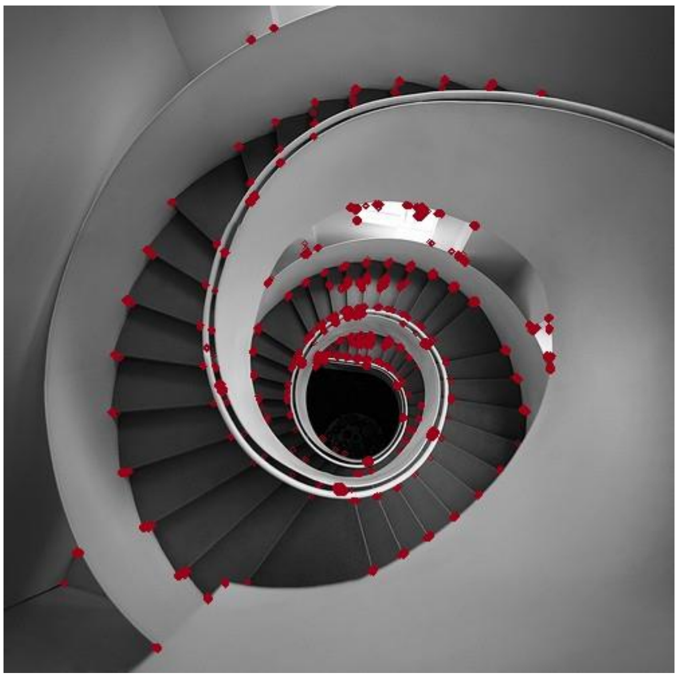


Fig 15 Example of Corner Detection

- ***Blobs***

As for blobs, it can provide an explanation for image structure with respect to regions instead of point-like oriented like corners, which can give us a new angle to examine our definition of image. In some cases, blob detectors may also contain a preferred point. If so, these detectors may also be thought as corner operators. Usually, blob detector examines areas which are too smooth to be found for an interest point detector.

Imagine shrinking the image and then execute interest points detection. The detector will find those points which are sharp in the new image. However, these sharp points may correspond to smooth points in the original image. In such case, the difference we are talking about between these two detectors will be somehow vague to define. More clearly, the difference can actually be eliminated by careful remediation with a proper scale.

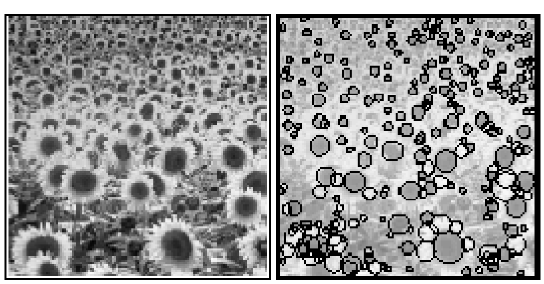


Fig 16 Example of Blob Detection

- ***Ridges***

A ridge descriptor calculated from a grey-style image can be treated as a generalization of medial axis. Practically, a ridge can be treated as a single dimensional curve which is related to the axis of symmetry. Additionally, it also has a local ridge width correlated to each ridge point. Nevertheless, to find out the ridge features from common grey-style images would be much more difficult than to extract edge, corner, or blob features.

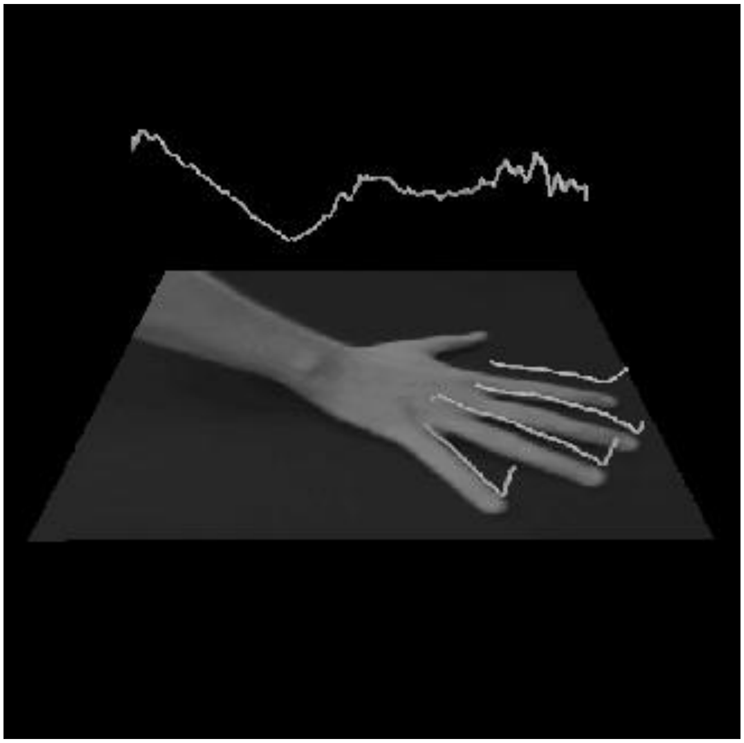


Fig 17 Example of Ridge Detection

- ***Common feature detectors***

Common feature detectors and their classification:			
Feature detector	Edge	Corner	Blob
Canny	X		
Sobel	X		
Kayyali	X		
Harris & Stephens / Plessey / Shi–Tomasi	X	X	
SUSAN	X	X	
Shi & Tomasi		X	
Level curve curvature		X	
FAST		X	X
Laplacian of Gaussian		X	X
Difference of Gaussians		X	X
Determinant of Hessian		X	X
MSER			X
PCBR			X
Grey-level blobs			X

4. Convolutional Neural Network

4.1 Introduction to Convolutional Neural Network

Similar to Artificial Neural Network, Convolutional Neural Network was inspired by animal vision system. As a feed-forward neural network, convolutional neural network is often used as a tool for image processing since its architecture was designed for manipulating inputs like images. Many famous computer vision models, such as ResNet and Inception, are also built based on the structure of convolutional neural network.

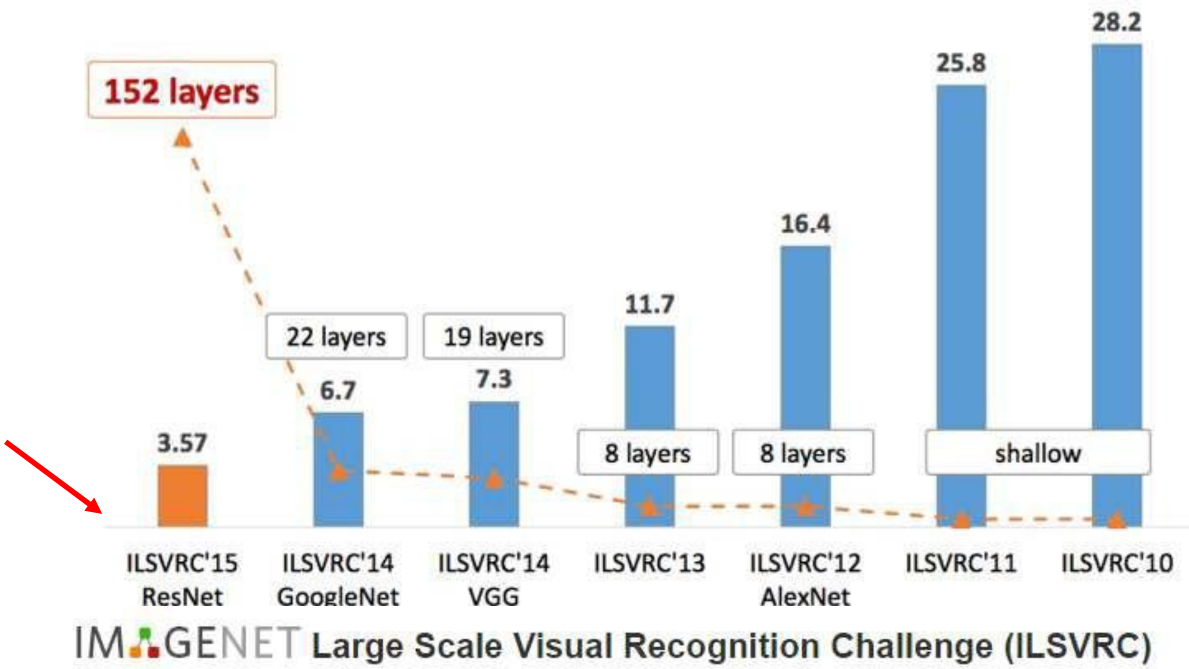


Fig 18 Models performance for ImageNet challenge

However, convolutional neural network is not exactly the same compare to the artificial neural network we have mentioned before. One significant difference between these two neural networks is the number of dimensions the neurons can form within one layer, while the number of dimensions in a convolutional neural network is three (width, height, depth) instead of two in the artificial neural network. Besides, contrast to the neurons in artificial neural network, the neurons in convolutional neural network do not connect all neurons in the previous layer. Instead, each neuron only connects to a small region of neurons. Such kinds of structure make it possible to handle inputs like images since fully connected structure would requires a huge amount of time and resources to compute the results which will result in a low efficiency and slow down the development of computer vision.

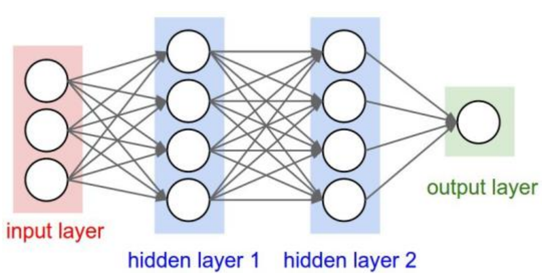


Fig 18A A regular 3-laye neural network

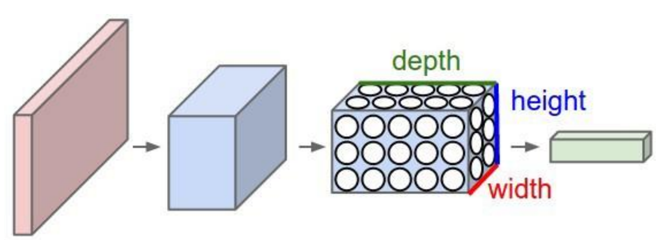


Fig 18B A CNN with its neurons in 3 dimensions

- ***Overall architecture***

Convolutional neural networks are sequences of layers where each layer transforms activations to other layers based on differentiable function. Its structure is built from three main types of layers: Convolutional Layer, Pooling Layer and Fully-Connected Layer.

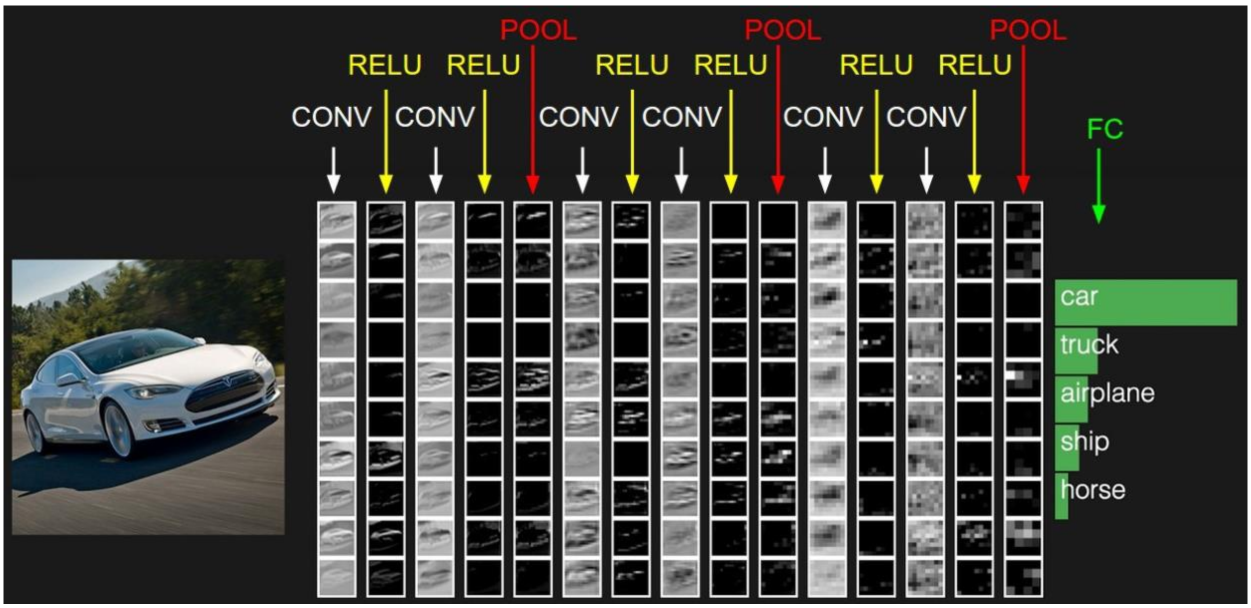


Fig 19 A regular convolutional neural network for CIFAR-10

Take the Fig 19 as an example, we have:

1. Input: a picture with width and height for 32 and with a color channels R,G,B.

2. Convolutional Layer: Calculate the output of dot product between

weights and the local regions where the neurons connected to in the input.

3. RELU layer: Apply the activation function, such as max(x,0). The size of the volume remains unchanged.

4. POOL layer: Perform down-sampling through the width and height dimensions. The size of volume will change to a smaller one.

5. Fully-Connected layer: Calculate the class scores which will have a size of 1*1*10. The 10 numbers represent the classification score for the 10 classes in CIFAR-10.

Through this structure, convolutional neural networks transform the input image from pixel values to final classification scores layer by layer. To sum up, a convolutional neural network would be the most intuitive case to show a series of layers which transform images into the output classification scores. Each layer would accept as well as output a 3D volume of data and can perform their functions with or without parameter and additional hyperparameters.

- ***Convolution Layer***

Convolution layer is the essential part of a convolutional neural network and the parameter in this kind of layer is the learnable kernels.

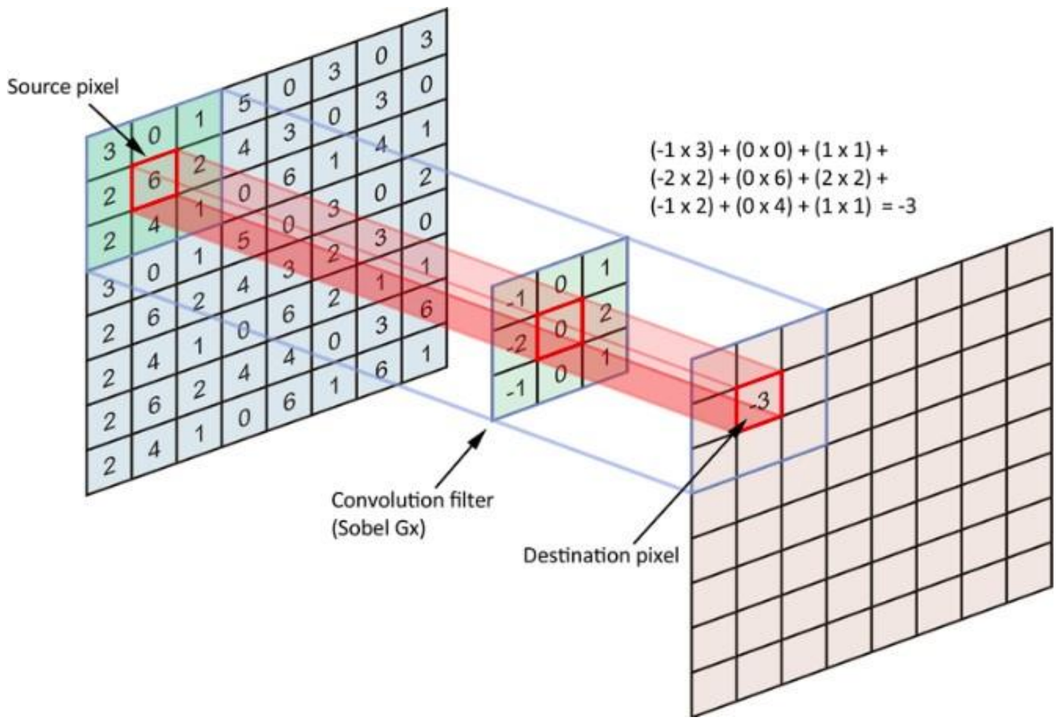


Fig 20 A typucal convolutional layer

Parameters of convolutional layer are composed of a series of learnable filters where each filter is small along height and width. However, it can extend to the full depth of the input volume. During the forward pass, each filter can be convolved with respect to the height and width of input. In such case, a two-dimensional activation map can be produced.

After then network can see some visual features such as edges as we have mentioned before, it will start to learn through the process. Eventually, we can have the whole filters in each convolutional layer.

In some cases, when we need to handle inputs with high-dimensional inputs, such as images, it is impossible to create a fully-connected structure. The receptive field of the neuron, also known as the filter size, can be introduced. The asymmetry in how we think the width and height and hoe wo treat the depth dimension needs to be emphasized as well, since the connections are local in space but full along depth.

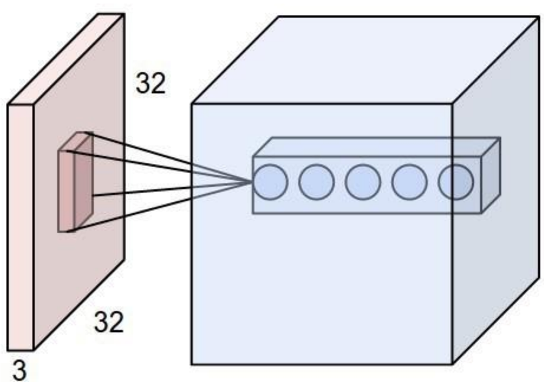


Fig 21A Example convolutional neural network

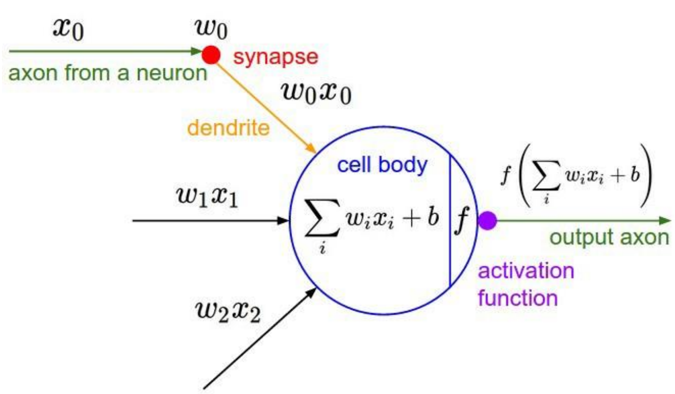


Fig 21B Example of neurons computation

- ***Pooling layers***

Usually we will insert a pooling layer between two convolutional layers in a standard convolutional neural network structure. The usage of pooling layer is to rapidly reduce the size of data in the middle layers so that we can reduce the amounts of calculations which results from a smaller number of parameters and weights of each neuron. Through this approach, the problem of overfitting can also be efficiently controlled.

The pooling layers can work independently with the MAX function, which can form the well-known max-pooling layers. One of the most popular instances of a pooling layer is layer with filters of size 2 * 2. This kind of pooling layer with stride 2. The depth, height as well as width of input will be reduced to the half of their original size, and thus the total size of output can be reduced to just 25% of the original size.

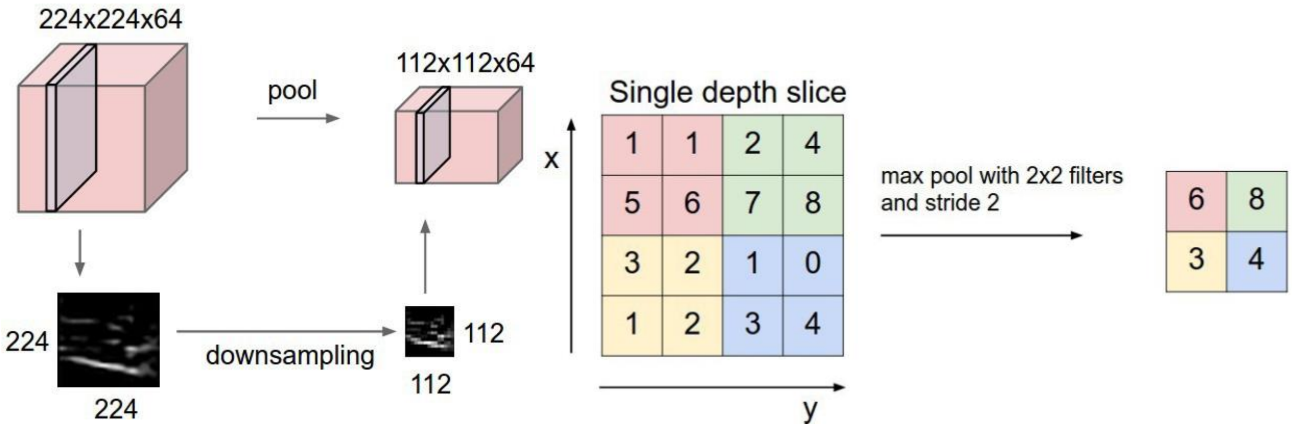


Fig 22 Example of max pooling

Take Fig 23 as example, pooling layer downsamples the input in both directions. In the left example, input with size 224*224*64 is transformed to output with size 112*112*64 after pooling layer. In the right, the most common pooling operation, max pooling, is showed with concrete examples. Some people do not want to use pooling and have proposed some ideas to get rid of pooling layers. This is also feasible especially for some generative models such as generative adversarial networks and variational autoencoders.

- ***Normalization Layers***

Different types of normalization layers have been incorporated into convolutional neural network models. Nevertheless, these layers are gradually being discard because of the small contribution to the final result.

- ***Fully-connected Layers***

Usually we use this term to represent the second last layer which has total connections to all activations and can compute with a matrix multiplication with a bias offset. The results of Fully-Connected layers will be passed to the final layer: Softmax layer where we can get a classification scores based on the numbers.

- ***Softmax Layer (Output Layer)***

In fact, softmax Layer is also fully-connected. After receiving the inputs from the second-last layer, softmax layer will compute a 1*n matrix which stores the classification scores for each of the n classes.

The mathematical function of softmax in this case is:

$$h_\theta\left(x^{(i)}\right) = \begin{bmatrix} p\left(y^{(i)} = 1 \middle| x^{(i)}; \theta\right) \\ p\left(y^{(i)} = 2 \middle| x^{(i)}; \theta\right) \\ \vdots \\ p\left(y^{(i)} = k \middle| x^{(i)}; \theta\right) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} \theta_1^T x^{(i)} \\ \theta_2^T x^{(i)} \\ \vdots \\ \theta_k^T x^{(i)} \end{bmatrix}$$

The cost function of softmax is:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)} = j\} log \frac{\theta_j^T x^{(i)}}{\sum_{l=1}^{k} \theta_l^T x^{(i)}}\right]$$

Finally, we can get the mathematical formula for softmax:

$$P_t(a) = \frac{e^{\frac{q_t(a)}{T}}}{\sum_{i=1}^{n} e^{\frac{q_t(i)}{T}}}$$

5. Depthwise Convolutions

5.1 Introduction to Depth Convolutions

Machine Learning algorithms are extraordinary at solving many complex problems, including Image Classification problems, Object Detection problems, Speech Recognition problems, Language Translation problems, and many more applications.

However, machine learning is extremely computation intensive—it's generally only viable to those powerful general-purpose GPUs. Unfortunately, mobile devices might have very limited computation capacity; hence, most architectures that have been very successful on desktop computers and servers cannot be directly deployed to mobile devices.

Given that, there are indeed some techniques which can make machine learning models efficient enough for mobile devices, and one of them is called "Depthwise Convolution".

Depthwise convolution builds the foundation of MobileNet models, which will be introduced later. Depthwise convolution is very similar to the original one but is different in some places which make it much more efficient than the original one.

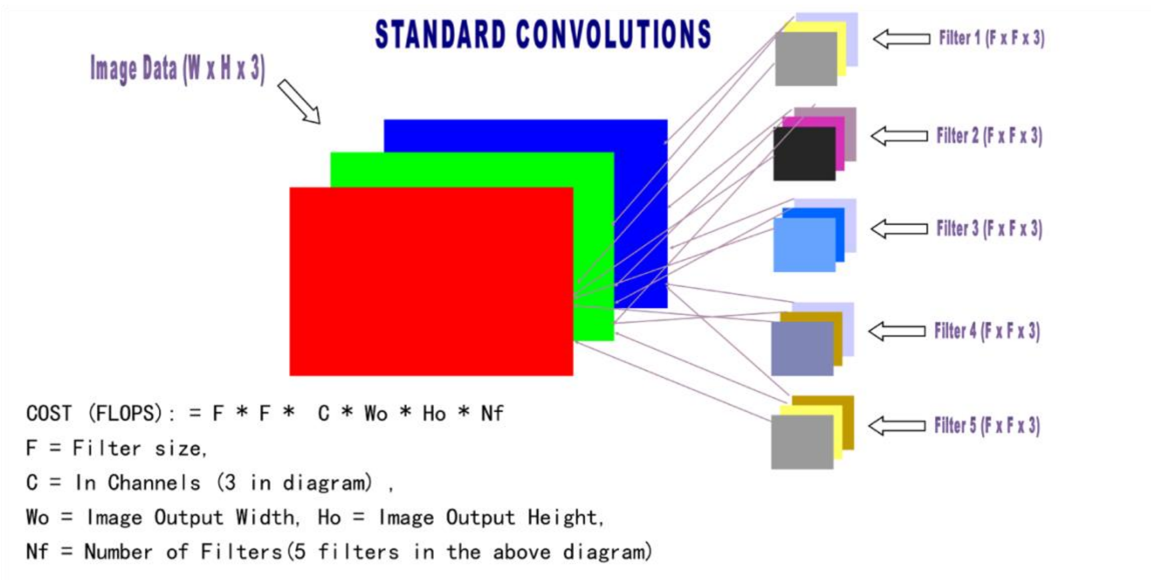


Fig 23 Example of standard convolution

In standard convolution, for a 3 *W * H image, we use five filters to perform convolution on the image. The size of filter is F * F, and thus the filter size is equal to 3 * F * F since our image has 3 channels.

The convolution result would be 5 * Ho * Wo. In such case, the computational cost would be: 5 * Ho * Wo * 3* F * F.

Thus, a 3 * 3 convolution with 128 filters, operating on an input of size 64 * 112 * 112, with both of the size of padding and size of stride being 1 will have a cost of: 3 * 3 * 128 * 64 * 112 * 112 * 1 * 1 = 924 Million Multiply Adds.

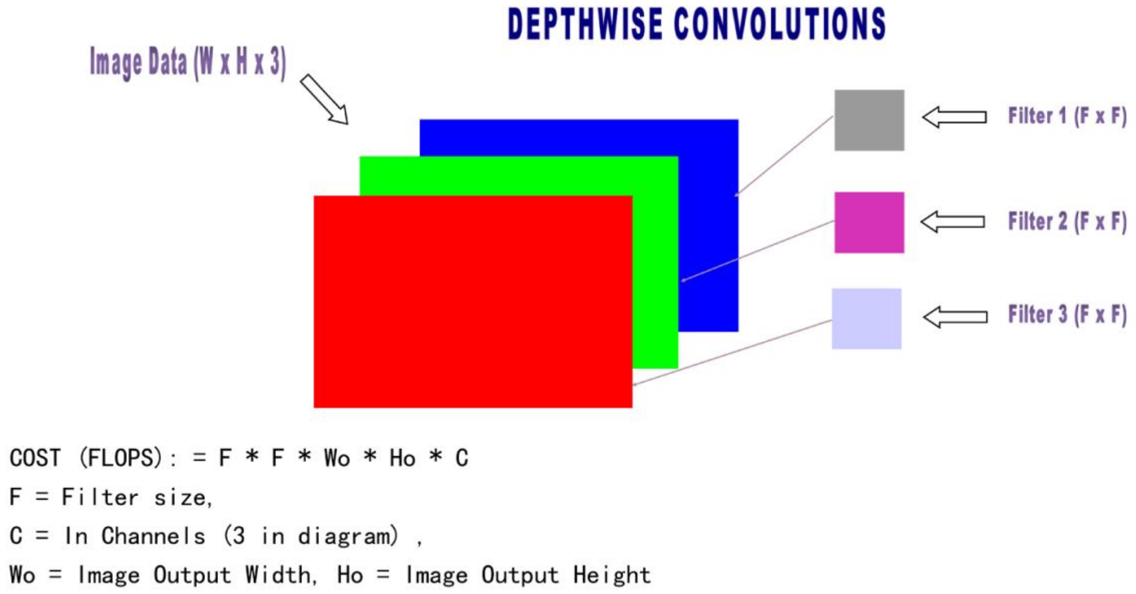


Fig 24 Example of Depthwise convolution

Compared to previous standard convolution, each filter of depthwise convolution will only operate on a single channel, and the number of image input channels is also equal to the number of filters.

These two conditions show that every filter will apply on each channel separately. Thus, an image with n channels will need n filters, each with an actual size of F * F. For example, 3 * 2 * 2 filters in the original one will be changed to 2 * 2 in depthwise convolution.

In depthwise convolutions:

FILTER DIM = F * F

Output DIM = C * Ho * Wo

Cost = Image Output DIM * FILTER DIM

The formula for depthwise convolutions is:

C * Ho * Wo * F * F

Thus, a 3 x 3 depthwise convolution operating on a input of size 128 * 112 * 112, with both of the size of padding and size of stride being 1 will have a cost of: 128 * 112 * 112 * 3 * 3 * 1 * 1 = 14.45 Million Multiply Adds

For the same input size, padding and stride, the cost of the original one is 924 million multiply adds, while the cost of depthwise convolution is 14.45 million multiply adds. The number of cost decreases a lot.

BACKGROUND

Methodology

1. Dataset

1.1 Dataset 22

Considering about that there is no public and suitable dataset for my task, I have built my own dataset by crawling pets' images from Google, Bing and Baidu. Right now, my dataset consists of about 18k images for 22 kinds of dogs and cats. All of the images have been verified by me and I will continue to add meaningful species into my dataset.

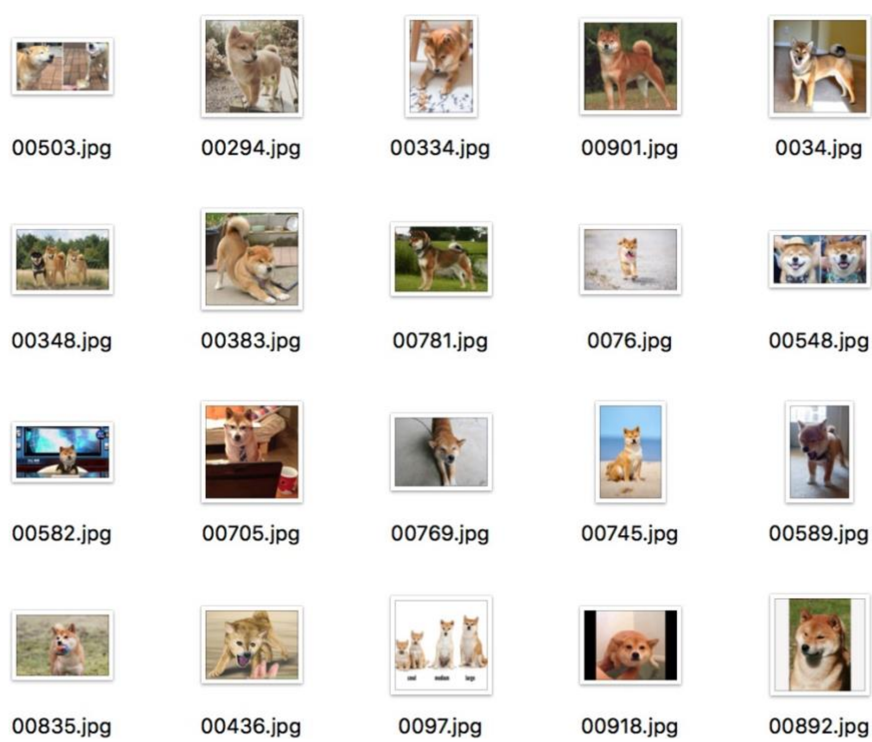


Fig 25 Example of my dataset

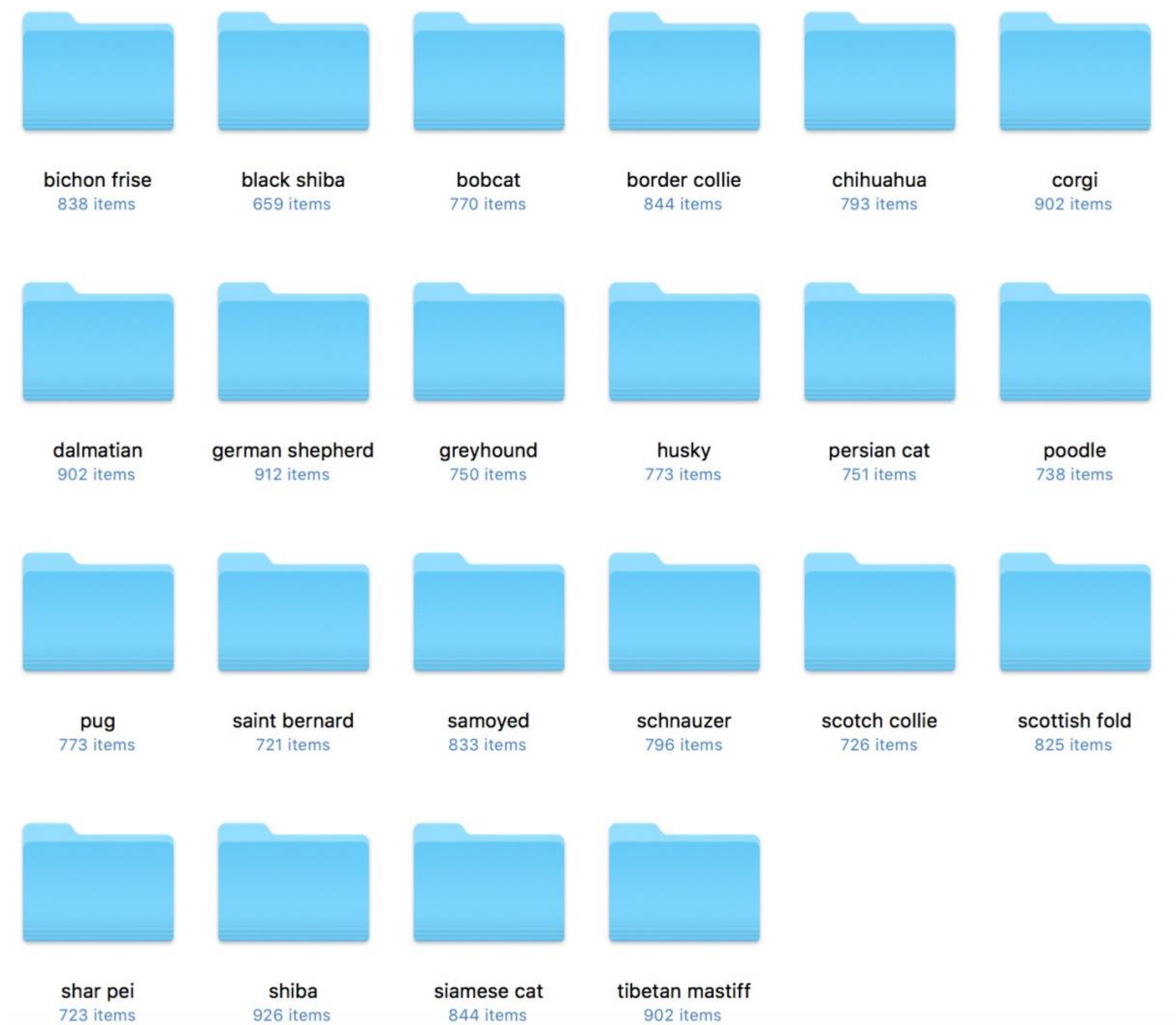


Fig 26 Example of my dataset

There are 18 kinds of dogs and 4 kinds of cats.

Large dog (7): German Shepherd, Greyhound, Husky, Saint Bernard, Samoyed, Scotch Collie, Tibetan Mastiff.

Mid-size dog (6): (black) Shiba, Border Collie, Dalmatian, Shar Pei, Pug.

Small dog (5): Bichon frise, Chihuahua, Corgi, Poodle, Schnauzer.

Cat (4): Bobcat, Persian Cat, Scottish Fold, Siamese Cat.

METHODOLOGY

1.2 Stanford Dogs Dataset

The Stanford Dogs dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization.

Contents of this dataset:

Number of categories: 120
Number of images: 20,580
Annotations: Class labels, Bounding boxes

Baseline Results

This section contains baseline results on two tasks:

1. Mean Accuracy:
   The number of training images per class is varied from 1 to 100.
2. Comparison of Accuracy per Class
   The accuracy of each class is compared for 15 and 100 training images per class.

Experimental Setting

All of the experiments use image regions from the bounding box only for both training and testing. The remaining parameters are set to the following values:

Type of SIFT descriptors: Grayscale

SIFT patch sizes: 8, 10, 14, 18, 22, 26, 30

SIFT grid spacing: 4 pixels

Spatial pyramid: 1*1+2*2+4*4 (3 levels)

Dictionary Size: 256

Kernel: histogram intersection kernel

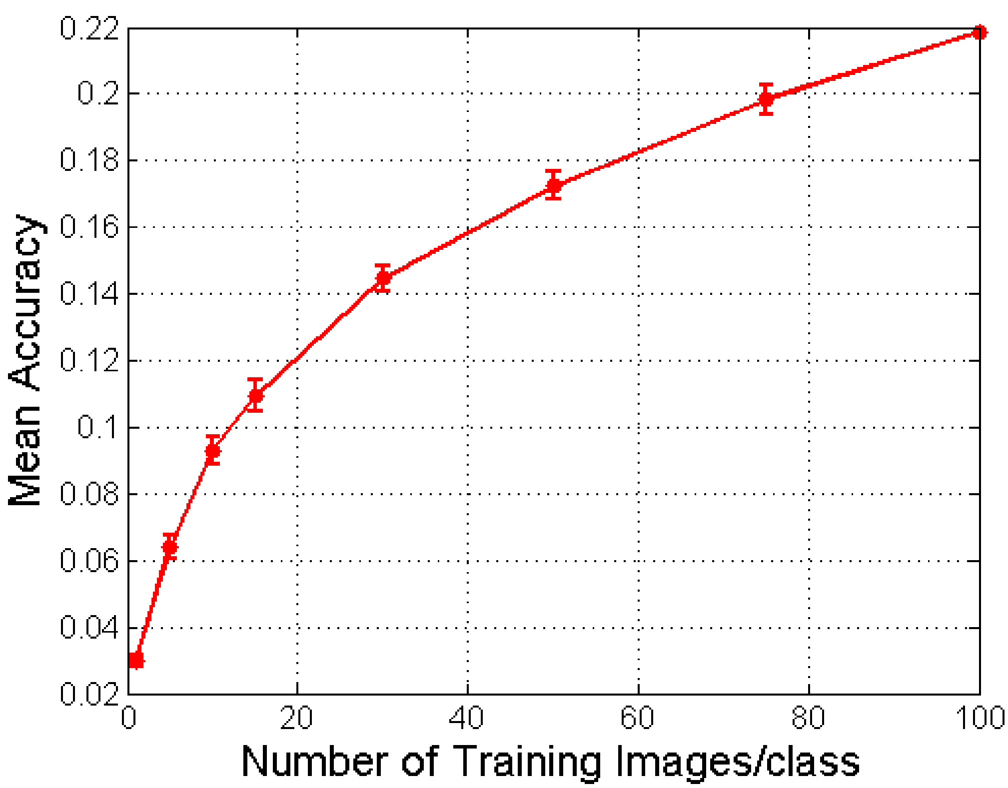


Fig 27 Example of Stanford dataset

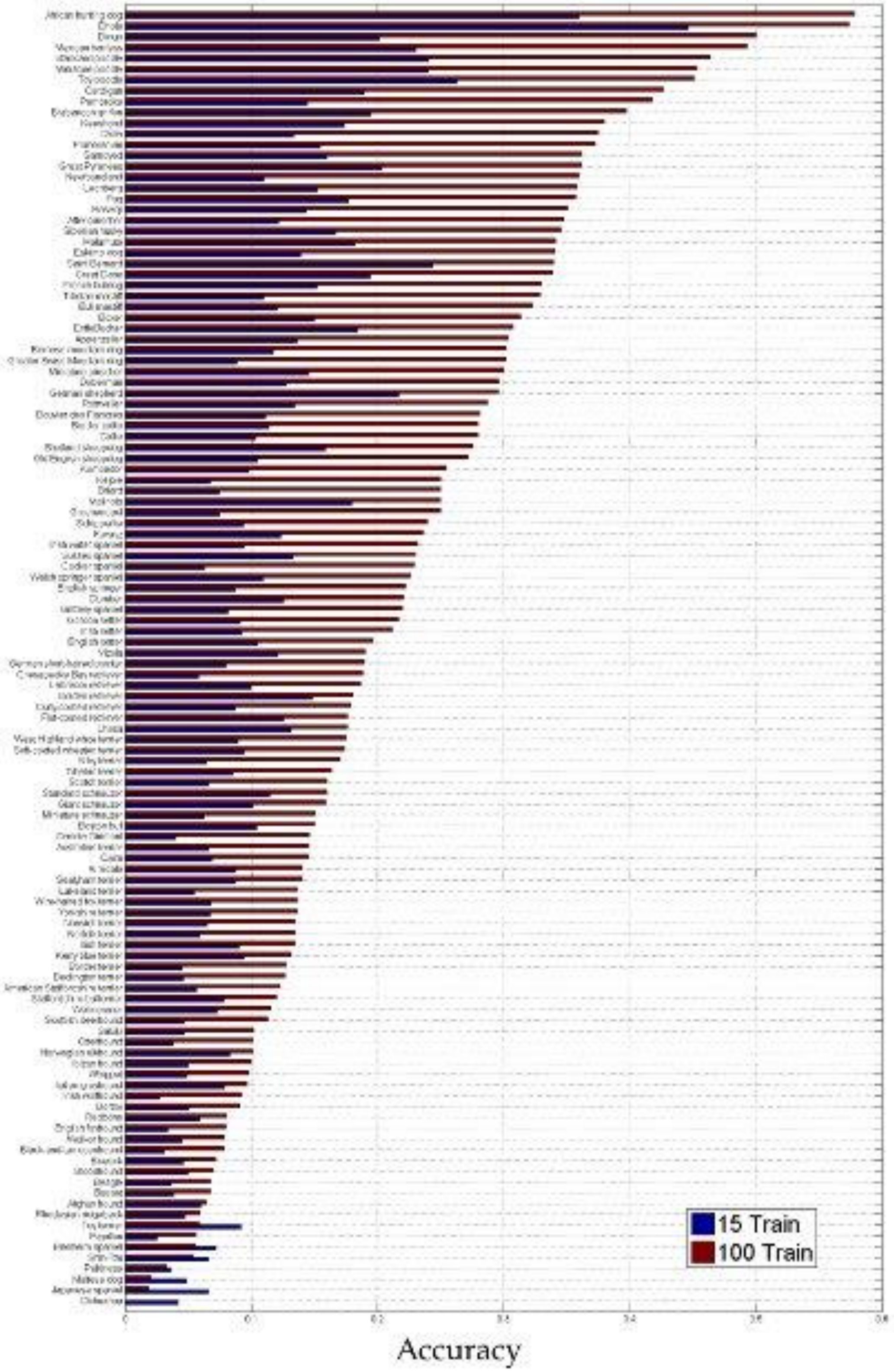


Fig 28 Example of Stanford dataset

METHODOLOGY

1.3 Dataset 133

In the last semester, our dataset only contains 22 classes of dogs and cats, which is kind of small compared to the real number of species of dogs and cats.

In this semester, with the help of Stanford Dogs Dataset [Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, Li Fei-Fei], we are able to increase our dataset to 133 classes which can cover most of the common species of dogs in our daily life.

Contents of this dataset:

Number of categories: 133

Number of images: 23,438

Each of the category contains roughly about 150~200 images. Compared to our original dataset, the number of categories increases (22 -> 133) while the number of images per category drops (~800 -> ~200). The reason is because our goal is to support more kinds of dogs, but such a large number of valid images would be hard to collect. Considering about that, we decrease the number of images per category in this semester.

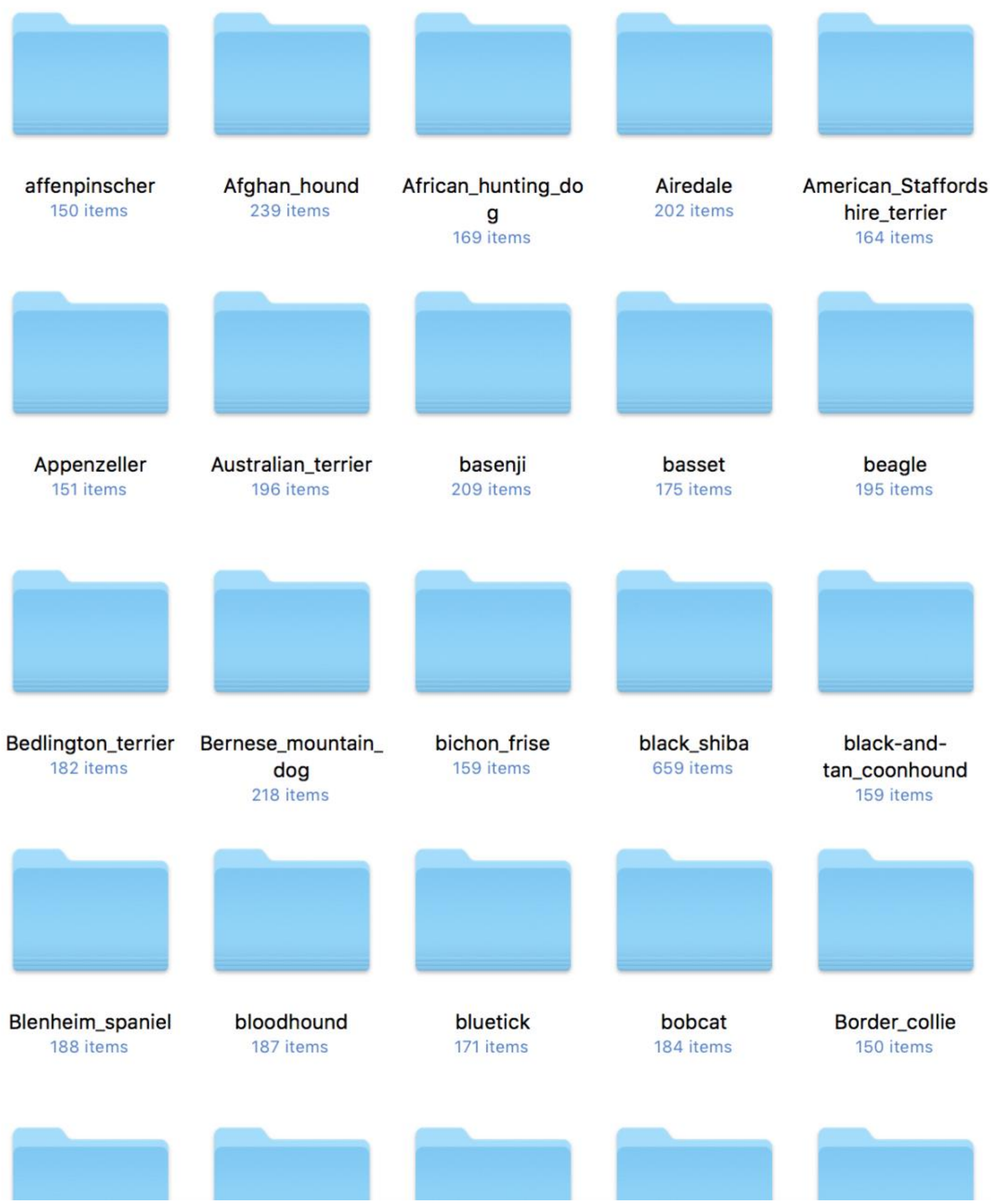


Fig 29 Example of New dataset

Fig 30 Example of Stanford dataset

METHODOLOGY

There are **129** kinds of dogs and **4** kinds of cats.

Large dog (31):

German Shepherd, Greyhound, Husky, Saint Bernard, Samoyed, Scotch Collie, Tibetan Mastiff, Afghan_hound, borzoi, chow, collie, Doberman, flat-coated_retriever, giant_schnauzer, golden_retriever, Gordon_setter, Great_Pyrenees, Ibizan_hound, Irish_setter, Irish_wolfhound, komondor, kuvasz, Leonberg, malamute, Newfoundland, redbone, Rhodesian_ridgeback, Saluki, Scottish_deerhound, Shetland_sheepdog, Sussex_spaniel.

Mid-size dog (53):

(black) Shiba, Border Collie, Dalmatian, Shar Pei, Pug, basenji, basset, American_Staffordshire_terrier, Bernese_mountain_dog, black-and-tan_coonhound, bloodhound, bluetick, Bouvier_des_Flandres, boxer, briard, Brittany_spaniel, bull_mastiff, Chesapeake_Bay_retriever, clumber, cocker_spaniel, curly-coated_retriever, dhole, dingo, English_foxhound, English_setter, English_springer, EntleBucher, German_short-haired_pointer, Great_Dane, Greater_Swiss_Mountain_dog, groenendael, Irish_terrier, Irish_water_spaniel, Italian_greyhound, kelpie, Kerry_blue_terrier, Labrador_retriever, malinois, Mexican_hairless, Norwegian_elkhound, miniature_schnauzer, otterhound, Rottweiler, schipperke, Staffordshire_bullterrier, standard_poodle, standard_schnauzer, Tibetan_terrier, vizsla, Walker_hound, Weimaraner, Welsh_springer_spaniel.

Small dog (45):

Bichon frise, Chihuahua, Corgi, Poodle, Schnauzer, affenpinscher, African_hunting_dog, Airedale, Appenzeller, Australian_terrier, beagle, Bedlington_terrier, Blenheim_spaniel, Border_terrier, Boston_bull, Brabancon_griffon, cairn, Cardigan, Dandie_Dinmont, French_bulldog, Japanese_spaniel, keeshond, Lakeland_terrier, Lhasa, Maltese_dog, miniature_pinscher, miniature_poodle, Norfolk_terrier, Norwich_terrier, Old_English_sheepdog, papillon, Pekinese, Pembroke, Pomeranian, Scotch_terrier, Sealyham_terrier, Shih-Tzu, silky_terrier, soft-coated_wheaten_terrier, toy_poodle, toy_terrier, West_Highland_white_terrier, whippet, wire-haired_fox_terrier, Yorkshire_terrier.

Cat (4):

Bobcat, Persian Cat, Scottish Fold, Siamese Cat.

Compared to previous dataset, now we have much more similar species in the new dataset.

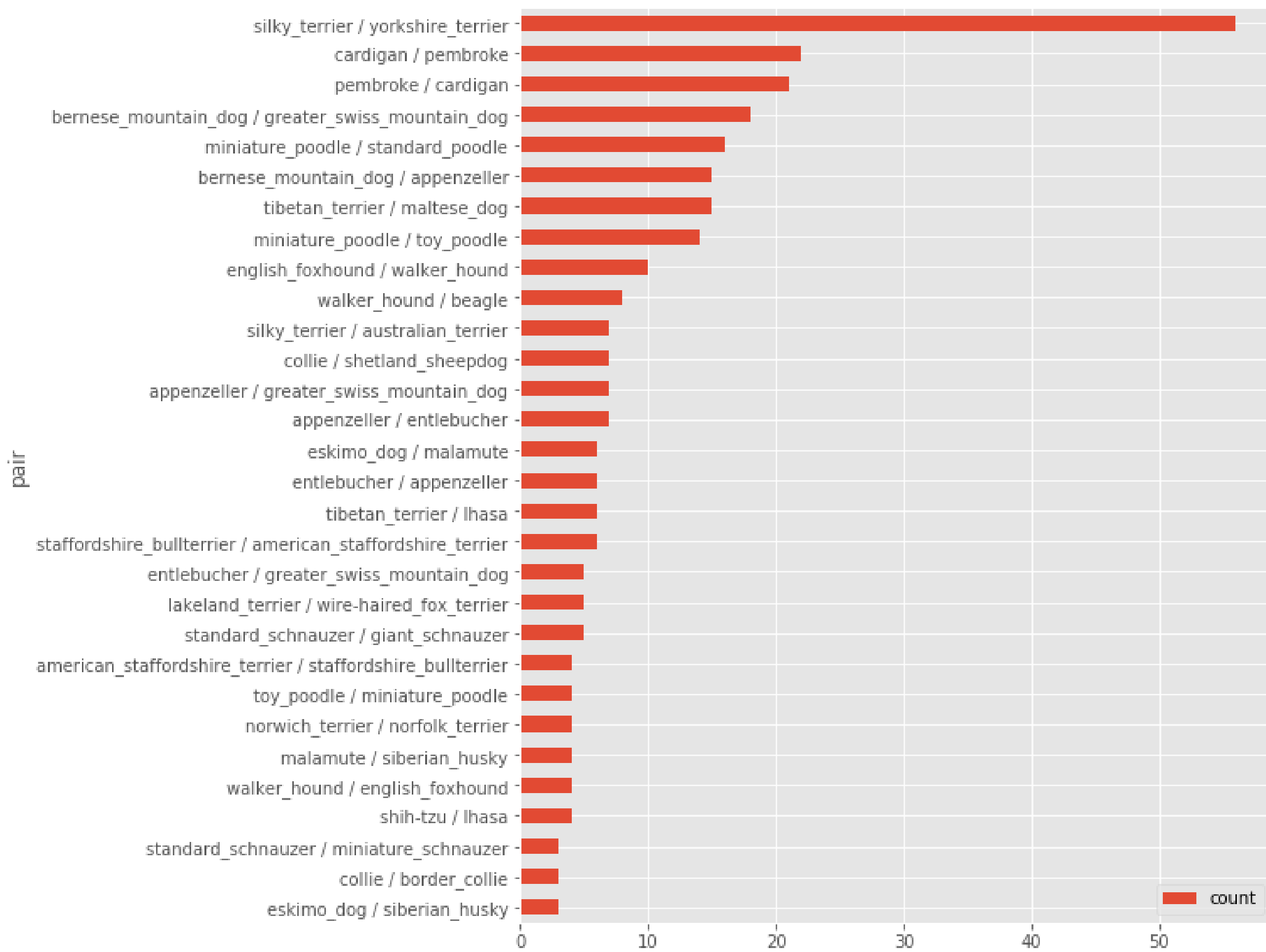


Fig 31 Misleading Pairs in the New Dataset

2. Inception

Inception is a model published by Google which is powerful for image classification

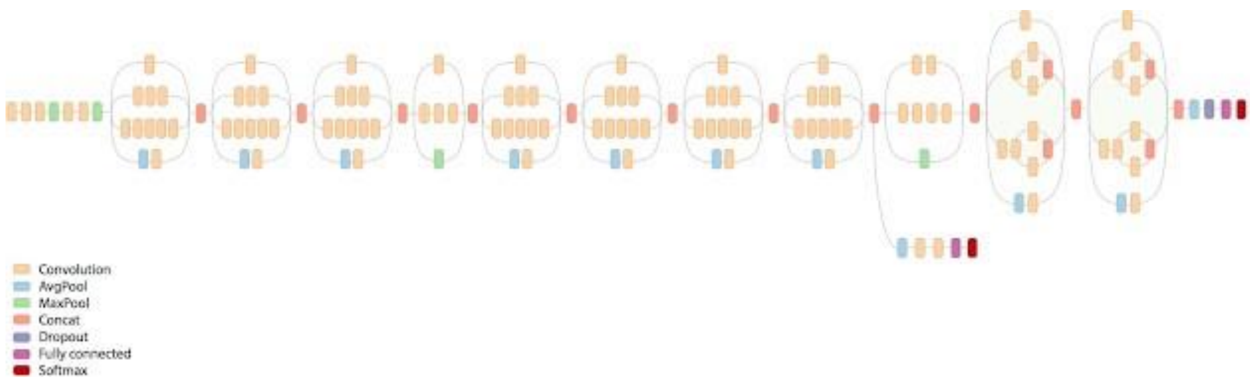


Fig 32 Sturcture of Inceotion V3

There are many versions of Inception, including the most popular one Inception-V3 which achives 0.78 for the top-1 classification accuracy for ImageNet, as well as Inception-ResNet-V2 which achives 0.804 for the top-1 classification accuracy for ImageNet, and others.

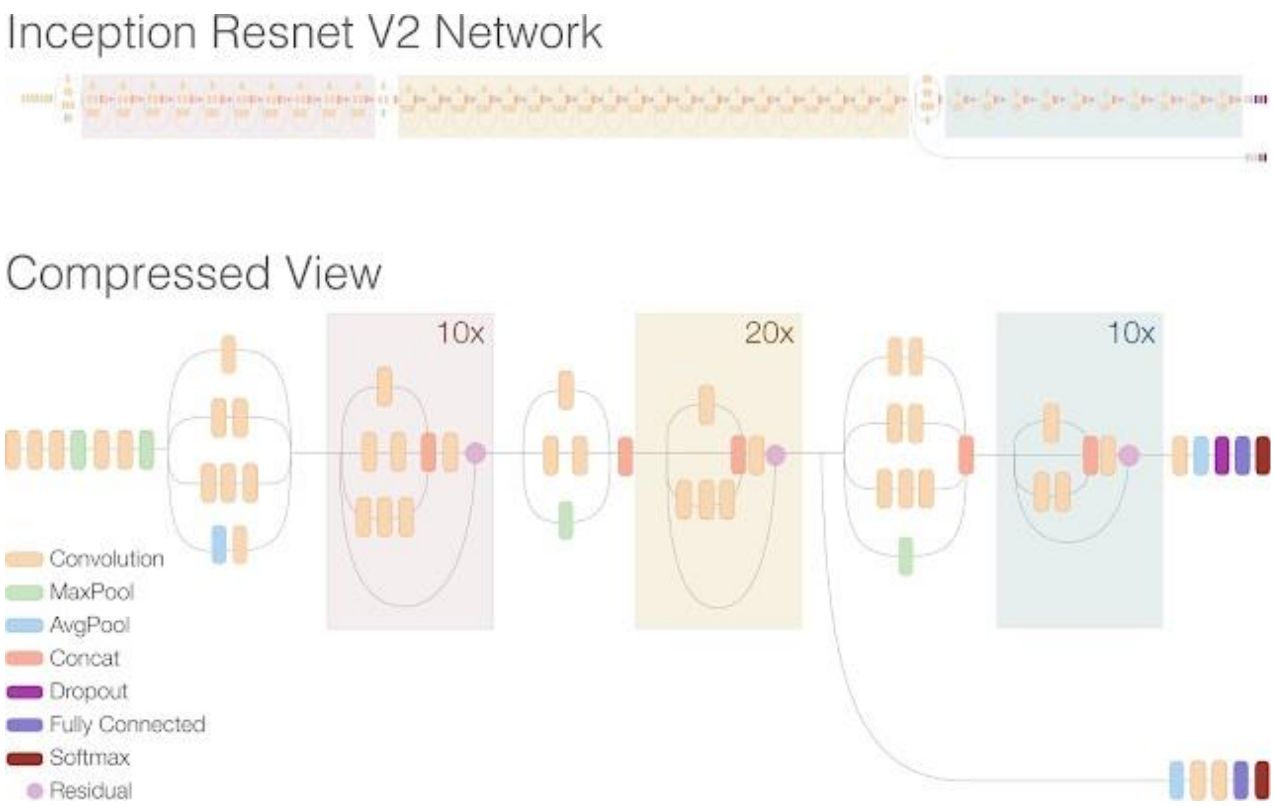


Fig 33 Sturcture of Inceotion-Resnet-V2

Inception-ResNet-V2 combines the idea of Inception and Resnet, further improved performance of Inception and make Inception more powerful to classify images.

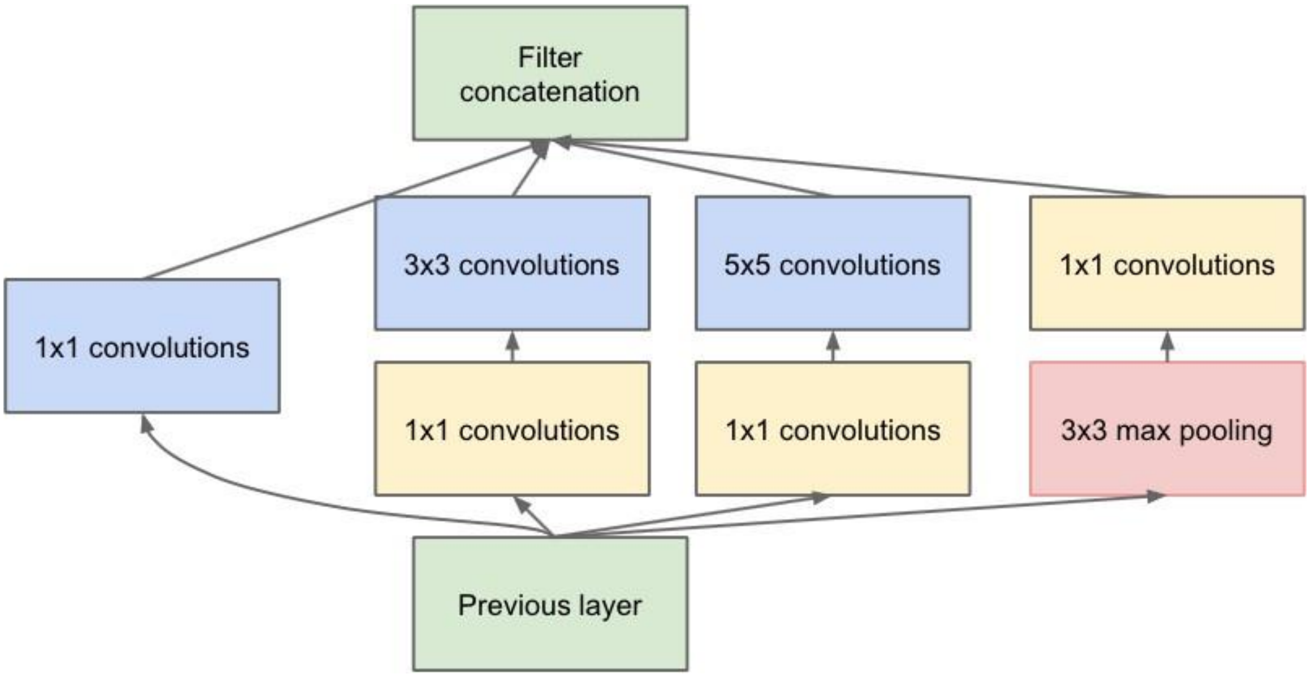


Fig 34 Basic Sturcture of Inceotion

Starting from dividing convolutional kernel into blocks, Inception has used convolutional kernel with size 1*1, 3*3 and 5*5. Considering about the powerful feature of pooling layer, pooling layer has been counted in here as well. Since the amount of calculation would be too large for the convolutional kernel with size 5*5, one additional operation has been added to merge all the output of previous layer. In such case, the amount of calculation has been reduced significantly.

type	patch size/stride or remarks	input size
conv	$3\times3/2$	$299\times299\times3$
conv	$3\times3/1$	$149\times149\times32$
conv padded	$3\times3/1$	$147\times147\times32$
pool	$3\times3/2$	$147\times147\times64$
conv	$3\times3/1$	$73\times73\times64$
conv	$3\times3/2$	$71\times71\times80$
conv	$3\times3/1$	$35\times35\times192$
$3\times$Inception	As in figure 5	$35\times35\times288$
$5\times$Inception	As in figure 6	$17\times17\times768$
$2\times$Inception	As in figure 7	$8\times8\times1280$
pool	8×8	$8\times8\times2048$
linear	logits	$1\times1\times2048$
softmax	classifier	$1\times1\times1000$

Fig 35 Network Sturcture of Inceotion

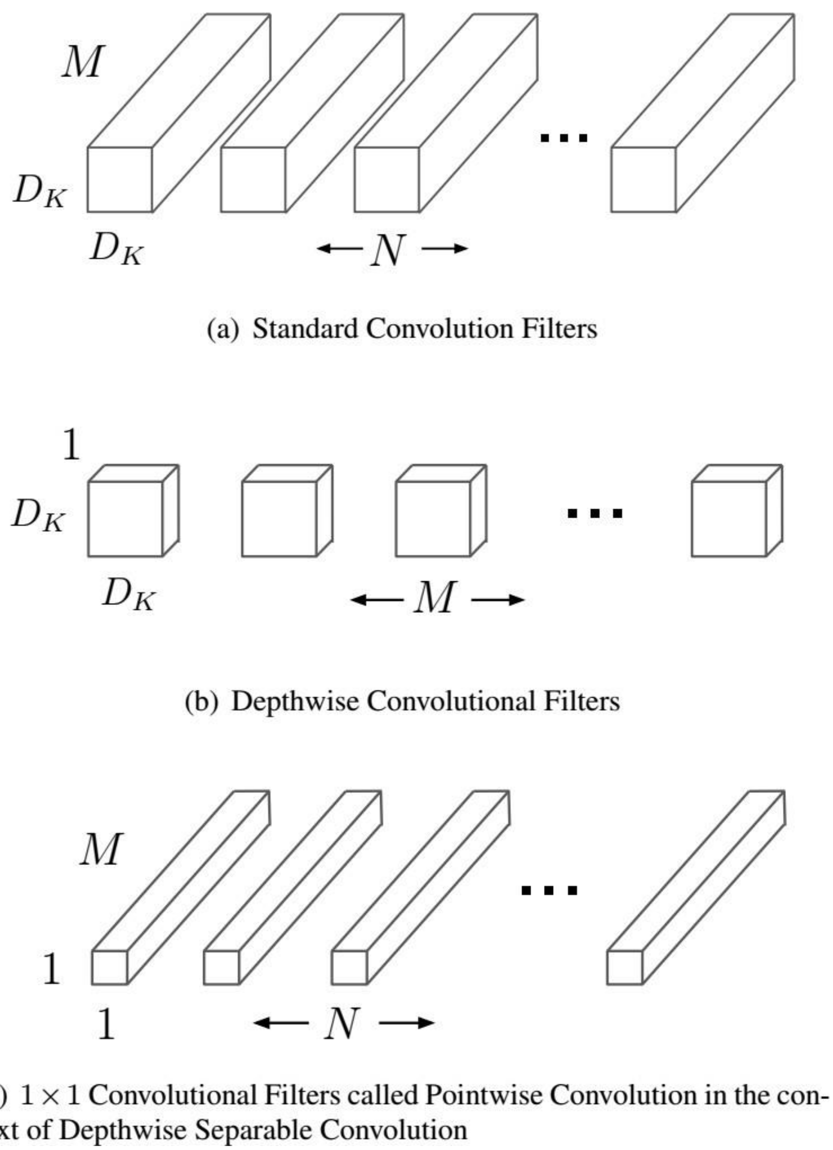
3. MobileNet

Deep learning has made a huge number of success in computer vision field in the recent years, and neural networks are also keeping improving and evolving in the same period. Besides for the Cloud Vision API, the fast growing of mobile devices' computational power can also give us a chance to deliver these cutting-edge technologies into billions of users at any place and any time.

In such condition, MobileNet has been developed to efficiently run a neural network on a resources-restricted environment.

Fig 36 Sample use cases for Mobile Net

MobileNet has used a Depthwise Seprable Convolution which is also the most important part of MobileNet. Besides, the network structure is also one the reasons why MobileNet can run such fast. While the width and resolution can be adjusted to make up for the latency and accuracy.



(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Fig 37 Basic Sturcture of Depthwise Seprable Convolution

Depthwise Separable Convolutions are factorized convolutions which can form a 1*1 convolution by factorizing a standard convolution. In MobileNet, depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1*1 convolution to combine outputs the depwise convolution. The above figure illustrates such an idea.

The following tables shows the structure of MobileNet:

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fig 38 Network Structure of MobieNet

Results and Analysis

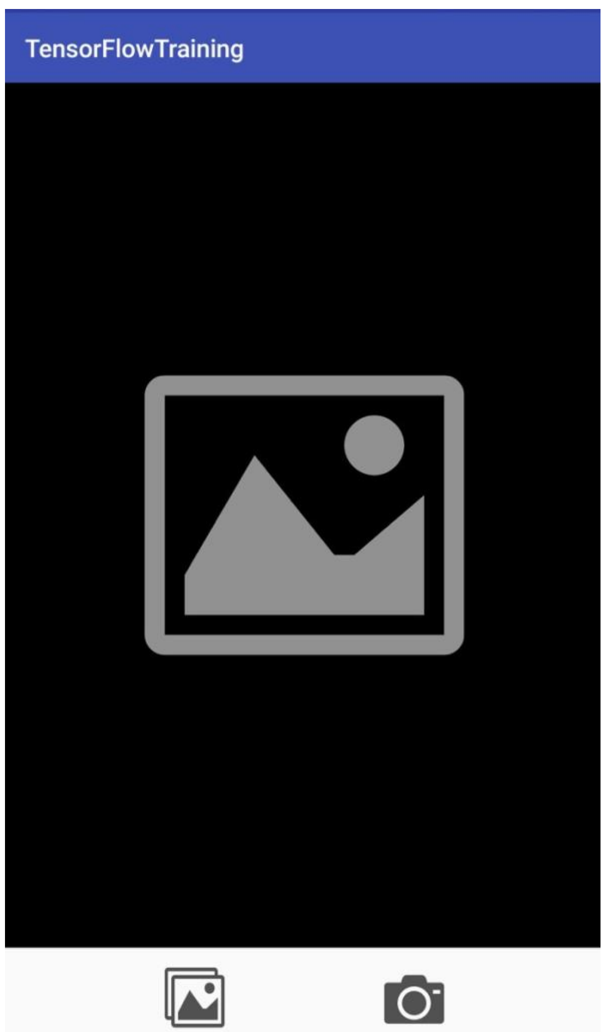
1. Training Accuracy and Model Size

Inception v3	Accuracy: 0.98	Size: 88M
MobileNet 100, 224	Accuracy: 0.94	Size: 10M
MobileNet 050, 224	Accuracy: 0.92	Size: 3M
MobileNet 050, 128	Accuracy: 0.91	Size: 3M
MobileNet 035, 224	Accuracy: 0.94	Size: 2M

2. Mobile Application Performance

There are two versions of my application, one focus on taking pictures or choosing pictures from library, while another one focus on real-time classification.

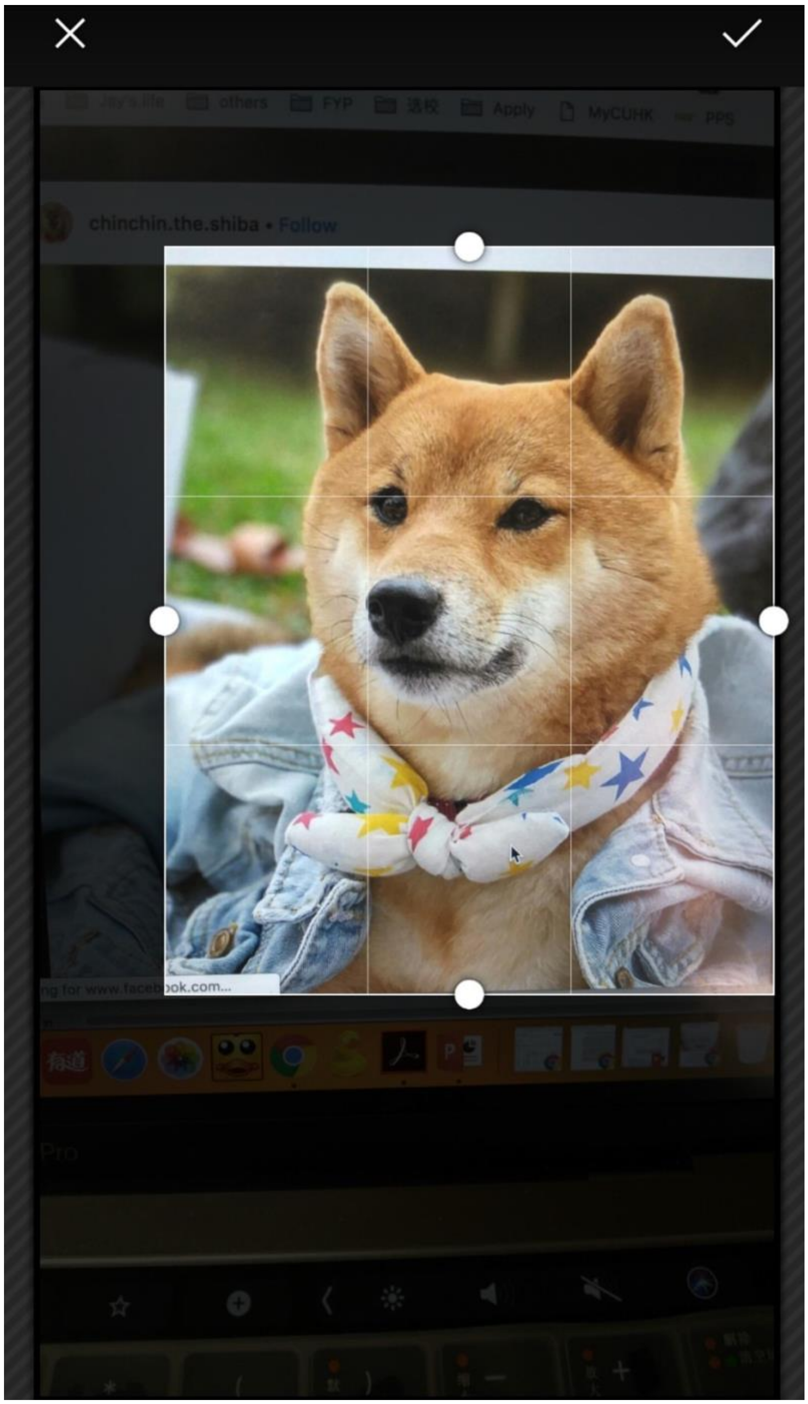
For the first version, the app looks like:



While users can choose to select a photo from library or to take a photo by using the system camera.

If a user chooses to take a photo, he will see the real-time camera view:

After touching the button, user shall see the interface like this:

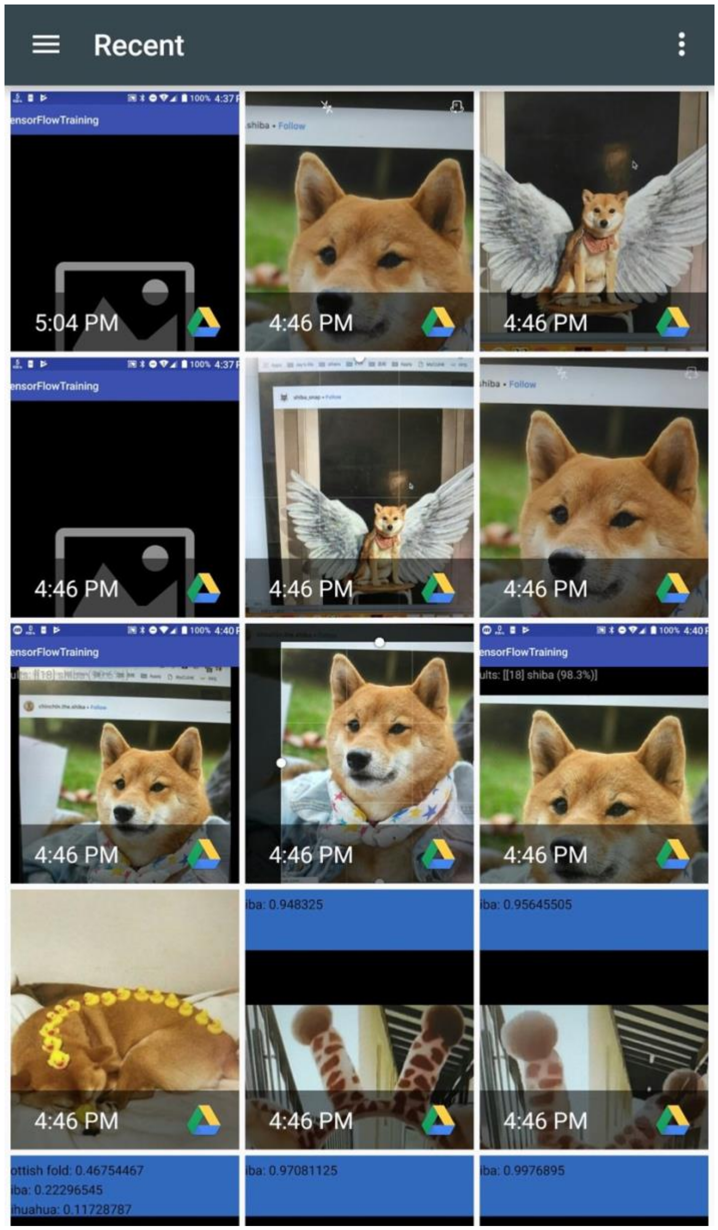


RESULTS AND ANALYSIS

User can choose to crop or not crop this photo. In most of the cases, crop will help increase the classification accuracy. Let’s see an example:

We can see that accuracy for not crop vs crop is 0.936 vs 0.983.

If the user wish to select a picture from album, the interface will be:



User can choose to select a photo from album, the following operations would be similar to the previous procedure.

RESULTS AND ANALYSIS

2.1 Classify Inaccurate Image

Now let's see an example for inaccurate image:

The accuracy for images with disturbance can still remain about 0.6.

2.2 Classify In Real-Time

Now let’s see the second version which supports real-time classification.

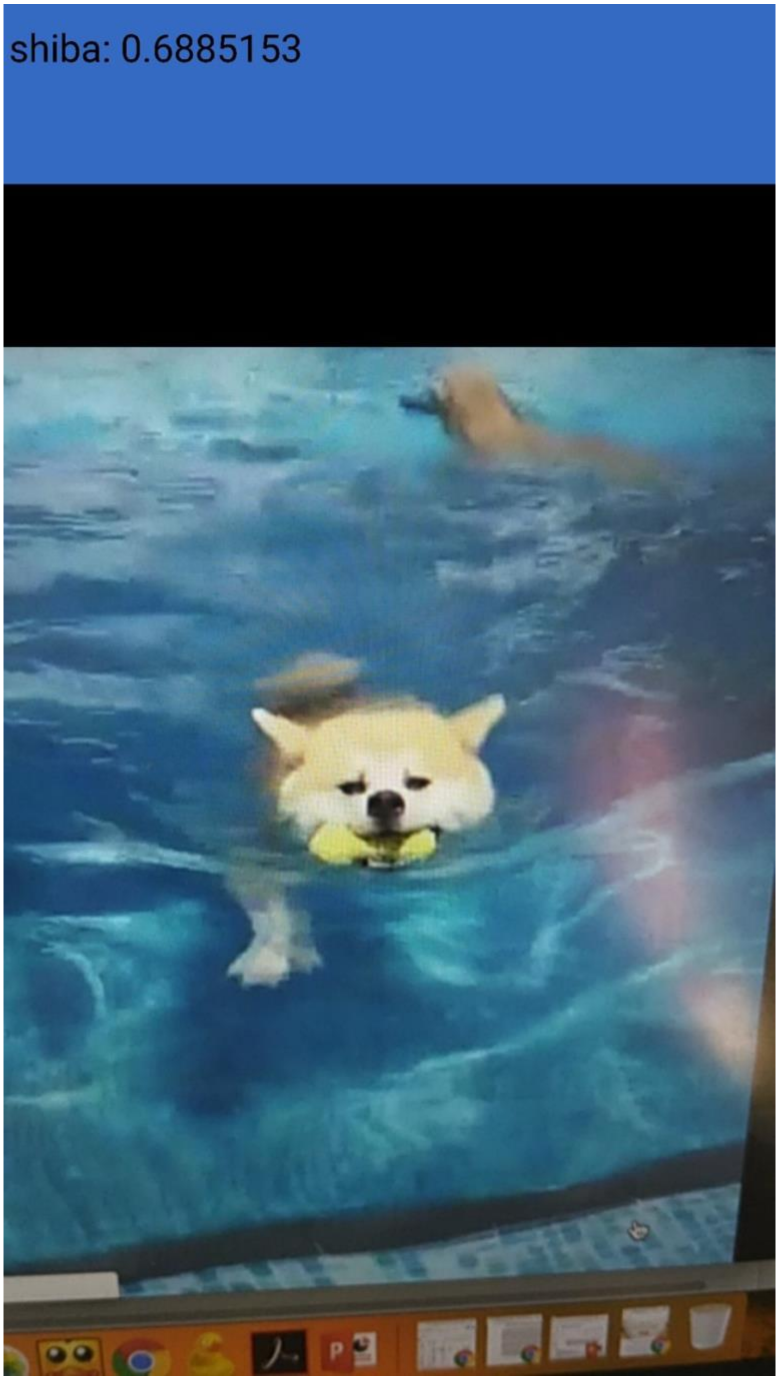
We can classify shiba from Instagram videos in real-time as well as achieve a high accuracy at 0.95.

2.3 Classify Blur Image

Classify blur images:

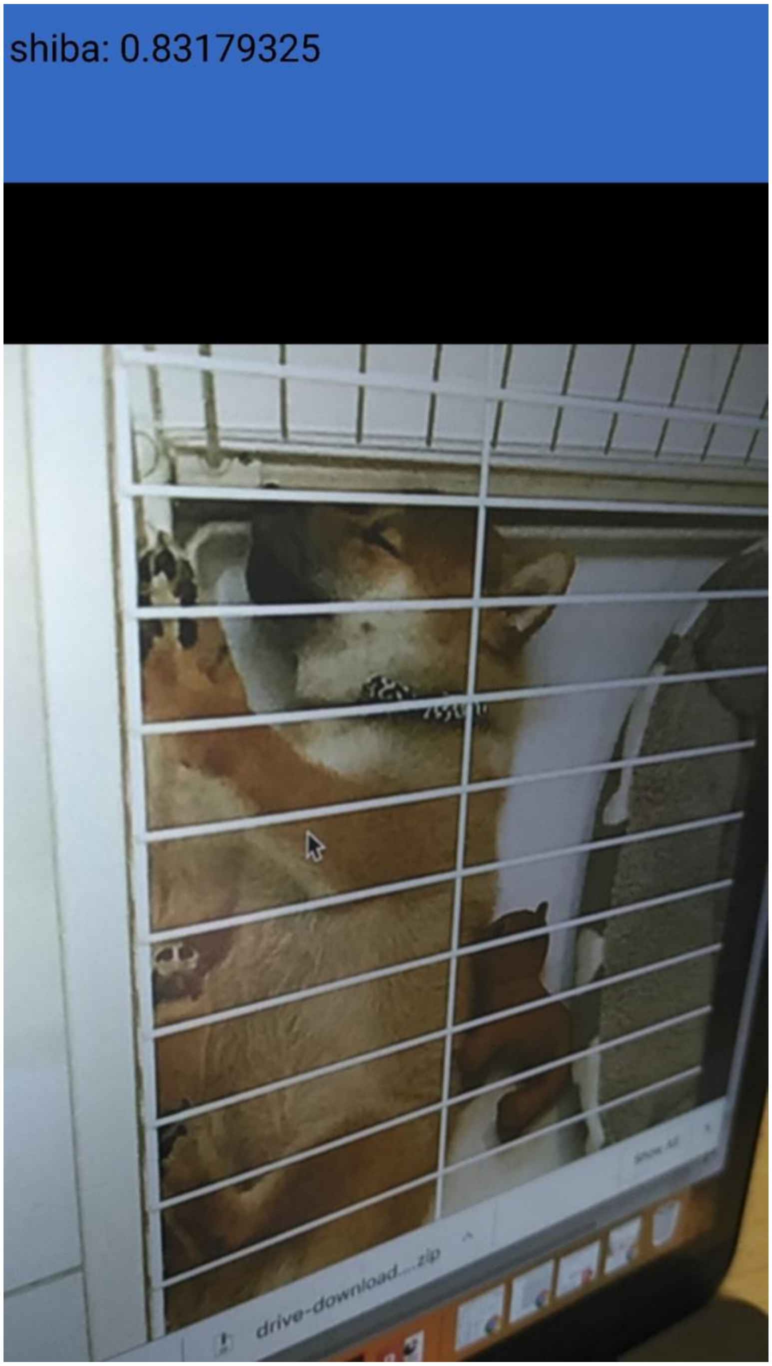
2.4 Classify Incomplete Image

Classify incomplete images:



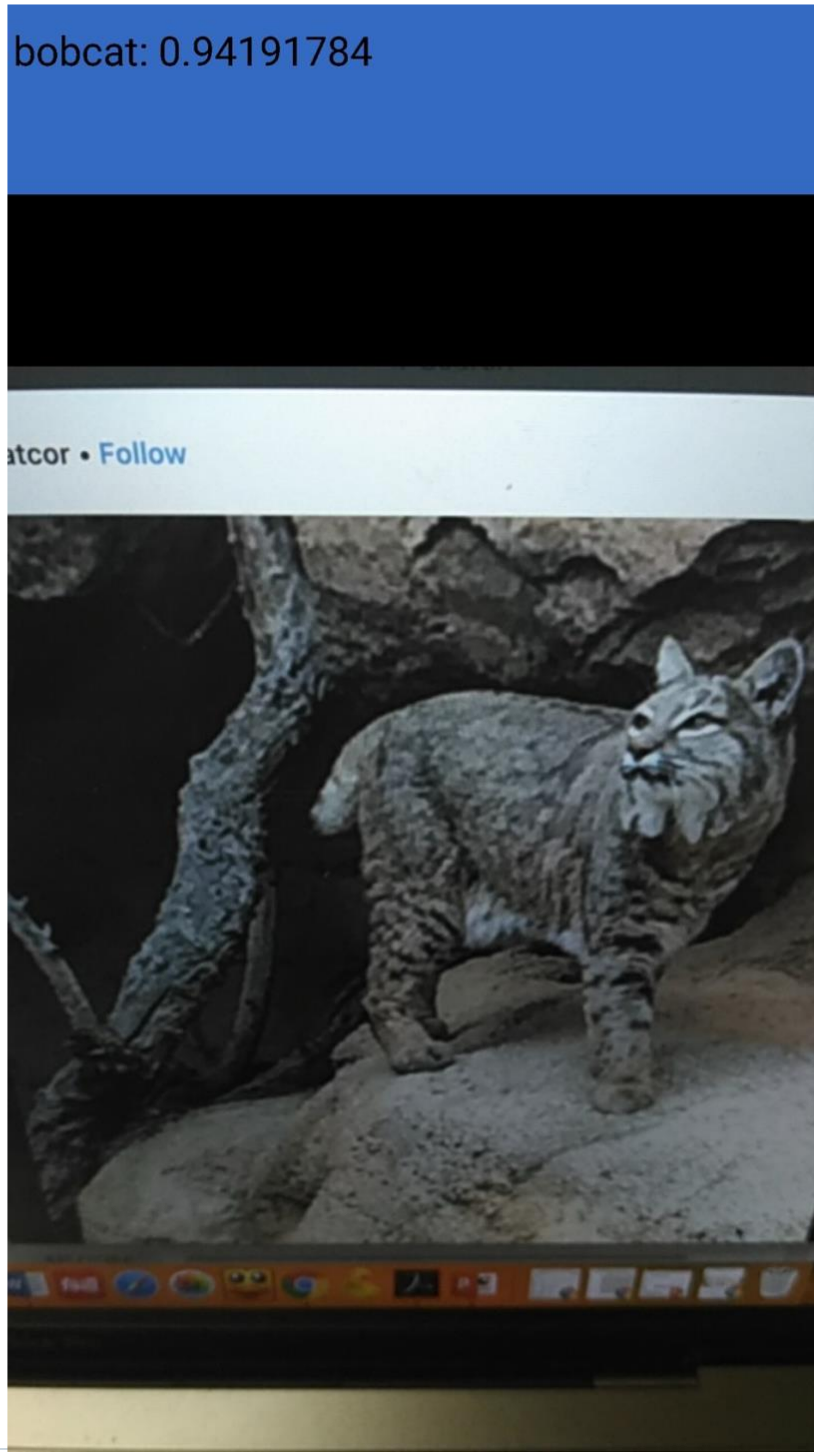
2.5 Classify from Different Angle

Classify pets in cage from a different angle:

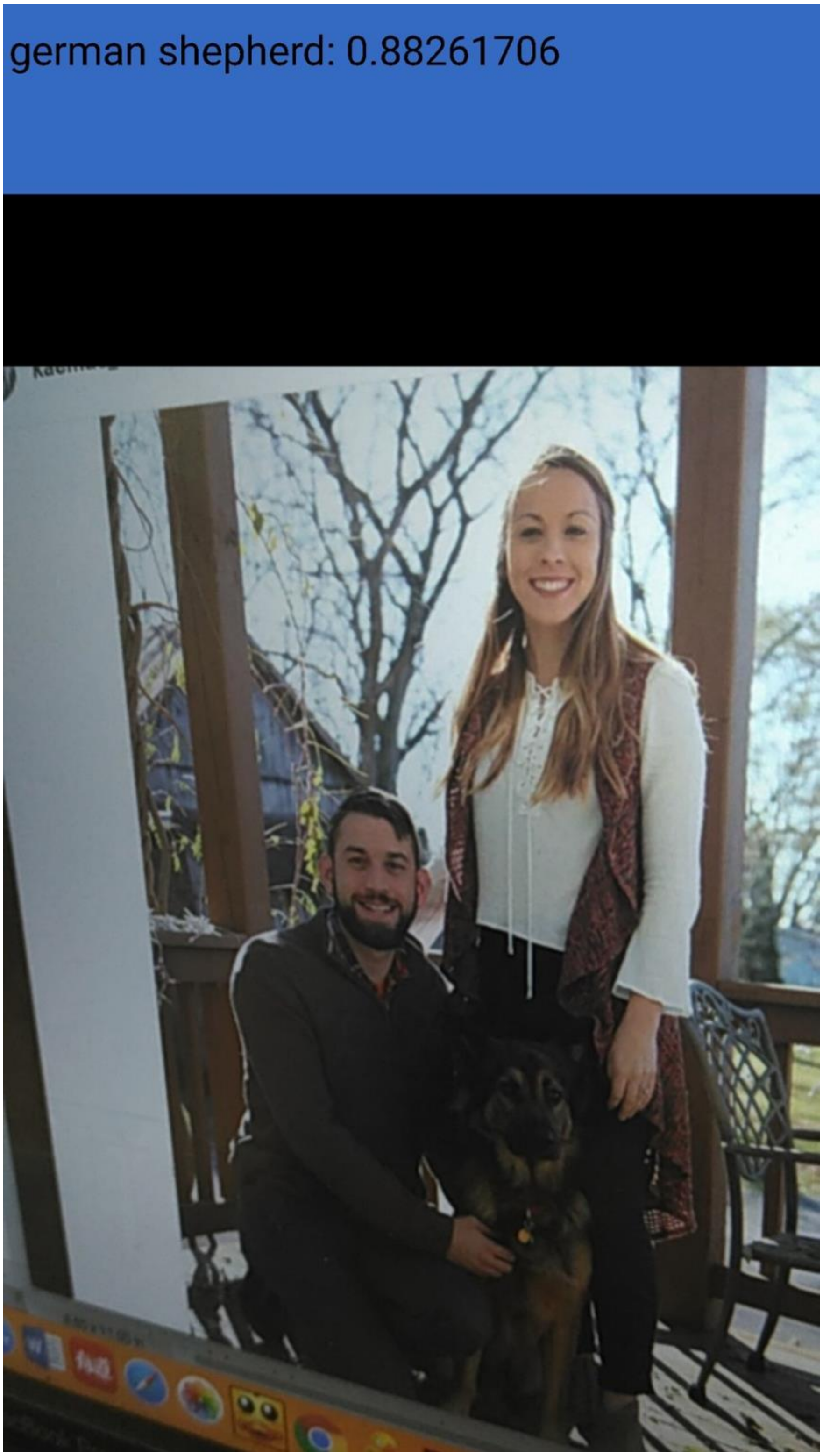


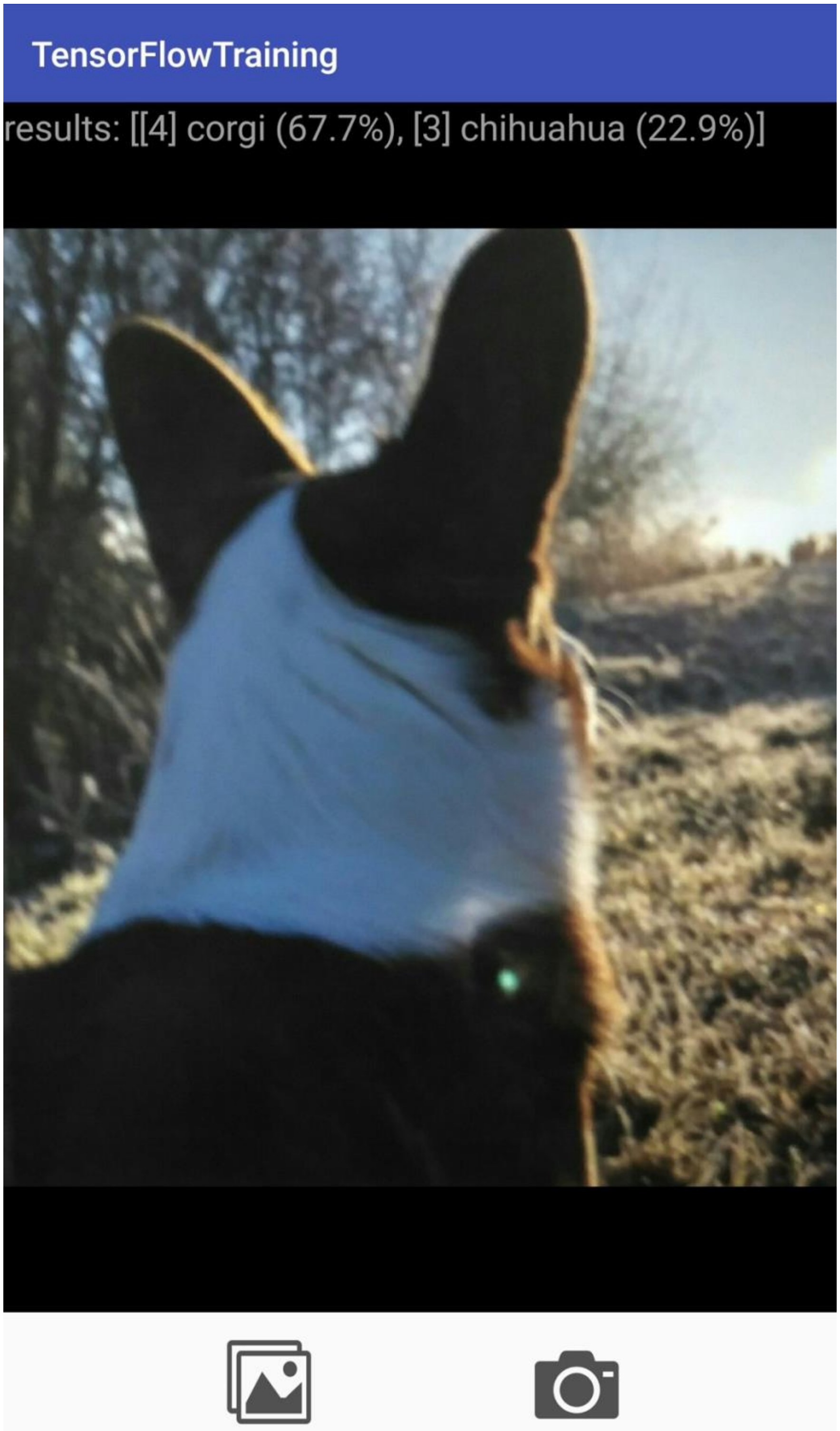
2.6 Classify Other Species from Different Angle

Classify other species from different angles:



RESULTS AND ANALYSIS

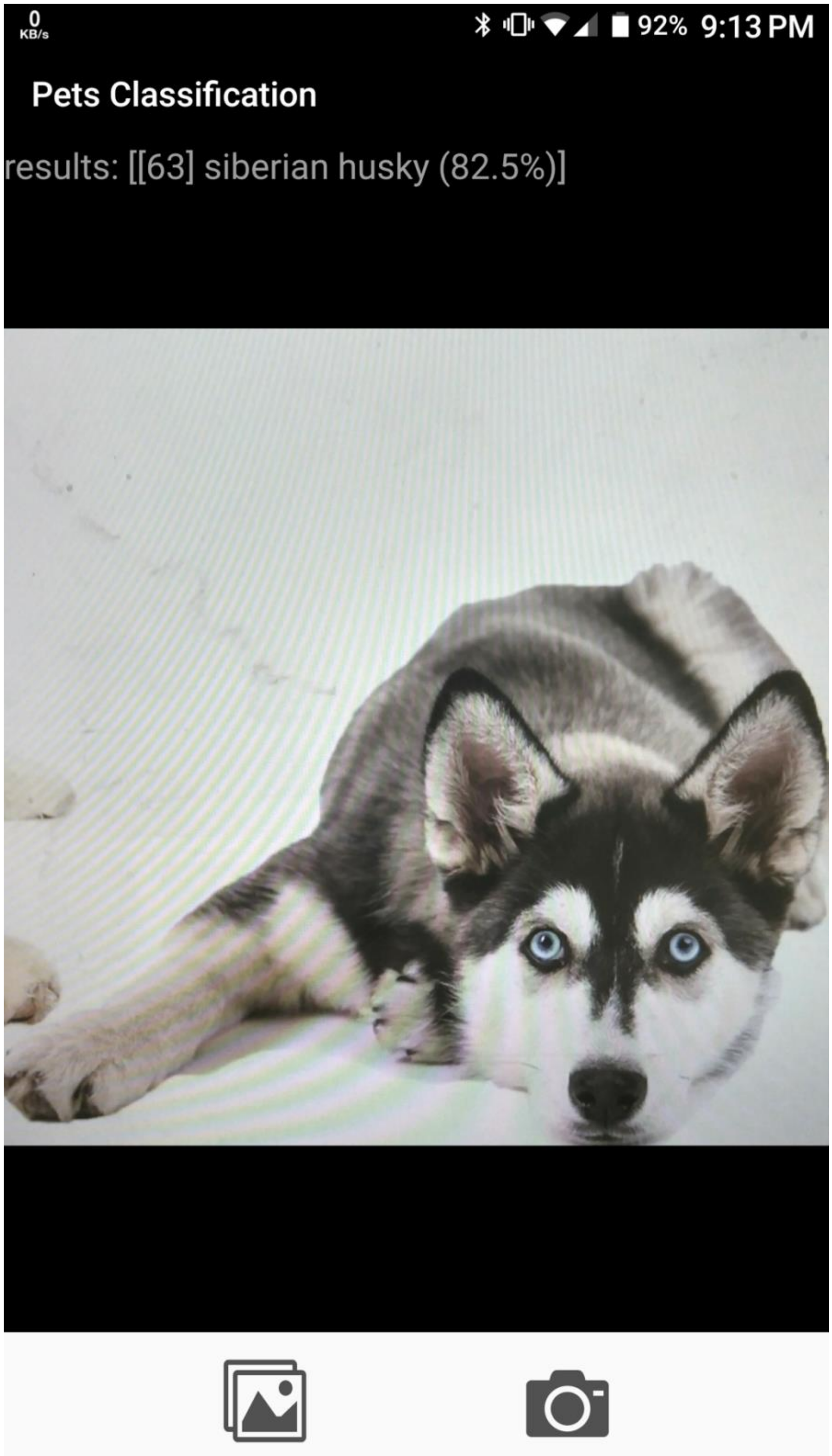


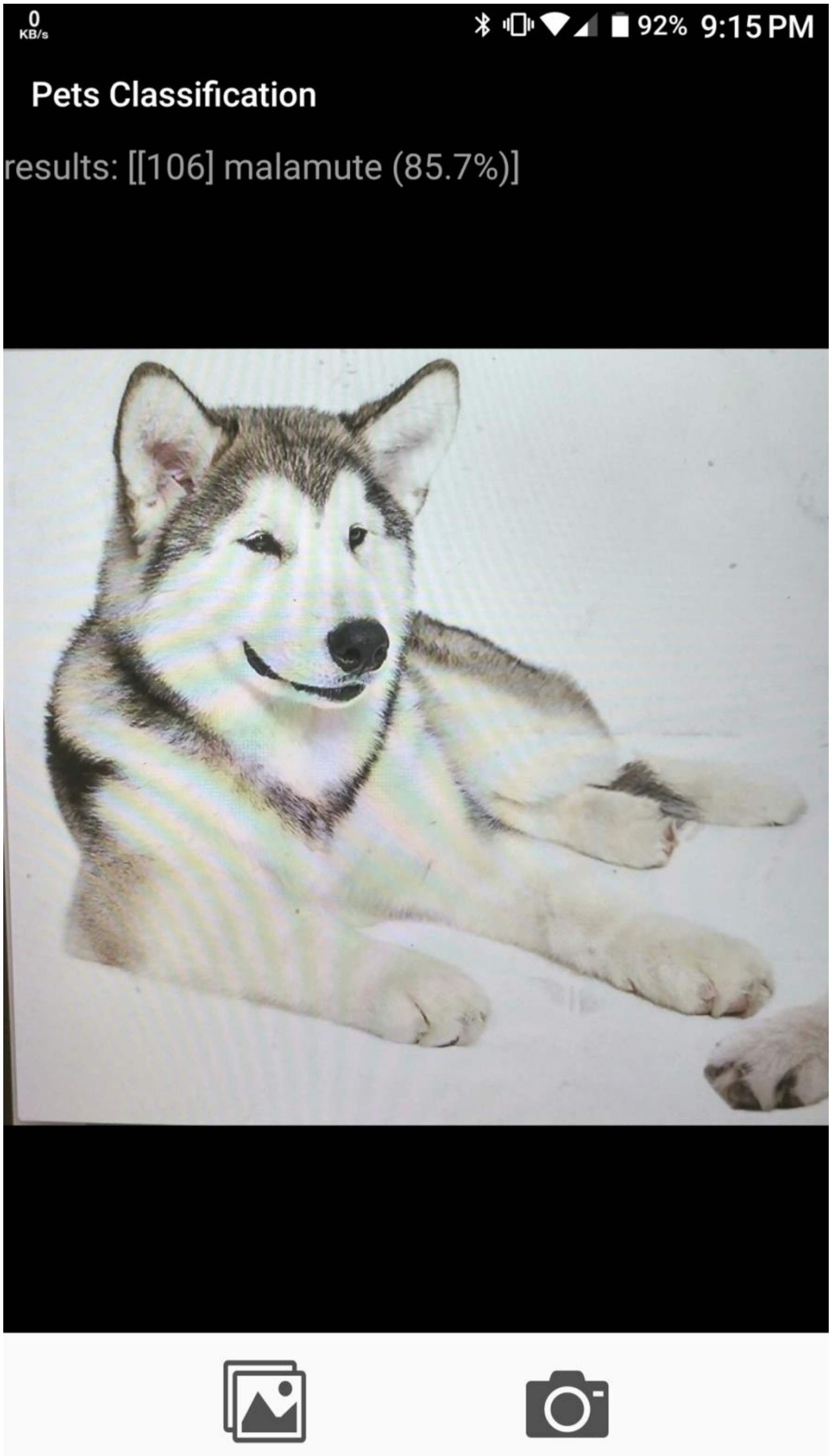

TensorFlowTraining
results: [[4] corgi (67.7%), [3] chihuahua (22.9%)]

2.7 Classify Similar Species

With the new dataset, we can try to explore more similar species in this semester, now let's try the first which is also the most common example:

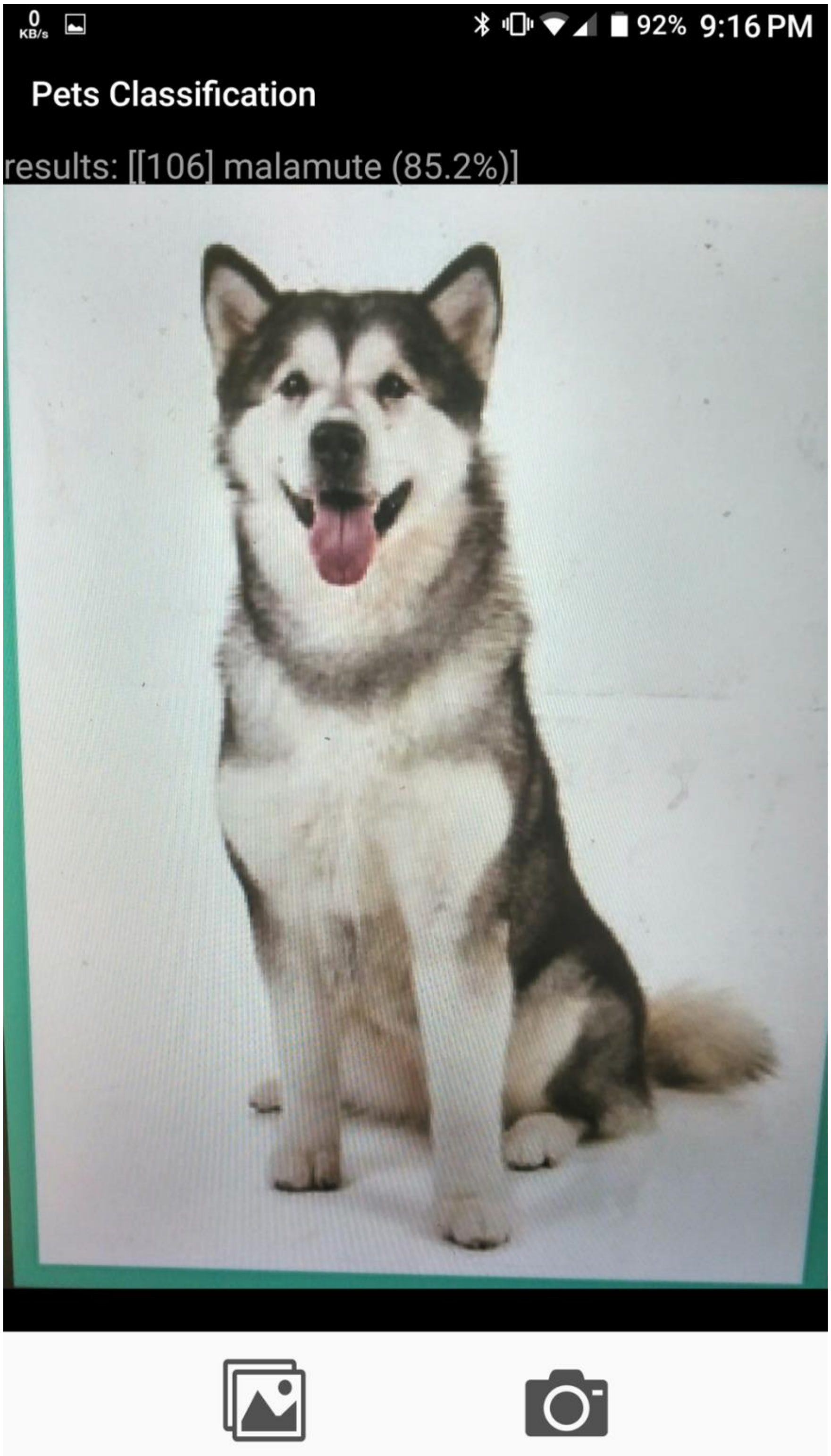
Husky vs Malamute (High Quality):


0 KB/s
92% 9:13 PM
Pets Classification
results: [[63] siberian husky (82.5%)]

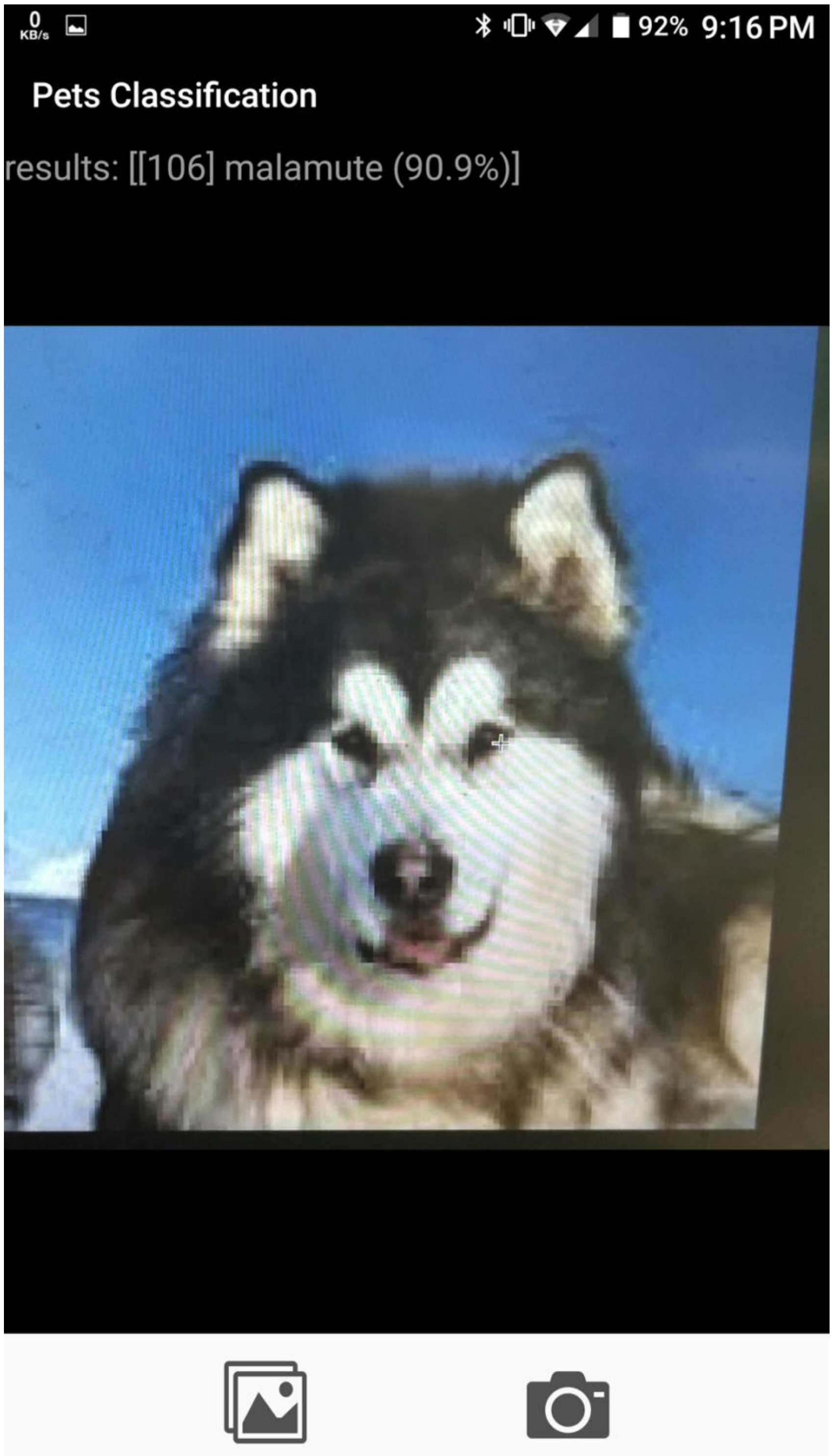

0 KB/s
92% 9:15 PM
Pets Classification
results: [[106] malamute (85.7%)]

Husky vs Malamute (Normal Quality):


0
KB/s
92% 9:15 PM
Pets Classification
results: [[63] siberian husky (75.6%)]


0 KB/s
92% 9:16 PM
Pets Classification
results: [[106] malamute (85.2%)]

Husky vs Malamute (Low Quality):


92% 9:16 PM
Pets Classification
results: [[106] malamute (90.9%)]

RESULTS AND ANALYSIS

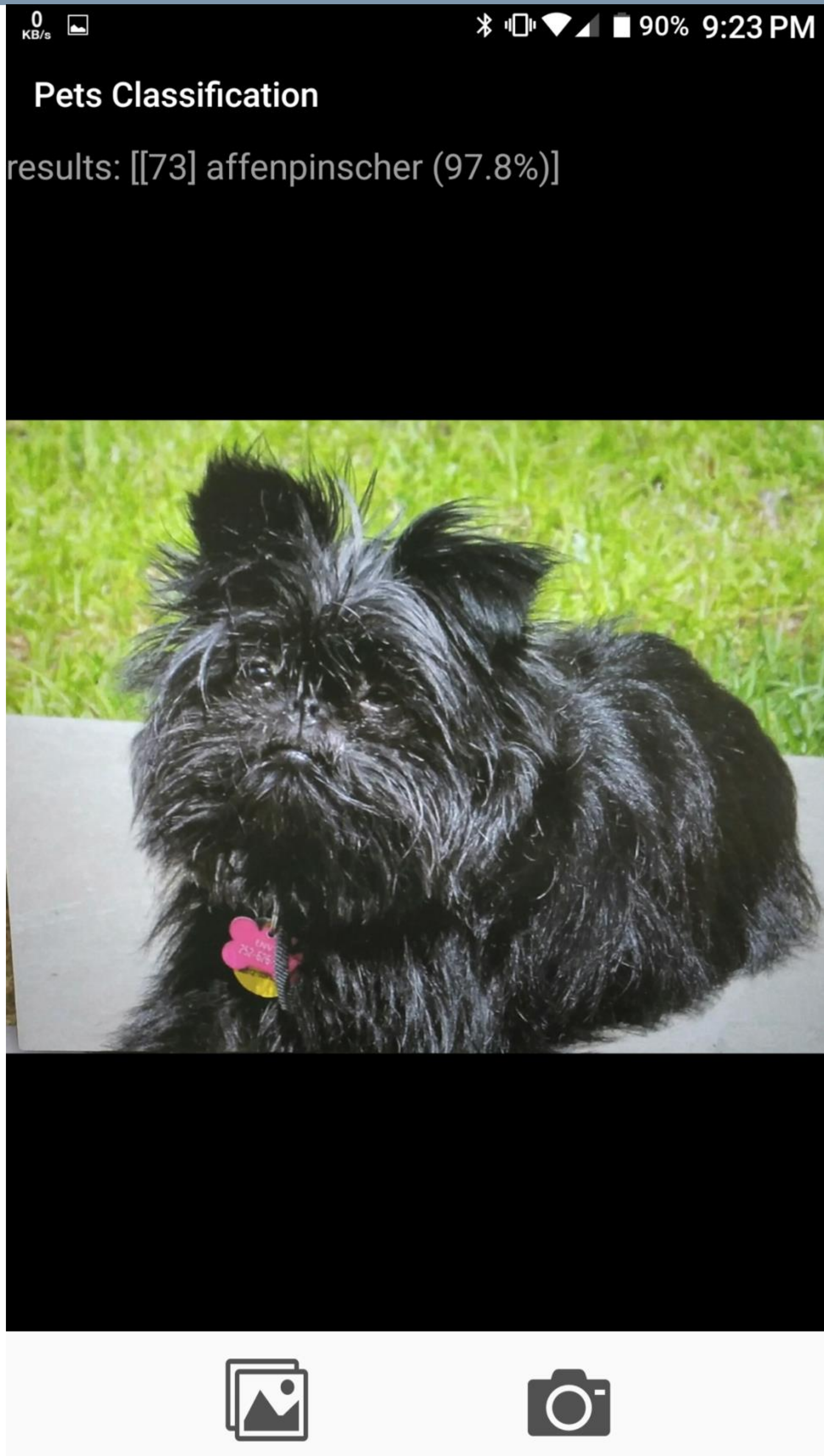
2.8 Classify Other Similar Species

RESULTS AND ANALYSIS

Ciarn (top left) vs Affenpinscher (top right)

vs Bouvier des flandres (bottom left) vs Schnauzer (bottom right):

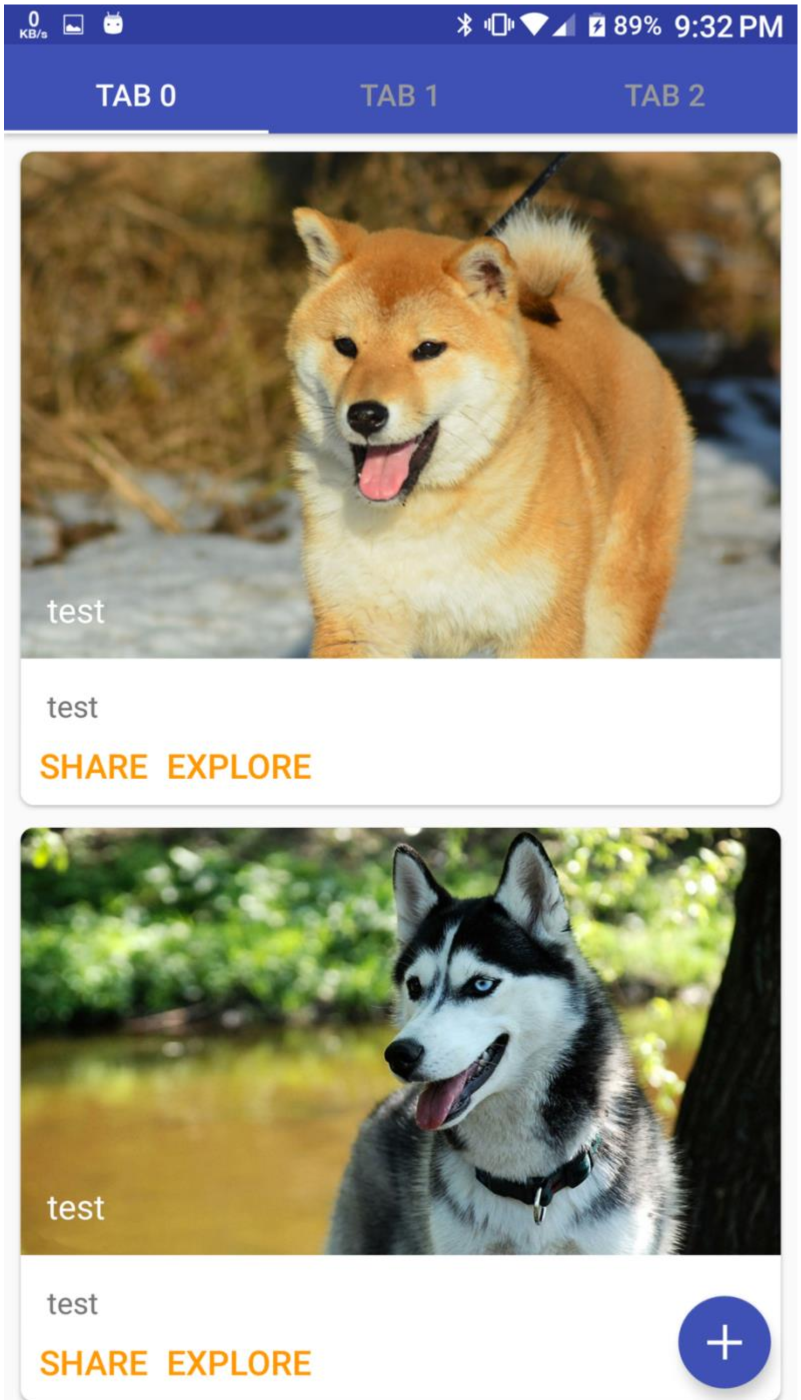

0 KB/s
91% 9:20 PM
Pets Classification
results: [[87] cairn (78.1%)]

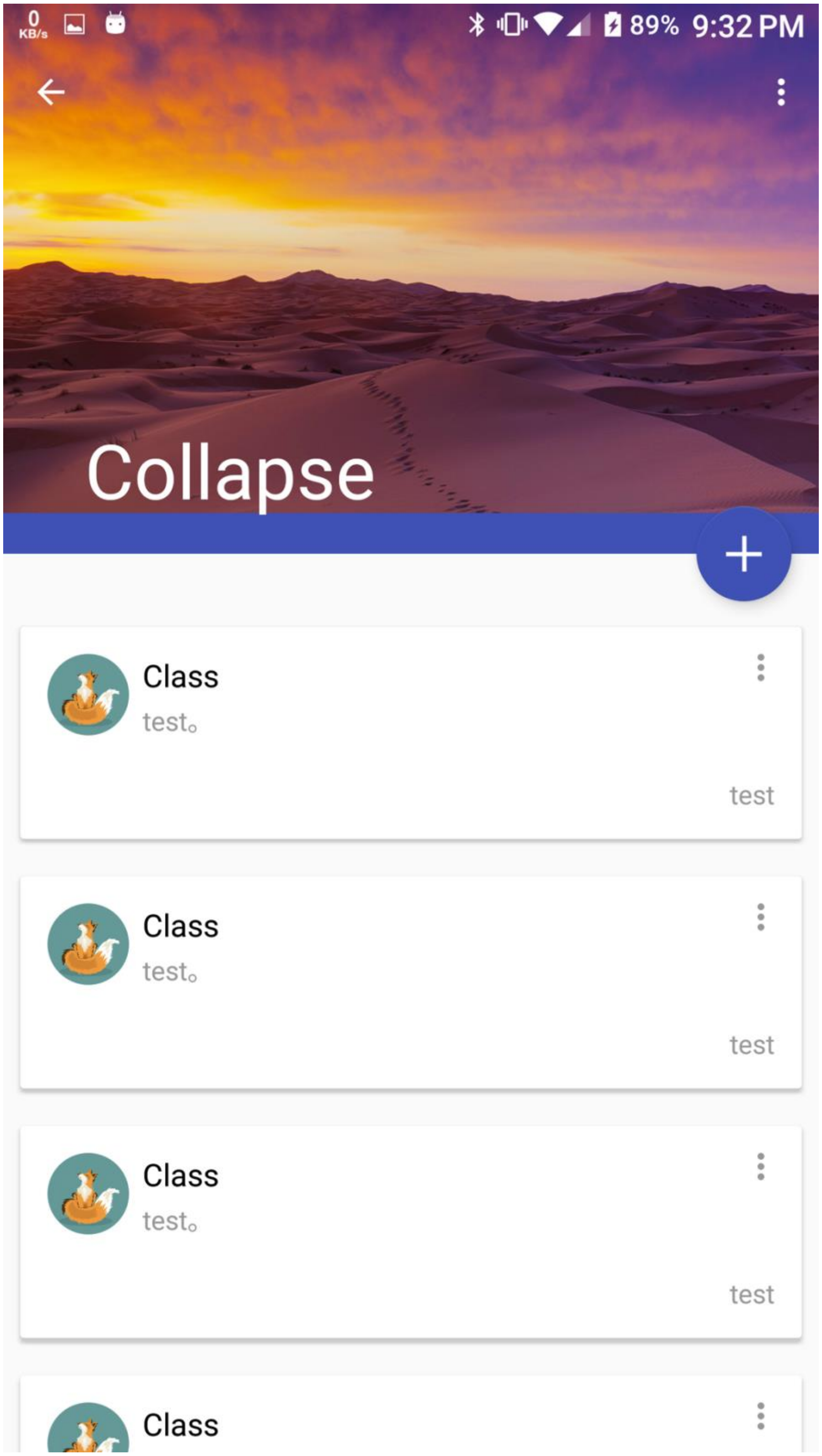

0 KB/s
90% 9:23 PM
Pets Classification
results: [[73] affenpinscher (97.8%)]


0 KB/s
89% 9:26 PM
Pets Classification
results: [[12] bouvier des flandres (43.0%)]


91% 9:19 PM
Pets Classification
results: [[98] giant schnauzer (39.2%), [118] schnauzer (33.9%)]

2.9 Possible Layout




0 KB/s
89% 9:32 PM
Collapse
Class
test。
test
Class
test。
test
Class
test。
test
Class

All the original training images are crawled from Google, Bing or Baidu in the beginning of this semester and all the new images are coming from Stanford Dogs Dataset, while all the images used for performance testing result are chosen from other sources at the time of testing. In such case, we can make sure that testing images are new to my model.

From all the results above, we can see that this model can achieve a high accuracy on classifying normal images of pets. Besides, it also supports for classifying many difficult-to-classify images such as blur, distorted or incomplete images as well.

Conclusion

1. Review

When I first started this project, I have no confidence in the final result since I need to start from collecting images and build my dataset. In the very beginning, I intentionally chose to pick those pets which look like different to each other to avoid no result in the end.

However, after I have started this project, the performance of the model really impressed me. Even though my dataset only focuses on high-quality and single-object images of pets, after I deployed my model to the mobile devices, it achieved a much better result than my expected.

Inspired by this result, I investigated more methods to get more insights into this project. Including searching for more suitable methods and using these methods on my tasks and starting a new real-time classify application to examine the performance of new model.

The pressure comes from the weekly meeting has also became the motivation which prompted me to keep improving. I really appreciate all the advice and help provided by Professor Michael Lyu and his PhD student Xu Hui. They suggested me to choose a meaningful dataset in the very beginning of my project. In the middle, they also provided a lot of useful tips which helped me continuously improve my project. In the end of the first semester, Professor Michael also provided me some extra chances to challenge myself.

In the second semester, I continue to work on the improvements of the first version of my application, including improving UI, increasing datasets to do similar species analysis, and developing new object detection application to do object detection on the new dataset. In this semester, professor Michael Lyu and his PhD student Xu Hui also have provided me with a lot of ideas and comments on both of the original task and the new object detection task. I believe that the techniques I have learned through this project will certainly help me in my future study as well as career.

2. Shortcomings and Future Works

2.1 UI is not quite fancy.

The new UI has been designed (page 98, 99) and I will combine the new design with the classification app in a short time.

2.2 Object detection function.

In this semester, I have also explored object detection in mobile devices. Right now, I am working on training an object detection model based on our new dog dataset. The new object detection application will come out in the near future if succeeded.

Reference

[1] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, 2017

[2] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”, 2016

[3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, “Rethinking the Inception Architecture for Computer Vision”, 2015

[4] Introduction to Image Processing, https://www.engineersgarage.com/articles/image-processing-tutorial-applications

[5] Rohan & Lenny, Neural Networks & The Backpropagation Algorithm, Explained, https://ayearofai.com/rohan-lenny-1-neural-networks-the-backpropagation-algorithm-explained-abf4609d4f9d

[6] Laurence Moroney, Using TensorFlow Lite on Android, https://medium.com/tensorflow/using-tensorflow-lite-on-android-9bbc9cb7d69d

REFERENCE

[7] Jayesh Bapu Ahire, The Artificial Neural Networks handbook: Part 1, https://www.datasciencecentral.com/profiles/blogs/the-artificial-neural-networks-handbook-part-1

[8] Machine learning on mobile: on the device or in the cloud, http://machinethink.net/blog/machine-learning-device-or-cloud/

[9] AlphaGo, https://deepmind.com/research/alphago/

[10] AlphaGo Zero, https://deepmind.com/blog/alphago-zero-learning-scratch/

[11] Digitital Image Processing, https://sisu.ut.ee/imageprocessing/book/3

[12] Feature detection, https://en.wikipedia.org/wiki/Feature_detection_(computer_vision)

[13] CS231n Convolutional Neural Networks for Visual Recognition, http://cs231n.github.io/convolutional-networks/#overview

[14] Joahua, Reading Note: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, https://joshua19881228.github.io/2017-07-19-MobileNet/

[15] Ke-Lin Du, M. N. S. Swamy, Neural Networks and Statistical Learning, 2013

[16] Calvin Wong, Applications of Computer Vision in Fashion and Textiles, 2017

[17] What Is Knowledge Representation (KR), ftp://ecs.csus.edu/csc215lu/ann.doc

[18] Joost Nico Kok, ARTIFICIAL INTELLIGENCE, 2009.

[19] Kevin Gurney, An Introduction to Neural Networks, 1997

[20] Kevin Swingler, Applying Neural Networks: A Practical Guide, 1996

[21] Stefano Ceri, Piero Fraternali, Aldo Bongio, Designing Data-Intensive Web Applications, 2003

[22] Introduction to Artificial Neural Networks, https://www.oreilly.com/library/view/neural-networks-and/9781492037354/ch01.html

[23] Mohamed Gad-el-Hak, Large-Scale Disasters: Prediction, Control, and Mitigation, 2008

REFERENCE

[24] The Machine Learning Dictionary, www.cse.unsw.edu.au/~billw/mldict.html

[25] Applied Deep Learning - Part 1: Artificial Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

[26] Paul Caspi, Christine Mazuet, Natacha Reynaud Paligot, About the Design of Distributed Control Systems: The Quasi-Synchronous Approach, 2001

[27] Building an Image Recognition Model for Mobile using Depthwise Convolutions https://heartbeat.fritz.ai/building-an-image-recognition-model-for-mobile-using-depthwise-convolutions-643d70e0f7e2

[28] Stanford Dogs Dataset, http://vision.stanford.edu/aditya86/ImageNetDogs/