A Mobile Application for Dog Breed Recognition and Detection based on Deep Learning

WenBin Chen*
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
wenbinc4@student.unimelb.edu.au

Fang Wu*
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
fangw@student.unimelb.edu.au

Richard O. Sinnott
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
rsinnott@unimelb.edu.au

(* = equal contribution)

Abstract

Deep learning is an approach from machine learning that provides the ability to tackle the problem of classification and prediction based on raw data to train models that learn the characteristics or knowledge from data. As a main research direction of deep learning, Convolutional Neural Network (CNN) models provide an approach for image classification and detection compared with other traditional computing vision methods. In this work we describe a method for classifying breeds of dogs from a natural static picture by capturing the position of the detected dog based on a Faster Regional CNN model and perform recognition of breeds through another CNN model through transfer learning. The results achieve nearly 85 percent accuracy for breed classification of 50 classes of common dog and 64 percent for 120 other dog types. This work also recognizes dog types from more complex images. An IOS application was designed to implement and actualize the two models in a mobile environment to facilitate the utilization of image classification.

1. Introduction

Deep learning is one of the recent developments of artificial intelligence which has garnered considerable attention. The idea of neural networks has been around for a long time, but significantly better performance has now been achieved for some relatively simple recognition tasks such as hand-written number classification and face detection. The ability to handle more complex problems is limited, however up to 2012, most researchers believed that computer vision system and models must be carefully hand-designed with comprehensive analysis to capture the characteristics of the desired tasks. A notable neural network model (AlexNet) showed outstanding performance in the 2012 ImageNet benchmark compared with traditional computer vision approaches [2]. More recently, Convolutional networks (ConvNets) have been viewed as the basic approach for solving graphics-related problems and more complex and diverse ConvNets now exist.

The fast development of convolutional neural networks (CNN) and computing power provide the opportunity to solve more complicated image processing problems, especially for detection of objects in an image with high accuracy and speed. The region proposal method based on convolutional neural networks (RCNN) provides improved performance over other vision analysis techniques, however, the requirement for extremely high computing power make them less ubiquitous for normal utilization. The improved methods and post-processing of feature maps by Fast/Faster RCNN drastically reduces the demands for computing power and hence drives the accuracy to a higher level, including meeting the requirements for real-time object detection [3]. Figure 1 shows that the region proposal phase happens before and after feature extraction (ConvNet sampling) for two models.

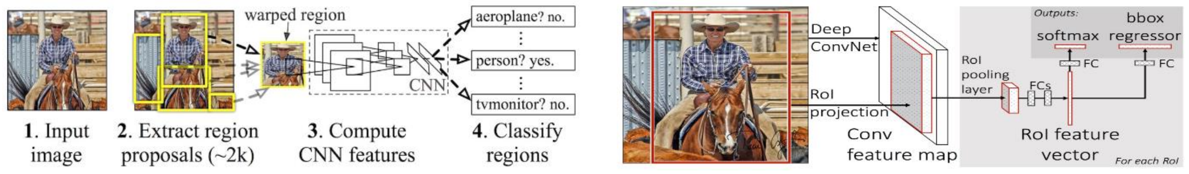


Figure 1: Difference between Region-CNN and Fast/Faster RCNN

In this paper, we describe a model that implements the Faster RCNN technique to achieve normal object detection and a fine-tuned convolutional neural network based on transfer learning for dog breed classification to achieve better overall performance. Initially, we pretrain a Faster-RCNN model with Regional Proposal Network (RPN) bounding box detection to achieve detection of 10 different objects (dogs) including the background and the dogs. A fine-tuned CNN dog classification model which support 120 different breeds is applied to the confidence area of dog objects within the image and the SoftMax score of probability for that confidence area is calculated. The final deployment of the system is delivered in a mobile environment facilitating the detection with convenient mobile devices. The system utilizes Cloud-based servers to address the drawbacks of limited computing abilities of mobile devices to achieve both the portability and speed required for the utilization of the model.

The rest of the paper focuses on the analysis of the Faster RCNN model and the fine-tuned convolutional neural network model. We describe the process of hyperparameter tuning and model suitability analysis. The accuracy and performance of the model are presented. The mobile realization is introduced, including the model transformation and underlying communicating model. Finally, the drawbacks and limitations of the model are presented, and areas of the future work are identified.

1.1 Related Work

Since the excellent performance derived by AlexNet as the first deep learning technique in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for vision analysis technology, multiple ConvNet architectures have evolved in the ILSVRC, such as extremely deep networks proposed by Visual Geometry Group (VGG) and inception module structures proposed by GoogleNet [4]. However, with respect to dog breed detection, none of the above architectures are perfectly suited for classifying considering the depth of the network and the number of useful parameters. As a result, a reasonable model should take the characteristics of the classical models into consideration and consider the amount and quality of the data and computing requirements.

AlexNet consists of eight layers for tuning weights, and an additional SoftMax layer generates the final classification score. Rectified Linear Units (ReLU) is the core innovation proposed by the AlexNet that becomes the most popular activation function [10]. This innovation achieves better data augmentation compared to other existing activation functions.

The main goal of the VGGNet is to explore the relationship between the depth of CNN and the model performance. The better contribution is generated by a much deeper network with small convolutions (3*3). Apart from this, the idea of scale jittering to crop raw images with random scales provides more robustness to feature learning process. And the pre-trained technique is also firstly derived by the VGGNet to accelerate the convergence of the network [10].

GoogleNet proposes the idea of Inception structure that each convolutional layer is composed of smaller convolutions (1*1, 3*3, 5*5), which aims to broaden the width of the network rather than the depth compared with the VGGNet. Furthermore, the model also enhances learning performance through the heuristics of extracting features from different scales simultaneously. And the idea of Model Ensembling is also proposed by the GoogleNet through training multiple model and taking average to enhance the robustness of the model [10].

The accuracy and time for classification provided by the final model are mainly decided by the structure of the CNN network. The core difference of multiple architectures of the CNN network lies in the placement of layers and the depth of the network. Deep CNN models tend to generate more parameters for classification and hence better accuracy, however, the classification time can be significant [14]. The trade-off between accuracy and classification time should be considered, especially for end-user applications.

Another key issue for improving model performance is the selection of optimization methods. The earliest and most classical approach is Stochastic Gradient Descent (SGD) method. However, this method requires too much manually tuning process and is not suitable for people who do not know the classification area initially. Multiple optimization approaches are available now with different characteristics, e.g. AdaDelta, AdaGrad, Adam and RMSprop [25].

Considering the problem of collecting numerous training data to construct a model entirely, transfer learning transfers knowledge from a large source domain to a target domain with smaller dataset [34]. Even the source domain and the target domain contain quite different learning fields, the feature spaces extracted from images could be shared by all the features maps in the higher convolutional layers through the fine-tuning process.

As an area of machine learning, transfer learning is suited for the problem of CNN image classifying [9]. Compared with training an entire CNN directly with all parameters such as initial weights fully assigned, a fine-tuned CNN model that views ConvNet as a fixed feature extractor introduces a solution for devices with limited GPU computing power, e.g. mobile phones. A fine-tuned classification model based on a pretrained CNN model maximizes the characteristics of nonlinearity within the ConvNets [10].

Rapid development in CNN architecture and performance provides opportunities for multiple classical image processing tasks besides image classification. Object detection is a long-standing problem in computer vision with focus on object localization [31]. Apart from object detection, multiple areas have combine convolution neural networks into their construction, i.e. object tracking, image captioning, text detection/recognition and pose estimation [32].

The most classical approach for object detection is based on the network of Regions with Convolutional Neural Networks (R-CNN). The model separates the problem of detection into two subtasks, namely object location prediction (object proposals) and image classification with CNN. The object proposal techniques should be viewed as an external module independent of the network constructions while the performance of the model mainly depends on the object detection module. However, the model proposed by the RCNN network is not an end-to-end realization, and the realization of end-to-end detector was achieved by the later network called Fast RCNN based on shared convolutional characteristics.

Extracting features from numerous images through ConvNets to achieve object kind classification is common in practice, but most of the work has focused on batch processing of the whole training image directly without generating confidence location of the image subpart as a preprocessing step, e.g. the dog. Faster RCNN involving dog breed detection aims to predict most of the normal image even if it does not include a dog object. An image without a dog is classified as an invalid input image [5]. Furthermore, preprocessing based on object detection provides a relatively reasonable regression bounding box which facilitates the classification of dog types with less disturbance on the input data flow into ConvNets, which clearly enhances the classification accuracy and ability [7].

Currently, most realizations of object detection are based on high-performance computers or clusters which contain significant computing performance. Although the demand of computing ability has reduced since the development of Faster RCNN detection architecture, a simple computer with GPU memory less than 4 gigabytes still cannot implement the Faster RCNN model with the VGG-16 image classifier. Moreover, a pretrained model with support for detection over hundreds of object types may not be implemented by any standard laptop [3]. Consequently, the implementation of Faster RCNN model and the number of supported object types for detection needs to be designed carefully.

The Apple products after Iphone8 provide a separate matrix computing module to support deep learning, hence the trends of placing deep learning networks on mobile devices are now more achievable [24]. However, computing modules can only support image classification that requires less memory and computing power, which cannot support object detection models with sizes larger than gigabytes. Especially for the Faster RCNN model, even the most advanced and high-performance mobile devices are unable to integrate the model into main memory. As a result, scalable server-side infrastructure is required.

The implementation of object detection in a mobile environment requires a scalable server-side infrastructure to be established to enhance the functionality and performance of the application. Obviously, the server needs to contain high performance computing power that enables the predictions for possible objects on raw image much to occur in a timely manner. Furthermore, the memory of the server should be large enough to read the whole model into main memory, and the server runs continuously for end users to utilize the object detection functions. Finally, the memory management strategy for mobile devices should be carefully designed to reduce the memory pressure from image classification, especially for real-time scenarios.

2. The Deep Learning Dataset

For most deep learning tasks in practice, the final performance of the neural network depends greatly on the quantity and quality of the input data. However, except for the image classification challenges with predefined training and testing datasets, almost all the industrial classification achievements require raw data from multiple sources to be considerably preprocessed. For dog breed detection, a sufficiently pure dataset with human-annotated labels is relatively difficult to obtain, although many open source dataset websites provide related dog breed images, most of them are not complete and hence unsuitable for classifying over 100 dog types, e.g. they are labelled differently.

2.1 The Source of Dataset

To accomplish the classification task of classifying and identifying multiple dog types, we acquire our data from several sources and unify their labels. The Kaggle challenge platform established a prediction competition focused on dog breed identification in 2017. This provides a strictly canine subset of ImageNet resources to serve as the dataset for the competition. The dataset contains 10k training data and classified images of 120 dog types. However, the dataset is not pure and unevenly balanced for each type [39]. The Stanford Dogs dataset is another resource that provides images of 120 breeds of dogs from around the world. It contains 20,580 images annotated by humans for each of the 120 categories. The bounding boxes is also offered in advance [40]. The ImageNet serves as a dataset with more than 15 million labelled high-quality pictures with over 22,000 categories. In this work, we extract 5k dog-related pictures with confidence labels that are consistent with the labels of the Kaggle dataset.

2.2 Image preprocessing

After collecting the required raw data, it is essential to consider the purity and availability of data from different datasets, hence we need to preprocess the collected image resources to enhance the performance of the model.

A. Filtering Images

For the raw image from the Kaggle dog and Stanford dog datasets, a small proportion of image data is invalid for many reasons. We filter the raw image based on the following principles. Firstly, the original scale (both the width and height) of raw images need to be larger than 128 pixels, since most of the CNN models require the input image size to be 256*256 pixels. Although we can reshape the image, this process involves some inherited loss of the quality of the original image, which influences the classification performance. Apart from this, the ratio between the width and height of an image cannot be too large or too small, since the reshaping process should maintain valid image characteristics as much as possible. Finally, to avoid too much disturbance and non-related image content, the percentage of useful content within an image must be greater than a fixed score, otherwise the ConvNets has an extremely high probability to learn parameters that are totally useless resulting in entirely mistaken templates or feature maps, and thus lead to false classifications [13].

B. Clipping Images

Although the original dataset, especially for the Stanford and Kaggle dataset provides mostly accurate labeling for raw images, the quality of the image varies significantly, especially for images with a higher proportion of image as background. For example, Figure 2 shows an image of an American Staffordshire terrier that has been labeled correctly in the Kaggle dog dataset. However, the proportion of useful information (namely, the actual dog) is too small and is overwhelmed by the background. If this image is used as the input image, the convolutional network would learn parameters mostly for the background which would influence the classification of this and other dog species.

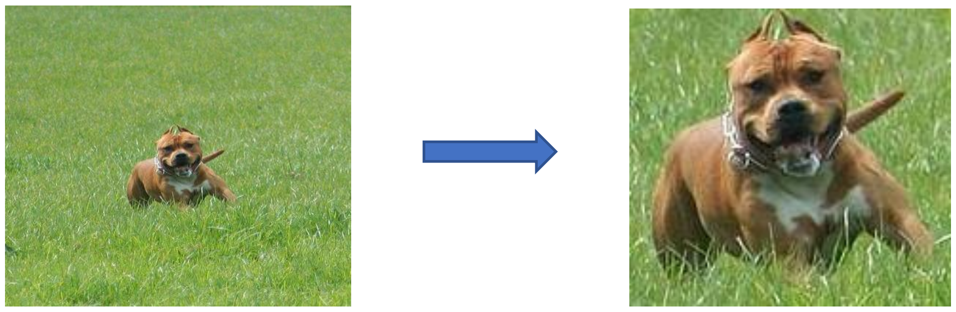


Figure 2: Clipping raw image to avoid useless image features

To avoid this program, we clipped the image according to the bounding box information provided for the Stanford dog dataset, to generate relatively more robust input training images [16]. However, for the Kaggle dataset and images from Image Set, this process need to be completed by the human to maximize the applicable information within the original image.

C. Purifying Images

Correct labeling is essential for any image classification tasks, wrong labels not only influence the classification performance, but it also affects the final gathered parameters from the whole training images. Although the original images from these datasets are mostly labeled accurately, the best approach to constructing an entirely pure dataset is based on human (manual) labeling. i.e. there are no fully-automated purifying approaches for dog breed identification at present. To enhance the quality of the input dataset, human preprocessing for image purification is applied for all images to minimize misclassifications.

D. Image Adjusting

In practice, a relatively feasible approach to solving the sparsity of input dataset is rotating or flipping images, although this approach can only provide limited performance increase [20]. This approach could be viewed as a naïve approach to increase the amount of data, however, the heuristics behind this is that flipped images increase the reception field of the feature map and the suitability of the model for all angles of the images. In Figure 3, we rotate the image 90 degrees in the left and right direction. This approach is only needed for some breed types with a limited quantity of images.

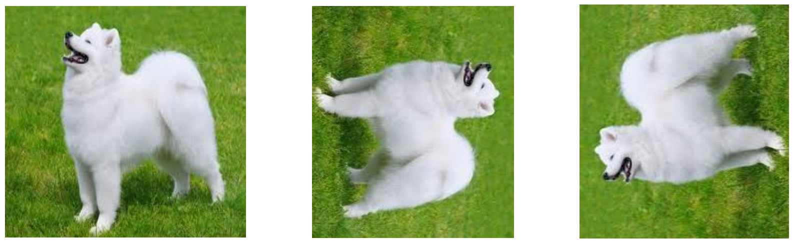


Figure 3: Flipping images to increase the input data quantity

E. Reshaping Images

To facilitate the slide window convolution in the training and testing process, all input images need to be reshaped to a similar scale to ensure the convolutional layers can generate feature maps of the same size for any input images [7].

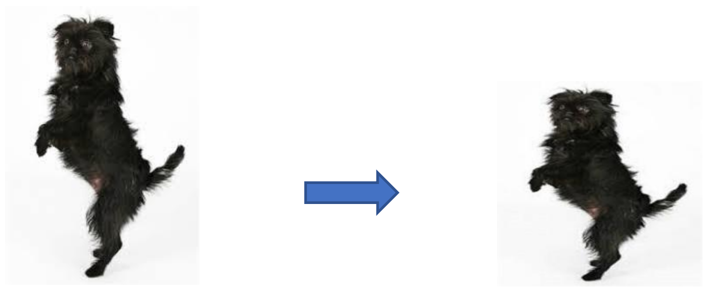


Figure 4: Reshaping images to 256*256 pixels

In Figure 4, we reshape images to 256*256 pixels needed for the training process. We randomly crop the input images to 224 * 224 pixels to maintain the stochasticity of the raw data needed for most of the classic ConvNet models.

F. Image Preprocessing – Mean subtraction and Normalization

To facilitate and enhance the robustness of the training process, the original input images need to be transformed into a uniform data structure [11]. The preprocessing mainly consists of mean subtraction and normalization. Mean subtraction aims to keep the sum of all RGB scores close to zero by subtracting the average of the score. The reason for mean subtraction is that the classification depends on the relative difference between image pixels rather than the absolute pixel score. A too large average may lead to extremely high parameter gradients [8].

Normalization is the process of making the variance of the image score close to one, thus the data in three dimensions could follow the normal distribution with a zero-average score. If the variance differs significantly in three dimensions, the optimization process is made much more difficult. The process of normalization is required in the training process for most deep learning architectures [8].

G. Dataset Separating

After accomplishing image preprocessing, about 30k eligible input images were used as the training and testing dataset. We separate the set of images into a training set, a development set and a testing set in the ratio 8:1:1.

3. Classification Framework

The ConvNet architecture of our network is based on the classical network AlexNet with a fine-tuning optimization process as presented in Figure 5. The CNN framework consists of eight hidden layers, with three Fully-connected (FC) layers, five convolutional layers, a single input layer and two output layers for loss and accuracy [2].

3.1 Structure of CNN model

We briefly describe the main features and layers of the network structure starting with the input layer, followed by the stack of convolutional layers and fully-connected layers and the hidden activation and pooling layers. The final output is derived from the last two layers, namely the loss layer and the accuracy layer.

3.1.1 Input Layer

The input of the whole ConvNets is the training dataset and the development dataset. As noted, raw images are preprocessed into uniform 256*256 pixels with correct labels for each image. Mean subtraction and normalization is applied to the whole dataset. In addition, a random extra crop is applied to the preprocessed image to 224*224 pixels to increase the suitability of the model. The batch size of the input image is set to 256 based on the capability of the GPU device. Finally, a LMDB file that contains the matrix of the whole images is used as the input to the model.

3.1.2 Convolutional Layer

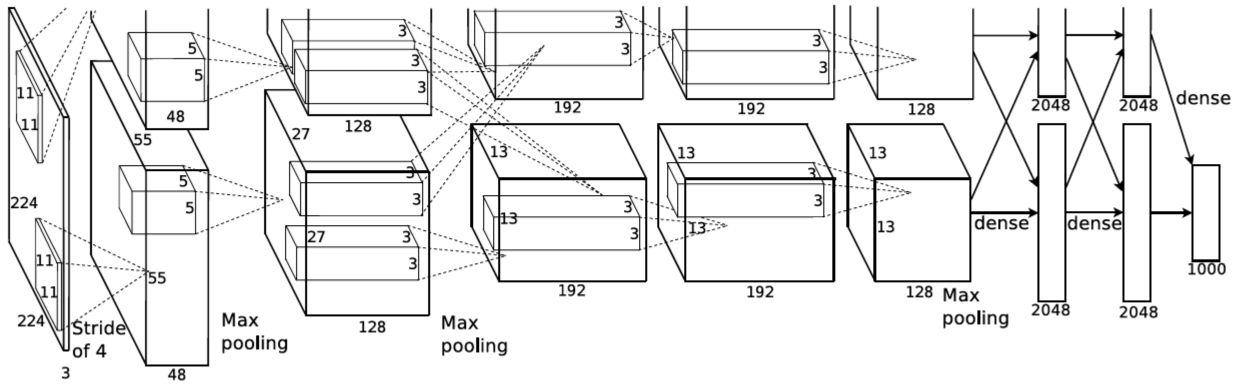


Figure 5: ConvNets architecture between two GPUs [2]

The whole network contains five convolutional layers where all of the weights for the ConvNet layers follow a Gaussian distribution with the bias weights initialized as zero. The first convolutional layer filters the 224*224*3 input images with 96 kernels of size 11*11*3, and the stride is set to four which represents the distance between the center of the reception fields within the feature map. The second convolutional layer creates a dot product with each feature map output from the first convolutional layer with kernel size 5*5*256. Similarly, the third, fourth, and fifth layer both accept the output from the previous layer with kernel size of 384*3*3, 192*3*3 and 3*3*192. It is noted that there is no pooling or normalization layer between the last three convolutional layers. The intention of the convolutional layer is to capture existing features from the input data with each kernel represented as a reception field. The number of accepting features is decided by the kernel size and stride.

3.1.3 Fully-Connected Layer

There are three Fully Connected (FC) layers placed after the convolutional layers. As before, all weights are initialized to follow a Gaussian distribution to reduce the number of iterations in the optimization process with bias factor set to zero. The first two fully-connected layers generate 4096 outputs, where each neuron is connected to all of the outputs from the previous layer. The last fully-connected layer generates 120 outputs which corresponds to the 120 dog breed types and hence the output is fed to the final SoftMax function to generate the corresponding loss and accuracy [2].

3.1.4 Max Pooling Layer

A pooling layer is added to the output of the first two convolutional layers and the first fully-connected layer. The reason behind this that not all of the parameters learned by the Conv or FC layer are meaningful, hence we filter some of the parameters through the pooling function to prevent overfitting which acts as a regularization function. Our network applies the max pooling function with a kernel size of 3*3, which does not disrupt the number of feature maps from the previous convolutional or FC layers.

3.1.5 Loss Layer and Accuracy Layer

The final output of the model consists of the loss of each iteration and accuracy for every predefined iteration. These two layers accept the 120 outputs from the final fully connected layer and calculate the corresponding loss and accuracy. For the loss layer, we apply the most common function type of SoftmaxWithLoss. The accuracy layer involves data from the development dataset.

3.2 Network Algorithm

For the constructed framework of the ConvNets with the initial parameters, we need to learn features from the input images to optimize the parameters that could maximally classify dog breeds correctly and enhance the performance. This process is accomplished by the optimization algorithm with associated loss function.

3.2.1 Loss Function

The performance of the training process is measured by the loss value. This represent the degree the model can find the best parameter that is able to separate the classes. The loss function also informs how well the classifier works. The loss function could be viewed as a predefined scale to calculate the loss which can be used to optimize the parameters. Multiple loss functions can be applied to measure loss, e.g. a full loss function, an SVM loss function and a SoftMax loss function. For our network we utilize the SoftMax function. This is given by:

$$soft\max(x) = normalize\left(\exp(x)\right)$$
$$soft\max(x)_i = \frac{\exp(x_i)}{\Sigma_j \exp(x_j)}$$

The function calculates the negative log of the probability of the true class, where s_i is the score function calculated by multiplying the input value with the weights plus a bias for the i_{th} input data. It is noted that the loss function is only utilized as a measure metric and has nothing to do with the optimization process.

3.2.2 Nonlinearity with Rectified Linear Unit Function

For any convolutional layer or fully-connected layer, an activation function or nonlinearity function is applied when data comes in. The activation function provides the ability to enable the parameters classify the types nonlinearly, and hence reduce the effect of linearization. A CNN classifies an incoming class by separating it from all other pre-separated classes, however, the original parameter can only segment the classes linearly, hence it cannot meet the demands of multi-class classification. As a result, a nonlinear function is added to serve as a soft classifier.

Multiple available activation functions are applied in the area of deep learning with different characteristics and drawbacks. Compared with the tanh and Sigmoid activation functions, the ReLU function *f(x) = max (0, x)* contains the characteristics of non-saturating nonlinearity in terms of training processes with a stochastic gradient descent. Both the tanh and Sigmoid functions would kill the gradients when saturated neurons happen and are relatively computationally expensive. The ReLu function enables the loss converges to occur much faster in practice.

3.2.3 Stochastic Gradient Descent Optimization with Momentum

Optimization is the process of minimizing the loss calculated by the loss function, thus supporting the process of discovering the weights that perform best [23]. The gradient is the vector or partial derivatives along each dimension. The direction of steepest descent is the negative descent, which drives loss down consistently. The key hyperparameter that influences the model in finding the most eligible descent value is the *learn rate*. This measures the speed used to determine the lowest loss that provides the models best performance. A relatively high learn rate makes the model miss the minimum and thus not decrease the loss, while too low a learn rate makes the converging process extremely slow and wastes computing resource.

To combine the advantages and avoid the drawbacks of high or low learn rates, we applied the momentum optimization approach to the stochastic gradient descent in our model. The idea is that we maintain a velocity with predefined iterations. Initially, we set a relatively high learn rate to make the gradient step into the right direction swiftly, and at every defined time step we decay the learning rate according to the current velocity. In this case, we allow our network to step fast to rapidly find the global minimum [8].

3.3 Fine-Tune the ConvNet with Transfer Learning

The optimization algorithm allows the model to find the global minimum, however, the dataset size is still not sufficient to generate an optimized performance. In practice, it is not possible to create an entirely new convolutional network especially with such small datasets [12]. Therefore, we take a pretrained ConvNets for 1000 classes and over 10 million training datasets as the original fine-tune model. However, here we replace the last fully-connected layer with another layer defined to support back-propagation. In this way the pretrained parameters can be utilized along with more image information captured by the model. The heuristics behind this method is that the earlier features of the ConvNet contain more generic features, while the later layers contain features that are more specific to the classes of the original dataset.

In this case, a much smaller learn rate is required for ConvNet weights that are being fine-tuned. The final FC layer can be viewed as a linear classifier that calculates the new optimization score of the new dataset.

4. Object Detection Framework

Deep learning provides a new avenue of research for object detection. This has shifted most traditional computer vision approaches that use neural networks to solve the problem of finding objects within images without a clear understanding of the characteristics of the raw pictures. The first successful realization of object detection with deep learning is the Region-based Convolution Neural Network(RCNN) [26]. This utilizes a pretrained CNN model to predict based on thousands of proposals selected the focus on selective search and subsequently selecting proposals higher than a given threshold. However, this approach involves extremely expensive computation resources. Another model called Fast Region-based Convolution Neural Network(Fast RCNN) solves this problem by applying a selective search only on the final convolutional feature map, thus the previous parameters can be shared by all proposals [3].

However, the proposals generated by selective search are mostly random and many of the proposals are not reasonable. In our network, we apply the Faster RCNN model and include the process of generating proposals as part of the training process. This results in much fewer proposals with high possibility provided along with their location in the image. This approach achieves almost cost-free proposal computation process within a deep convolution neural network.

4.1 CNN Construction

The realization of the Faster RCNN model is based on the basic CNN architecture, which uses the final feature map of the CNN model to generate proposals and identify bounding boxes. For our object detection model, we utilize the classical VGGNet as the CNN classifier. The model consists of 19 learning layers where the kernel size is consistent for all convolutional layers (as 3*3 with stride 1). Furthermore, the model automatically reshapes the image to multi-scales and generates multiple ConvNets. It utilizes the model ensembling to predict the image where the final class of input image is the class with the highest score provided via these generated ConvNets.

Moreover, a deep model has a much higher probability to produce useless parameters. To tackle this, a dropout layer is added to the model to actively abandon some of its parameters. The reason for applying the VGG network as the training network is due to the fact that more parameters are required for the proposal generation process.

4.2 Region Proposal Network (RPN)

The proposals are generated through a region proposal network, which is constructed by adding a few convolutional layers that perform bounding box regression and provide “objectness” scores simultaneously. Thus, the RPN network can be viewed as a full convolutional network (FCN) that enables end-to-end proposal generation. The process of unifying the RPN with the convolutional neural network is to provide a fine-tuning procedure for object detection and proposal generation. The first module of the Faster RCNN applies deep FCN to generate regions and the second module detect the proposed regions.

The RPN accepts the feature maps from the previous neural networks, and outputs multiple reasonable object proposals with their corresponding objectness score. This is accomplished by sliding multiple windows with multiple scales(1:1, 1:2, 2:1) across these incoming feature maps where the sliding window is mapped to a lower-dimensional feature (256-d) of the original dimensions [3].

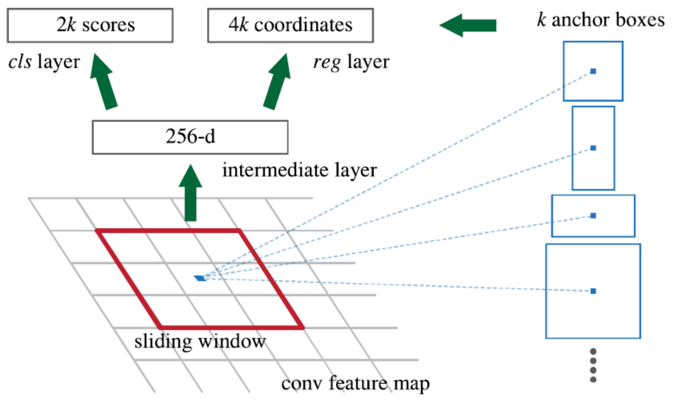


Figure 6: The Process of Region Proposal Network

The sliding window enables the parameters within the FCN to be shared across different spatial positions. Furthermore, each anchor proposal of a sliding window generates four regression parameters, which denotes the coordinates of the bounding box, together with two score parameters, which represents the score of objectness as shown in Figure 6.

4.3 Loss Function

The RPN outputs six parameters with four regression coordinate parameters and two class label parameters, hence an extra loss function must be applied. The idea of Intersection-over-Union (IoU) represents the overlap between the predicted box and the ground-truth box. A region is reasonably proposed if it contains the highest IoU value or the IoU value is higher than 0.7 [3]. Consequently, the loss function contains two parts. The first part is the loss of the predicted probability, i.e. an anchor proposal is an object, and the second part is the regression loss reflecting the distance with the ground truth box.

4.4 Training Region Proposal Network

As the RPN is applied to a basic convolutional neural model, the training process is accomplished through back-propagation and adopting stochastic gradient descent. An extra loss function needs to be applied to optimize the object detection and bounding box regression process. The weights for all additional layers added is based on the CNN model. This is initialized randomly to follow the Gaussian distribution with a standard deviation of 0.01, however, all other parameters are initialized by the pre-trained convolutional model based on the dataset from ImageNet. The hyperparameters need to be determined carefully. A relatively normal learn rate of 1e-3 is utilized only for the first half mini-batches, and the learn rate for the subsequent mini-batches is set to 1e-4 to rapidly identify the lowest loss value.

4.5 Implementation Details

Compared with the pure CNN model whose size of the input image is set to the 256*256 pixels to facilitate the random crop, Faster RCNN models restrict the shorter side of the input image to 600 pixels for both the training and testing images. Furthermore, the size of the sliding anchors is set to 128*128, 256*256, 512*512 with the scale of 1:1, 1:2, and 2:1 [3].

5. Experiments and Analysis

The overall training process is separated into two parts depending on whether the fine-tuning process is combined with the overall training iterations. Apart from the fine-tuning process, we implement the training process for different class sizes and data sizes and2 consider different learning rates to express the influence of this hyperparameter.

5.1 Training towards different size without fine-tuning

This section introduces the training process that combines different sizes of the data and classes. We start our training with a small training set which contains 30 kinds of dog. Each kind has about 80 pictures, so the total training set has about 2400 samples. The learn rate was initialized to 0.01 and the batch size is set to 256 for 5000 iterations. The purpose of this stage is to provide a general view of how to train a network on the platform and to check whether the hyperparameters are set appropriately. To show the procedures of the training process, we display the loss for every 50 iterations and evaluate the model on the test set for every 1000 iterations and calculate accuracy. It is noted that the aim of the testing part is only to show the performance of the model, i.e. it does not affect the training process. The result is shown below in Figure 7.

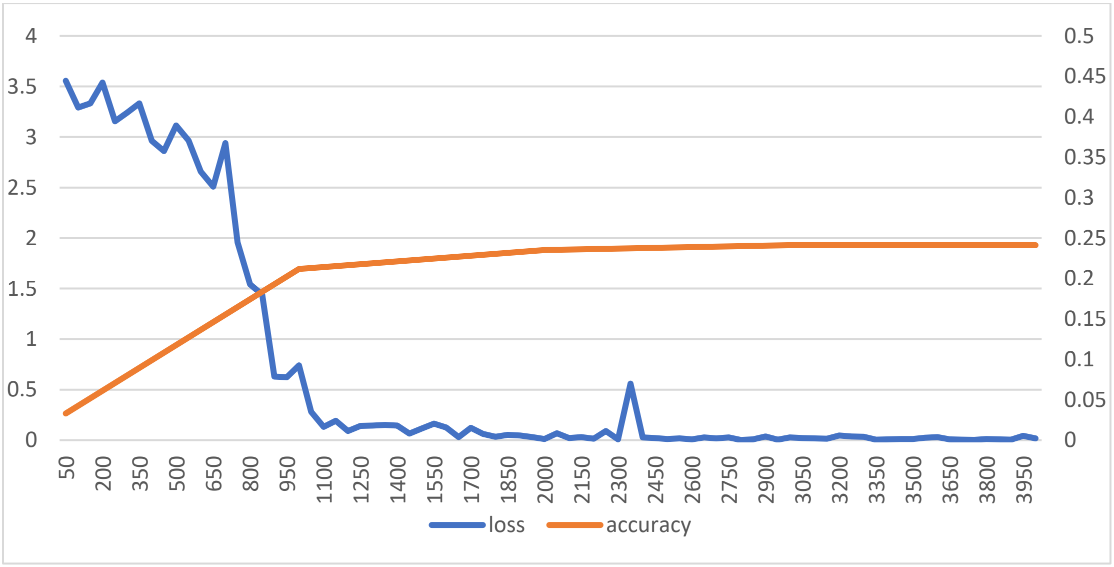


Figure 7: Training with class=30 learn rate=0.01 samples=2400

As seen in Figure 7, the loss reduces to a very low level after 1100 iterations of training and stabilizes at almost 0 after 1850 iterations of training. The accuracy reaches 25% which is relatively poor but compared with the baseline 3.33%, the results of 25% shows that the model works well. We then increase the training set to 50 classes and increase the sample to 4000. The result is presented below in Figure 8.

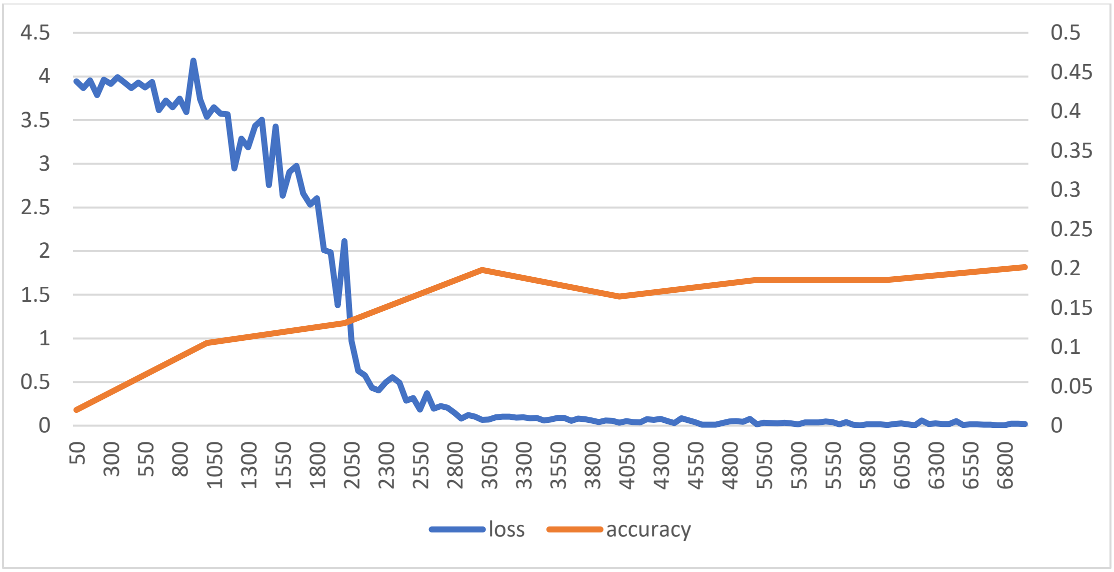


Figure 8: Training with class=50 learn rate=0.01 samples=4000

As seen the model reaches a low-level loss after about 2500 iterations and loss reduces to almost 0 through 4000 iterations. The new dataset needs above 2.5 times longer time to train the model although the size of the dataset increases (about 170%). The accuracy increases to 19%. Comparing the result with baseline (2%), we can say the model and setting work. We predict that if we train the whole dataset which contains 100 classes of dog and has 8000 samples, it may need 6000 iterations to train the model to get a stable low-level loss. The result is presented below in Figure 9.

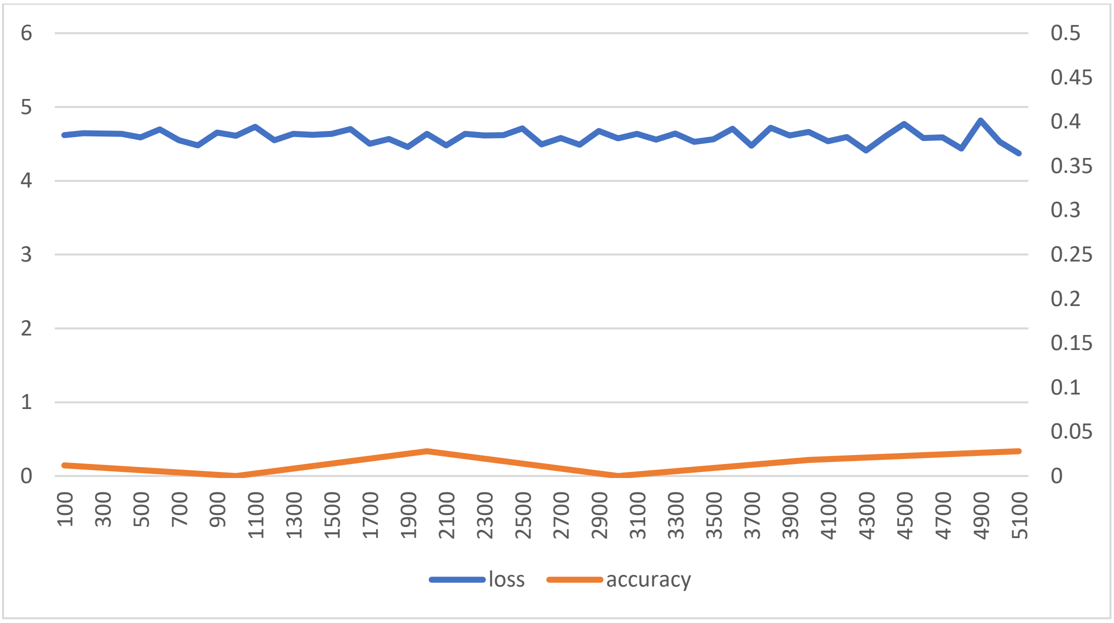


Figure 9: Training with class=100 learn rate = 0.01 sample=7600

The loss does not reduce as our predicted. It keeps steady at 4.7. As seen, the accuracy does not increase, and it is close to the baseline which means that the model's prediction is a random guess. In this situation, we say that the model is not trained. We consider, what happens if we increase the dataset size to 120 classes (the whole training set).

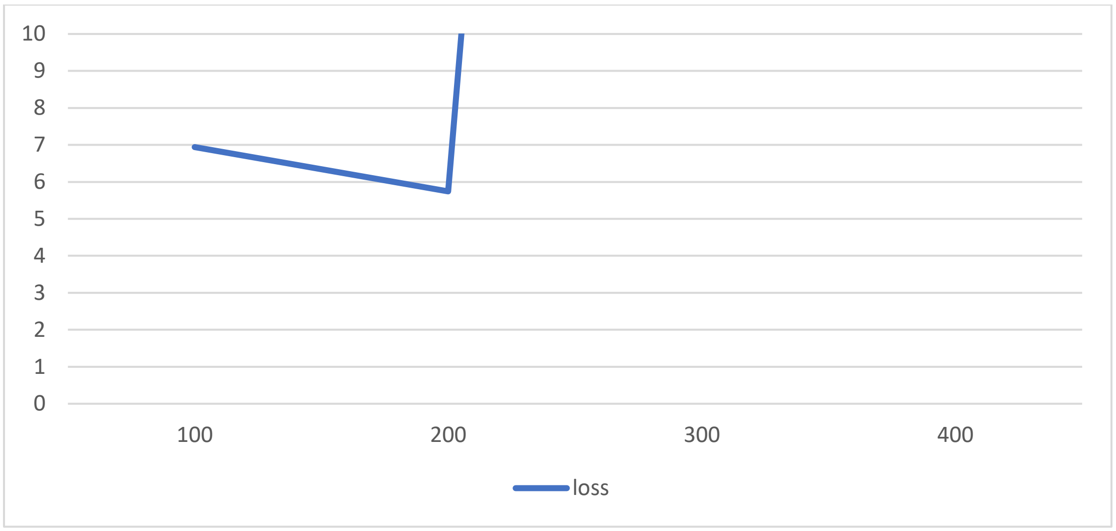


Figure 10: Training with class=120 learn rate=0.01 samples=8000

After 200 iterations' training, the loss increases from 5.7 to 87.3 and keep at this very high loss level following training so we stop the training. The result is far beyond our expectation, i.e. the configuration seems inappropriate for classes of over 100 datasets.

Clearly, without a fine-tuning process, the performance of the model decreased significantly when the number of classes and data increases. Multiple reasons would drive the performance down. First of all, the more classes and raw images we fit into the model, the longer time the model takes to reduce the loss. Consequently, too much data may increase the difficulty for the model to find the global minimum. Furthermore, the number of parameters required to train the model is significantly increased with too many classes, and the baseline of the model drops clearly hence the interference increases in detection of the dog breed types.

5.2 Training with different learning rate without fine-tuning

Once we've compared with the performance of different training processes with regards to the different sizes of classes, the influence of the different learning rates needs to be taken into consideration. To compare the model with various learning rates, we set the learning rate of the model from Figure 10 to a lower degree, i.e. 1e-3. The result is presented below in Figure 11.

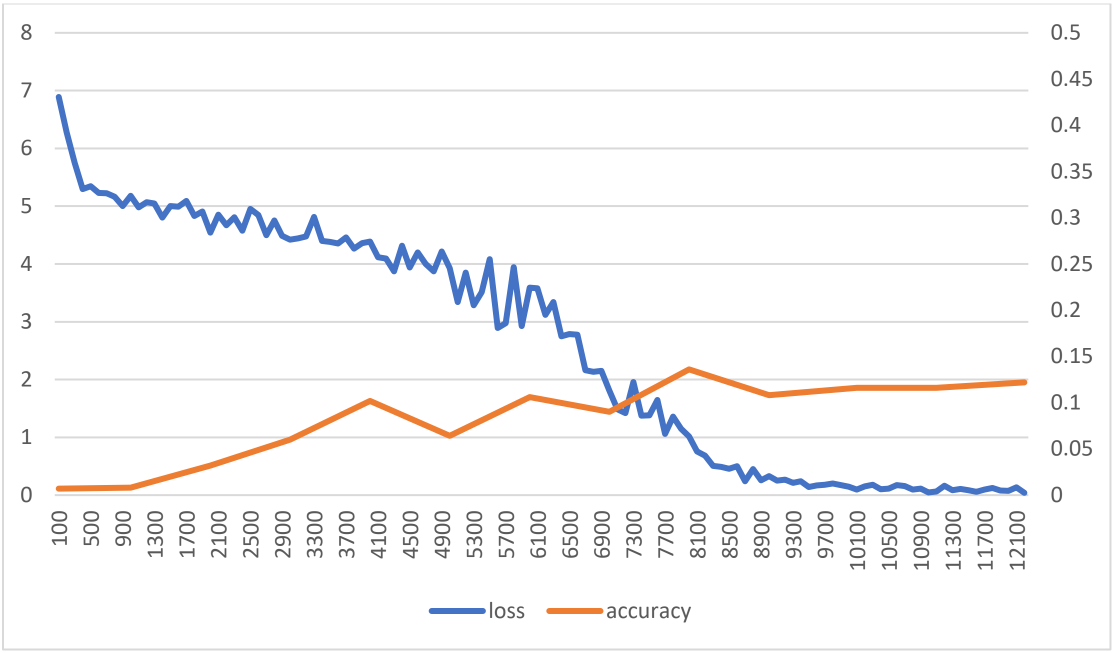


Figure 11: Training with class=120 learn rate=0.001 samples=8800

The model was trained successfully with 0.001 learn rate. After 10,000 iterations in training, the loss reduces to a low level and the accuracy increases to 12%. The lower learn rate works for 120 classes. We compare the results of the 0.01 and 0.001 learn rates on the previous dataset (30&50 classes) and the loss charts are presented below in Figure 12.

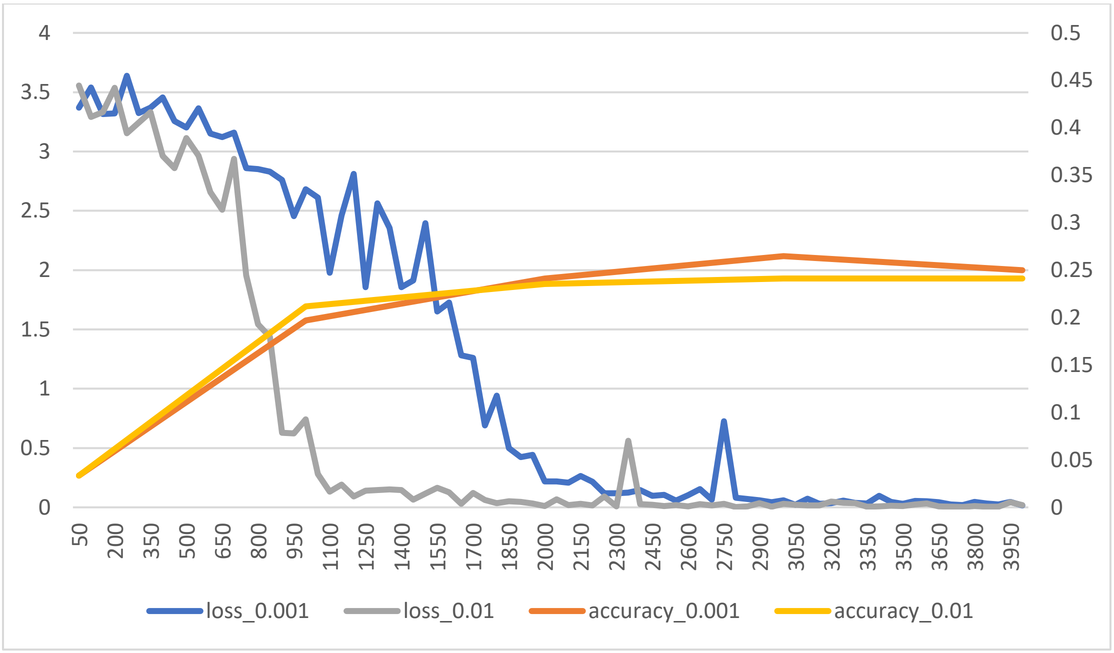


Figure 12: Training with class=30 learn rate = 1e-2/1e-3

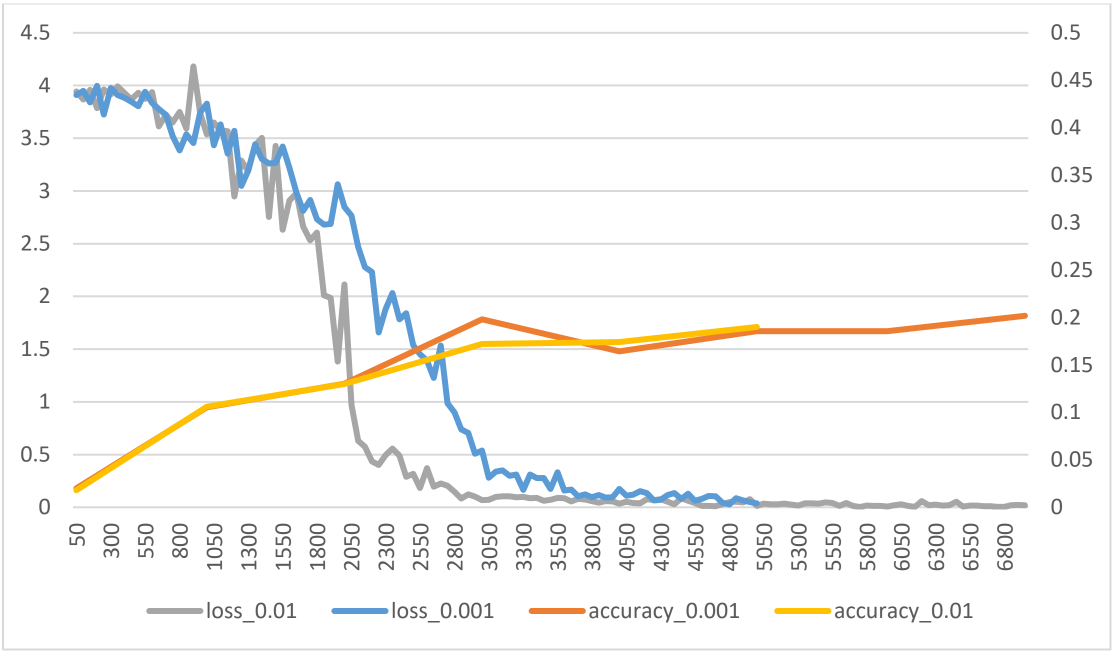


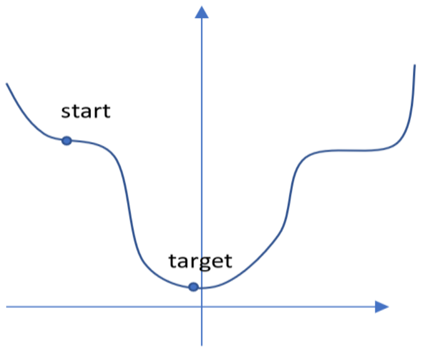
Figure 13: Training with class=50 learn rate = 1e-2/1e-3

The 0.001 learn rate can also be applied to the smaller dataset with almost the same result as the 0.01 learn rate. However, even though we achieve a similar result, the lower learn rate leads to a longer training time, so for the smaller training set, a higher learn rate will be better.

Through the comparison of training processes with different learn rates, a clear understanding is that a higher learn rate would drive the model to find the better parameters more quickly. However, the accuracy of the final model may be influenced by a relatively higher learn rate. For a lower learn rate, the model definitely finds the better parameters that are suitable for the models compared with a higher learn rate, however, the training time is significantly increased as a result. The trade-off between performance and time needs to be considered when performing image classification.

5.3 Impact of loss function

To give a simple explanation of the learn rate, we present two graphs to show the loss function in two dimensions (in fact the loss function has millions of dimensions) in Figure 14.



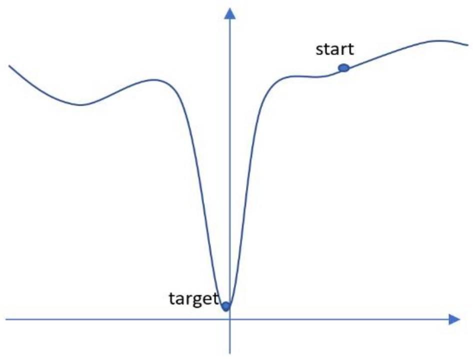


Figure 14: a) Simple loss function **b) Complicated loss function**

For a simple loss function starting at the "start" point, we can change the parameters to go to "target" point as shown in Figure 14:a. After calculating the gradient, the parameters are changed to follow the direction that can reduce the loss to eventually reach the target point. For a larger dataset, if the network wants to match the training samples, it will produce more parameters and the loss function will be more complicated. The loss function may look like Figure 14:b.

The target point is in a valley and because the loss function is complicated, the loss function will rapidly fluctuate. The entrance to the target point will be tight and if the learn rate is too high, the point may not go into the valley, and step over to the other side and then step back. That is why when we train the 100 classes using 0.01 learn rate, the loss cannot converge. If the loss function is even more complicated, the point may go out of this graph to an unexpected place, and that is why for 120 classes, the loss increases to 87.

5.4 Performance Enhancement

5.4.1 Training with Image Preprocessing

Based on the above we identify that the learn rate is key to successfully train the network on a dataset of 120 classes, but the accuracy is not acceptable (12%). Because training whole dataset is time-consuming, we mainly research the development of our network on a dataset of 50 classes.

Firstly, we found that the dog in some training pictures comprises only a small part of the image and the background may distract the network's training. We thus crop the picture and retrain the model. The accuracy increases by 2%.

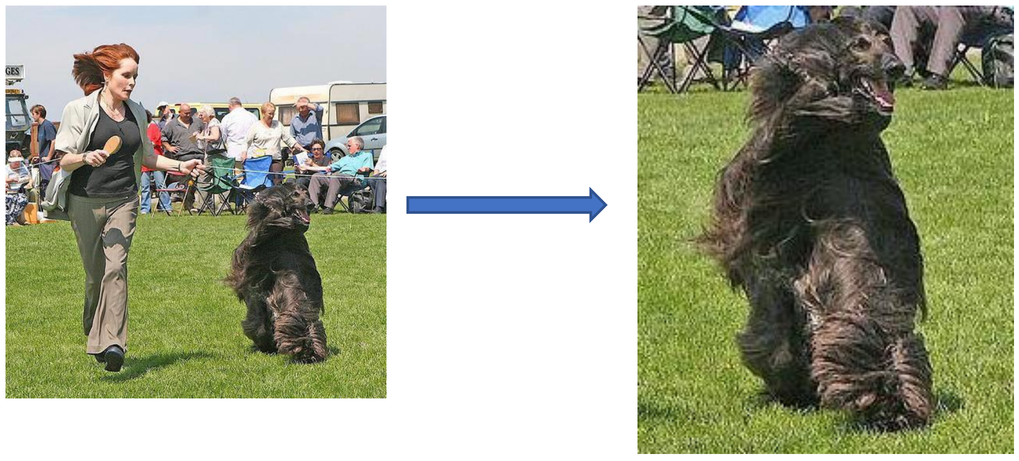
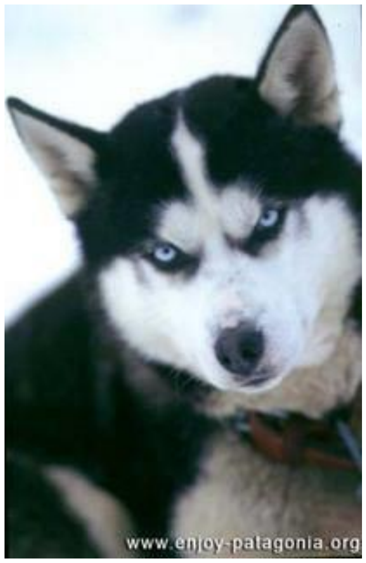


Figure 15: Clipping Images

The accuracy is still not acceptable, so we collect more pictures for each kind of dog. The samples for each dog are increased from 80 to 250. Because the number of samples increases, we use 0.001 as the learn rate to train the dataset for 50 classes. This time the accuracy increases to 25%. But this result is still not enough. Many dogs are very similar and are even hard for humans to distinguish them, e.g. Husky and Eskimo as shown in Figure 16.



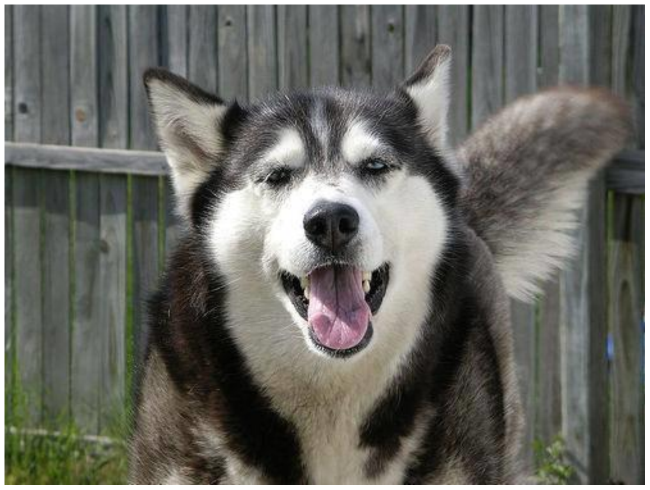


Figure 16: **Separating similar dog classes**

As such we pick 50 kinds of dog that different from each other and train the model. The accuracy increases again to 27% but this still not enough.

5.4.2 Fine-tuning the Network

Based on the above, it seems that training a model from the start is not a good idea, hence we try to use the pretrained model to initialize the network parameters. The pretrained model is trained by other organizations for a long time (weeks) using a powerful computer and the dataset that containing many different classes with a great number of samples. What we do is to train only the last layer of the model. As the convolutional layer extracts the features from the picture and the fully-connected layer categorizes the picture, we only need to adjust the last fully-connected layer to make the model fit our data. The result of using fine-tuning on the dataset of 50 classes is presented below.

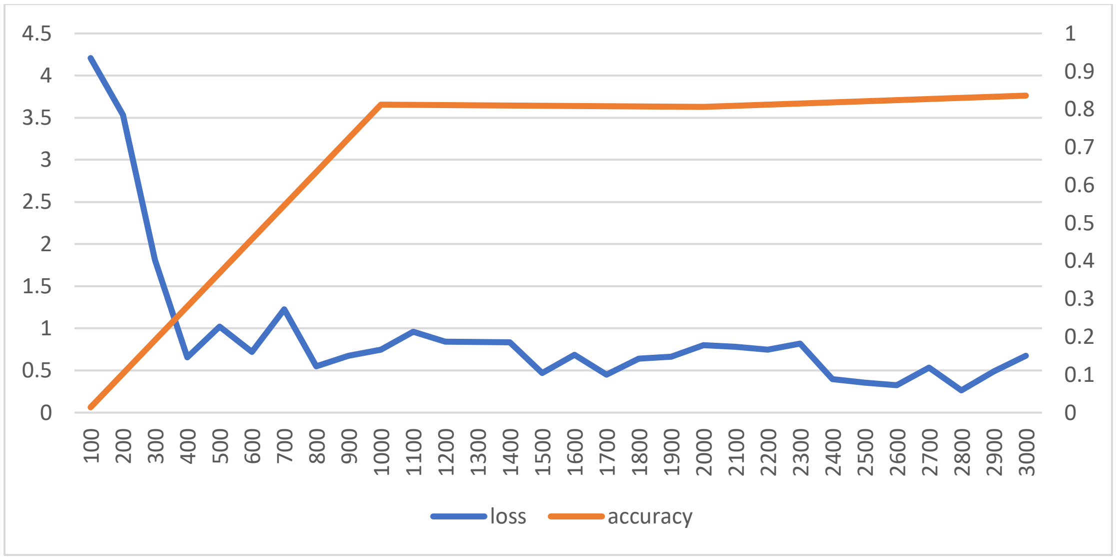


Figure 17: Fine-tuning 50 classes with learn rate = 1e-4

The result of whole dataset (120 classes):

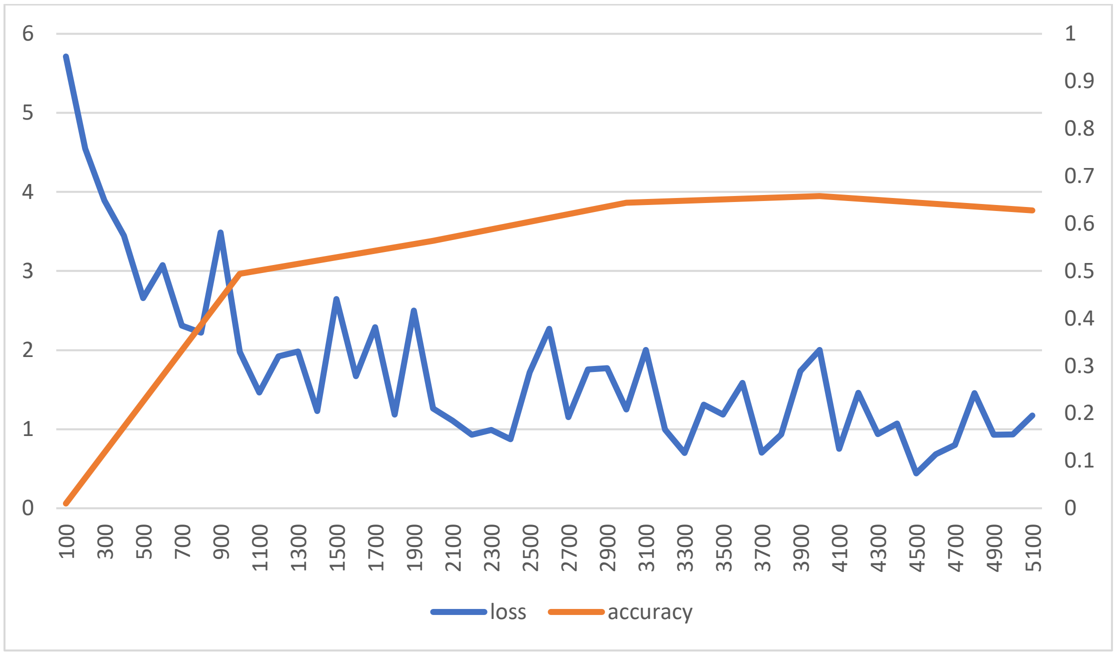


Figure 18: Fine-tuning 120 classes with learn rate = 1e-4

Fine-tuning shows a considerable increase in the accuracy and requires a very short training time. For 50 classes, the accuracy can reach 85%, and for 120 classes it reaches 63%. This result is good enough considering that some dogs in 120 classes are so similar that it is even hard for humans to recognize them. However, the loss cannot reduce to a very low level compared with the previous training. This is likely caused by training only one layer makes it hard for the network to match the training set.

Noted that the training set of the pretrained model does not contain our dataset, but it still shows a good adaptation on the data. The principle behind this is that the pretrained model is trained on a huge dataset, so on its convolutional layer, it already has the ability to extract useful features to tell the difference between pictures. Thus, we only need to train the fully-connected layer to categorize the picture. We also try to train the last two and last three fully-connected layers (there are three FC layers in AlexNet), and training only the last layer can get the best accuracy. For previous training, even if we get a very low loss, we cannot get a good accuracy because the network matches the training set too much. The dataset is still not enough if we want to train a model from the beginning (even 250 pictures for each category). The scale of the training set is the key requirement to train a good network. A larger dataset requires a lower learn rate but at the cost of a longer training time.

6. Implementation on Mobile Environment

The objective of this paper is to provide an effective and available realization of dog breed classification and object detection within a mobile platform (IOS). The key design issue involves the linkage and between several correlated View Controllers, and the transformation of the ConvNets models to adapt to the special running environment of mobile devices. Furthermore, the User Interface must be designed to be friendly to enhance the overall usability. The application should be designed efficiently to reduce the occupancy of the limited main memory on mobile devices with the release of unnecessary resources.

The IOS application contains seven views with one navigational View Controller, the main view is responsible for controlling access to different views. Two views are responsible for implementing the dog breed classification service and the object detection service, and there is direct communication between these two views. Another four views are designed to facilitate the utilization of the mobile application for end users.

Memory management is crucial for mobile devices, especially for such a large network model (about 500MB). Each time the application is used for detecting and predicting a new input image, the previously processed image must be released from memory. This requirement is quite important for dynamic dog breed classification.

6.1 ConvNets on IOS

The fine-tuned convolutional model is relatively small compared with Faster RCNN models thus it can be transformed and placed on IOS environment. Each time an image is an input, the model classifies the image based on the pre-trained parameters, which generates a score of possibility for all supported dog breeds. However, for real-time or dynamic breed identification, we implement the AVFoundation packages to capture real-time images automatically and feed the image into the model with minimal intervals. For every input image, the model takes about 0.1 seconds to generate the corresponding results.

6.2 Object Detection on IOS

However, the model trained through Faster RCNN is too large for mobile, and there are no available devices that are able to provide the satisfactory environment for Faster RCNN models. To solve this problem, we establish a Cloud-based server supporting object detection services and record the detection history. A mobile client is realized through the GSDAsyncSocket interface services.

The interface between the object detection view and the dog breed classification interface involves the communication of data. The detected dogs with a probability of higher than 90% will be sent to the dog breeds identification interface for classification.

The whole application is designed to be user-friendly by providing a simple instruction interface and error warnings. A detailed introduction of the supported dog breeds is provided by the application that allows users to check the performance of the model.

7. Conclusion

This paper introduces an implementation of a dog breed image classification application and a fast object detection network that utilizes deep learning in a mobile environment. The dog breed recognition model supports the classification of 120 types of dog breeds. A final realization on the mobile environment implements the dog breed detection model and fast object detection model.

To enhance the final performance of the image classification model, several approaches and multiple training processes have been applied to ensure the model could provide satisfactory performance for end-users.

First of all, the quality and quantity of the image dataset influence the performance of the model significantly. To enhance the quantity of the raw images, we collect dog breed images from multiple sources, and support image adjusting methods to ensure that all of the classes contain the same number of images, thus the model will not overfit towards classes with relatively more input images. Multiple preprocessing methods have been utilized to enhance the quality of the data, image filtering and purifying are used to ensure that every image is eligible to be fit into the model and is correctly labeled with their true classes. The clipping process significantly helps to reduce the inference from background pixels, which is shown in the increase of performance for models. Another key preprocessing method adopted is mean-subtraction and normalization, which will not influence the performance of the model.

The choice of various learn rate impacts the performance to some degree, whereby a higher learn rate makes the model generate a normal performance more quickly and the number of iterations needed to converge is small. However, only a lower learn rate ensures the model discover the global minimum rather than finding a local minimum. The disadvantage is that much more time is required to train the model.

Finally, all of the above approaches are not enough to generate the best performance for dog breed detection. Although the hyper-parameters have been adjusted to fit in the dog breed detection conditions, the number of input images and the performance of the devices have impacts on the experience of end-users. Thus, the approach of transfer learning has been applied to the training process including fine-tuning the pretrained model with additional dog breed images using a much lower learn rate. This enables the convolutional neural layers to capture the characteristics and information from the raw images directly.

The realization on the mobile platform needs to take memory management into consideration, i.e. an application that utilizes too much computing power and memory will affect the availability of the system. The application has thus been designed with these considerations in mind.

8. Future Work

There are several shortcomings in this paper needed to be addressed for a final dog breed detection model. To make the application more general and tolerant, multiple feasible approaches and techniques could be combined together to enhance the performance.

First of all, the dataset is still too small to generate a satisfactory model for dog breed recognition. Deep learning requires numerous data to be fit into the model to learn useful parameters. The best image classification models currently utilize millions of raw images to train the model. However, the limited amount of data is not the only issue we need to improve, the computing power is still insufficient to train robust models. A better GPU device with larger memory could read a larger batch of images each time. Parameters could also be shared by more raw images to help the model converge more rapidly.

Apart from this, the architecture model could be improved. There are many parameters learned by our model that are useless for image classification tasks. Multiple hyper-parameters could also be set more accurately. For example, initial weights could be given that are suitable for the characteristics of the dog breeds rather than just initializing them randomly. These parameters could also be learned through multiple iterations.

For object detection tasks, there are some drawbacks in the training data, the bounding box of the input data could be labeled more accurately. Furthermore, the whole application is not an end-to-end version, since the object detection and image classification have to be separated. However, with enough dog breed images and predefined bounding box labels, a mobile application could combine the two tasks together rather than involving the transfer of image data. Finally, the time for object detection task is still too long for end-users. A better detection model could be developed with fewer parameters to give more accurate results in less time.

A video demonstration of the mobile application for dog breed classification and identification is available at: https://www.youtube.com/watch?v=0jbydCXJraw.

Convolutional model and training process is available at: https://github.com/wenbinc-Bin/dogCNN.

IOS realization is available at: https://github.com/fangwu4/iOS-dog-.

Reference

[1] Shaoqing, R., Kaiming, H., Girshick, R., & Jian, S. (2017). Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Transactions On Pattern Analysis And Machine Intelligence, (6), 1137.

[2] 2014. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." 2014 IEEE Conference On Computer Vision And Pattern Recognition, Computer Vision And Pattern Recognition (CVPR), 2014 IEEE Conference On, Computer Vision And Pattern Recognition (CVPR), 2013 IEEE Conference On 580. IEEE Xplore Digital Library, EBSCOhost(accessed April 22, 2018).

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, (6). 84.

[4] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.

[5] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., & ... Rabinovich, A. (2014). Going Deeper with Convolutions.

[6] Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., & Weinberger, K. Q. (2017). Snapshot Ensembles: Train 1, get M for free.

[7] Kumar, S. K. (2017). On weight initialization in deep neural networks.

[8] Qian, S., Liu, C., Wong, H. S., Liu, H., & Wu, S. (n.d). Adaptive activation functions in convolutional neural networks. Neurocomputing, 272204-212.

[9] Zeiler, M. D., & Fergus, R. (2013). Visualizing and Understanding Convolutional Networks.

[10] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., & ... Fei-Fei, L. (2014). ImageNet Large Scale Visual Recognition Challenge.

[11] A performance analysis of convolutional neural network models in SAR target recognition. (2017). 2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA), SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA), 2017, 1. doi:10.1109/BIGSARDATA.2017.8124917

[12] Hinton, G. E., Osindero, S., & Yee-Whye, T. (2006). A Fast Learning Algorithm for Deep Belief Nets. Neural Computation, 18(7), 1527-1554.

[13] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors.

[14] Yu, D., Seltzer, M. L., Li, J., Huang, J., & Seide, F. (2013). Feature Learning in Deep Neural Networks - Studies on Speech Recognition Tasks.

[15] Gradient-based learning applied to document recognition. (n.d). PROCEEDINGS OF THE IEEE, 86(11), 2278-2324.

[16] What is the best multi-stage architecture for object recognition. (2009). 2009 IEEE 12th International Conference on Computer Vision, Computer Vision, 2009 IEEE 12th International Conference on, 2146. doi:10.1109/ICCV.2009.5459469

[17] Deep convolutional neural networks for LVCSR. (2013). 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, 8614. doi:10.1109/ICASSP.2013.6639347

[18] Learning mid-level features for recognition. (2010). 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2559. doi:10.1109/CVPR.2010.5539963

[19] Convolutional networks and applications in vision. (2010). Proceedings of 2010 IEEE International Symposium on Circuits and Systems, Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, 253. doi:10.1109/ISCAS.2010.5537907

[20] Zeiler, M. D., & Fergus, R. (2013). Stochastic Pooling for Regularization of Deep Convolutional Neural Networks.

[21] Best practices for convolutional neural networks applied to visual document analysis. (2003). Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings., Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on, Document analysis and recognition, 958. doi:10.1109/ICDAR.2003.1227801

[21] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., & ... Chen, T. (2018). Recent advances in convolutional neural networks. Pattern Recognition, 77354-377. doi:10.1016/j.patcog.2017.10.013

[22] Yadan, O., Adams, K., Taigman, Y., & Ranzato, M. (2013). Multi-GPU Training of ConvNets.

[23] Le, Q., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. (2011). On optimization methods for deep learning. MACHINE LEARNING -INTERNATIONAL WORKSHOP THEN CONFERENCE-, 265.

[24] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. International Conference on International Conference on Machine Learning (Vol.38, pp.III-1139). JMLR.org.

[25] Wager, S., Wang, S., & Liang, P. (2013). Dropout Training as Adaptive Regularization.

[26] Deep Residual Learning for Image Recognition. (2016). 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, 770. doi:10.1109/CVPR.2016.90

[27] Fast R-CNN. (2015). 2015 IEEE International Conference on Computer Vision (ICCV), Computer Vision (ICCV), 2015 IEEE International Conference on, Computer Vision, IEEE International Conference on, 1440. doi:10.1109/ICCV.2015.169

[28] Karpathy, A., & Fei-Fei, L. (2017). Deep visual-semantic alignments for generating image descriptions. IEEE Transactions On Pattern Analysis And Machine Intelligence, (4), 664.

[29] Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. (2017). 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, CVPR, 3296. doi:10.1109/CVPR.2017.351

[30] Object Detection Using Convolutional Neural Networks in a Coarse-to-Fine Manner. (2017). IEEE Geoscience and Remote Sensing Letters, Geoscience and Remote Sensing Letters, IEEE, IEEE Geosci. Remote Sensing Lett, (11), 2037. doi:10.1109/LGRS.2017.2749478

[31] Zhongling, H., Zongxu, P., & Bin, L. (2017). Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data. Remote Sensing, 9(9), 1-21. doi:10.3390/rs9090907

[32] Yuan, Z., & BiMing, S. (2017). A Novel Approach for Regularization of Convolutional Neural Network. Journal Of Digital Information Management, 15(1), 30-36.

[33] Spoerer, C. J., McClure, P., & Kriegeskorte, N. (n.d). Recurrent Convolutional Neural Networks: A Better Model of Biological Object Recognition. Frontiers In Psychology, 8

[34] Deep Shrinkage Convolutional Neural Network for Adaptive Noise Reduction. (2018). IEEE Signal Processing Letters, Signal Processing Letters, IEEE, IEEE Signal Process. Lett, (2), 224.

[35] Limonova, E., Sheshkus, A., Ivanova, A., & Nikolaev, D. (2018). Convolutional Neural Network Structure Transformations for Complexity Reduction and Speed Improvement. Pattern Recognition And Image Analysis, (1), 24. doi:10.1134/S105466181801011X

[36] Lin, P., Li, X. L., Chen, Y. M., & He, Y. (2018). A deep convolutional neural network architecture for boosting image discrimination accuracy of rice species. Food And Bioprocess Technology, 11(4), 765-773. doi:10.1007/s11947-017-2050-9

[37] Chongchong, Y., Lan, Z., Xin, W., Jingzhu, W., & Qian, L. (2017). Hyperspectral detection of unsound kernels of wheat based on convolutional neural network. Food Science, China, 38(24), 283-287. doi:10.7506/spkx1002-6630-201724046

[38] Robust abdominal organ segmentation using regional convolutional neural networks. (2017). Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), doi:10.1007/978-3-319-59129-2_4

[39] Dog Breed Identification-Determine the breed of a dog in an image Available at: https://www.kaggle.com/c/dog-breed-identification/data (Accessed: March 6, 2018).

[40] Stanford Dogs Dataset Available at: http://vision.stanford.edu/aditya86/ImageNetDogs/ (Accessed: March 6, 2018).