Fantastic Beasts and Where to Find Them

Zhangjie Lyu
zlyu25@wisc.edu

Chen Xing
cxing6@wisc.edu

Lingfeng Zhu
lzhu88@wisc.edu

Abstract

Pre-classification of wild animals offers scientists and volunteers a better approach to protect wild animals in time. This report is trying to use machine learning techniques to accomplish this task. Based on image augmentation and edge detection with sobel operators, we build a gray scale image data set of 2,000 samples with 5 species from Caltech 101. Afterwards, we execute two experiments, the first one builds the model based on k-NN, SVM and other ensemble algorithms, among them, random forest is the best with micro-AUC-ROC 0.85, macro-AUC-ROC 0.84, mean accuracy 71.4%. While in the second experiment, we build the model based on best three models from first experiment with PCA and redo random search, among them, random forest is still the best with micro-AUC-ROC 0.93, macro-AUC-ROC 0.92, mean accuracy 76.52%.

1. Motivation

It is a well-known fact that the number of wild animal species is decreasing. In order to protect them, a very popular way is to bring them from the wild,build a center where volunteers and animal experts feed them and breed them so that their population can grow faster. After the population grows large enough, animal experts will finally send them back to the wild. However, it is not the end.Experts will need to monitor the behavior of these animals for a rather long period of time to make sure that it is unlikely that they will become endangered again. Moreover, animal experts will also monitor their behavior to improve their studies. Therefore, monitoring animal behavior is an important part in this whole process. To do this,a common way is to set mini cameras and sensors in the wild and analyze the video or photos from the sensors.However, since these sensors work 24 hours a day, 365 days a year, the amount of pictures and video shot by them is huge and it is impossible to analyze these data manually.Our group want to find out a convenient and inexpensive method to deal with this problem.We want to build a model that can analyze the image and separate these images into different classes automatically.

2. Introduction

Our project is about image classification.Our data set comes from Caltech 101. It contains images about 101 kinds of different object and it was collected in September 2003 by Fei-Fei Li, Marco Andreetto and Marc 'Aurelio Ranzato. Due to the limit of computing resource, We choose 5 class(Rhino,Cougar,Elephant,Hedgehog,Dolphin) over the 101 classes to classify. Our ultimate goal is that our model can tell us which of the 5 class it belongs to when we input a new image.

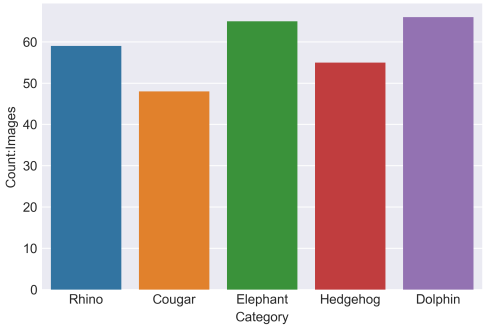


Figure 1: Number of images in each of the 5 class

The plot above shows how the images separated between different classes.The images are evenly separated with 50-70 images in each class.The data set is balanced.In addition,we can see that the number of images in each of the class is rather small compared with the size of input(most of the image has the size 300*300 pixels). This may lead to over fit when we are building our model.We deal with this problem through data augmentation.

The background of the picture are quite different. Since our task is to detect the animal in the image, the background of the image is then the noise of in the image. In order to make our prediction better, we use some of the feature extraction method to erase the noise in the image or to enhance the feature in the image.

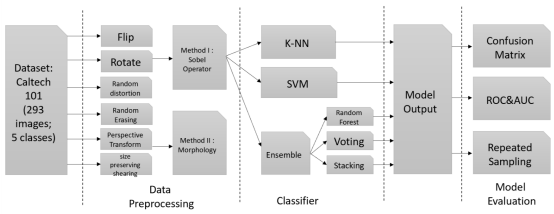


Figure 2: Pipeline of the project

3. Related Work

3.1. Related work on animal image classification

In previous work, different machine learning algorithms have been used to classify animal images into different classes. [4][2][8]. Previous study has shown that basic machine learning tools such as KNN,SVM and random forest all performs well in image classification problem.Therefore,in our project, we choose these three as our base classifiers. We discuss about how these methods can be improved to fit the data set better.

3.2. Related work on Caltech 101 data set

Caltech 101 is a really famous image classification data set. This problem has been the subject of many recent papers .[5][9][6] However, most of methods implemented on this data set are neural network. There are lots of papers discussing how different kind of neural network perform on this data set. There are few paper talking about how basic machine learning and classification tools can be implemented and how these methods can be improved to better fit the data. This is what our project mainly discuss.

4. Proposed Method

We choose k-Nearest Neighbor algorithm (k-NN), Support vector Machine algorithm (SVM) and Random Forest algorithm to classify the images. We first split the data set into training data (70%) and test data (30%).

4.1. k-Nearest Neighbor

Since k-NN is one of the simplest algorithms in machine learning, we first try to use this method. After the data preprocessing, we have already obtained a 128×128 matrix for each image in the data set.

To perform the k-NN algorithm, we transform the matrix into a 16384×1 vector by reshaping it by row. Consider the target function f that assigns the class label of our training example. That is, if we let y denote the class label of a training data x, we have:

$$f(x^{[i]}) = y^{[i]}.$$

Choose a measure of distance, we can find k nearest neighbors of a query sample $x^{[q]}$:

$$x^{[1]}, ..., x^{[k]}.$$

Then, the k-NN algorithm will classify $x^{[q]}$ into class $h(x^{[q]})$, while $h(x^{[q]})$ satisfies:

$$h(x^{[q]}) = mode(\{f(x^{[1]}), ..., f(x^{[k]})\}).$$

There are many distance measures we can choose to find the k nearest neighbors. for example, we can use Euclidean distance:

$$d(x^{[a]}, x^{[b]}) = \sqrt{\sum_{i=1}^{m} \left(x^{[a]} - x^{[b]}\right)^2}.$$

In addition, we can choose the Manhattan distance: (Here m denotes the dimension of the data)

$$d(x^{[a]}, x^{[b]}) = \sum_{i=1}^{m} \left| x^{[a]} - x^{[b]} \right|.$$

Also, we can use Minkowski distance:

$$d(x^{[a]}, x^{[b]}) = \left[\sum_{i=1}^{m} \left(\left| x^{[a]} - x^{[b]} \right| \right)^p \right]^{\frac{1}{p}}.$$

We know that k-NN is very sensitive to the curse of dimension$[X]$, and sometimes it's expensive to compute. We can use data structures like KD-trees and Ball-trees to make it more efficient.

We use random search to perform the hyper-parameter tuning. In this case, that means to find the best hyper-parameter k, the best measure of distance, and the best data structure. And the results indicate the we should use $k = 6$, choose Euclidean distance and consider KD-trees.

4.2. Multiclass Support Vector Machine

We can also use SVM to classify the images. Experimental results show that in image clasification, SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. [WIKI SVM] Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. [WIKI SVM]

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem.

There are many kernel functions to choose. For two vector $x^{[i]}$ and $x^{[j]}$: Polynomial (homogeneous):

$$k(x^{[i]}, x^{[j]}) = (x^{[i]} \cdot x^{[j]})^d$$

Polynomial (inhomogeneous):

$$k(x^{[i]}, x^{[j]}) = (x^{[i]} \cdot x^{[j]} + 1)^d$$

Gaussian radial basis function: (Here $\gamma > 0$)

$$k(x^{[i]}, x^{[j]}) = exp\bigg(- \gamma \Big|\Big|x^{[i]} - x^{[j]}\Big|\Big|^2 \bigg).$$

Hyperbolic tangent: (Here $k > 0$ and $c < 0$)

$$k(x^{[i]}, x^{[j]}) = tanh(kx^{[i]} \cdot x^{[j]} + c)$$

Since our class labels are drawn from a set of several elements, we have to use multiclass SVM. We will reduce the multiclass problem into multiple binary classification problems. We can either build binary classifier between one of the labels and the rest or between every pair of the classes.

We already obtained series of vectors from our images after pre-processing, to perform SVM algorithm, we just need to perform hyper-parameter tuning. The effectiveness of SVM depends on the selection of kernel, the kernel's parameters and soft margin parameter C. The random search algorithm indicates that we should use Polynomial as kernel, and we choose kernel parameter $\gamma = 0.1$, soft margin parameter $C = 20$.

4.3. Random Forest

Random Forest has a good and robust performance in most of the classification problems. We also use random forest as one of our classifiers. In most cases, random forest performs much better than individual decision tree and bagging. Meanwhile, it is much faster than other ensemble method such as stacking and boosting which is really important since we are now dealing with thousands of samples. Therefore, we only choose random forest in our project among all the tree method.

Same as KNN, we use the 128×128 matrix obtained from the data pre-processing.Since random forest is implemented upon a vector instead of a matrix, we transform our matrix into a 16384×1 vector.

The main idea of random forest is to implement lots of decision trees to improve the robustness of the algorithm and then make decisions through majority vote.According to Breiman [1], the upper bound of the generalization error among ensemble method is $\frac{\overline{\rho}(1-s^2)}{s^2}$.Therefore, to make the generalization performance better, random forest has different bootstrap samples in each of the tree and has random feature subset in each node.

The major hyper-parameter in random forest are number of trees, method of splitting in each tree, minimal features in the leaf node(or maximal leaf node). We use random search to tune the hyper-parameter and we estimate the performance each hyper-parameter set through 5-fold cross validation.

4.4. Principle Component Analysis

Principle Component Analysis (PCA) is a common method in dimension reduction.It aims at extracting principle features (or principle component) from redundant features and accelerate the training speed of models with relatively low loss in models' essential information. However, be cautious when implementing PCA, since the principle component after PCA remains no specific physical meanings.

In the problem of image classification, even the gray scale images still possess features (pixels) more than 10,000 in most cases, resulting in implementation of dimension reduction a natural choice in solving these kinds of classification problems.

Basically, the method how PCA finds principle components is similar with what linear regression does. For example, if the raw data set has m observations, each observation has n dimensions($m \gg n$), to reduce the dimension to $k(n > k)$ dimensions with PCA is to find k observations(whose dimension is n) and span a new k dimensional space with these k observations (hence theses k observations are projected to k dimension),thereafter, also projecting the rest observations to k dimensional space. Be noticed that the trick of finding such k "pioneer" observations is to find a combination of k observations who has a least projection error.

More mathematically, the process of PCA is as follows: Suppose the dimension of raw data is n, our object is to reduce the dimension to $k(n > k)$ and the number of features reduces from n to k:

Step 1: Standardized Features

To eliminate the discrimination of features, it is essential to transform all features to a universal scale. To express mathematically:

$$x_j^{[i]} = \frac{x_j^{[i]} - \mu_j}{s_j}$$

where μ_j is the mean of feature $j, j = 1, 2, ..., n$, s_j is the standard deviation of feature j, notice that, each observation is n dimension, which means n features and $i = 1, 2, ..., m$.

Step 2: Calculate co-variance matrix

Calculate co-variance matrix of standardized features by:

$$(\Sigma)_{i,j} = \frac{1}{m}\sum_{k=1}^{m} x_i^{[k]} * x_j^{[k]}$$

where m is the number of total observation,$\Sigma \in R^{n\times n}$ $x_i \in R^{m\times 1}$ is the vector of feature i over all m observations, and by standardization, the mean of feature becomes 0, as a consequence of which, $X - \bar{X} = X$.

Step 3: Get eigenvalues through SVD decomposition

Basically, any matrix Σ, regardless of its size, can be decomposed as

$$\Sigma = U\Lambda V^T$$

where U, V are two orthogonal matrix, which corresponds to shape-preserving rotation (one kind of rigid motion), while Λ is a diagonal matrix of singular values (which is the eigenvalue of $\Sigma\Sigma^T$ if Σ is a normal matrix). Through this process, it can be obtained that:

$$(U, \Lambda, V^T) = SVD(\Sigma)$$

Where $\Sigma \in R^{n\times n}, U, V \in R^{n\times m}, \Lambda \in R^{m\times m}$.

Step 4: Constructing new features from the last step

If it is required to get a reduced feature matrix (design matrix) of k dimension, use first k columns of matrix U directly, from which it can be obtained that

$$U_{reduced} = [u^{[1]}, u^{[2]}, ..., u^{[k]}], k \leq n$$

thereafter, the new eigenvector is:

$$z^{[i]} = U_{reduced}^T x^{[i]}, z^{[i]} \in R^{k\times 1}$$

where $U_{reduced} \in R^{n\times k}, x^{[i]} \in R^{n\times 1}, i = 1, 2, ..., m$, and $z^{[i]}$ is the new k features of observation i.

4.5. t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding[7] (t-SNE), which plays a similar function with PCA, reduces features' dimensions (reduce number of features). Literally, PCA is based on decided mathematical formula, t-SNE is from a probabilistic view. Contrary to PCA, t-SNE is a non-linear dimension reduction method. When comparing with PCA, t-SNE would show a better performance in dealing with non-linear or unknown structure theoretically.

Basically, all dimension reduction methods are trying to accomplish one single thing – mapping high dimensional data to low dimension, clustering data who should be of the same type. From this view, dimension reduction looks more like a pre-classification method applied to data sets when regular ways don't work well. This is also true with t-SNE.

If we illustrate PCA's criteria of measuring similarity as Euclidean metric (projecting raw data point to new coordinator with lower dimension), t-SNE's idea of judging similarity is based on comparing condition probability of two data points. Additionally, both t-SNE and PCA is pretty suitable for data visualizing.

4.6. Feature Detection in Gray Scale Images

In this project, we apply Sobel operator for edge detection, which is our main image features, on our gray scaled images.

Sobel operator is a typical convolution method in detecting edges, where we assume 'edges' to be pixels who has a sharpest change in pixel's intensity (since we are processing gray scale images). Based on this idea, sobel operator is developed to detect location of pixels in a image who has a higher gradient (derivative in 1-d case) than neighboring pixels (or at least higher than a threshold). To be more mathematically, we set two kernels G_x(detect horizontal changes) and G_y(detect vertical changes) and do convolution on image X(which is a $m \times n$ matrix), and calculate the magnitude of gradient at certain pixel by $G = \sqrt{G_x^2 + G_y^2}$, here:

$$G_x := \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y := \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

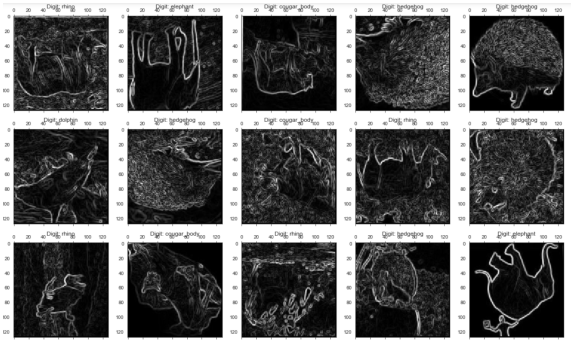


Figure 3: Samples of images operated by sobel operator+image sharpen, extracted from augmented images of 5 species randomly

The figure 3 shows samples of gray scale operated by Sobel operators.

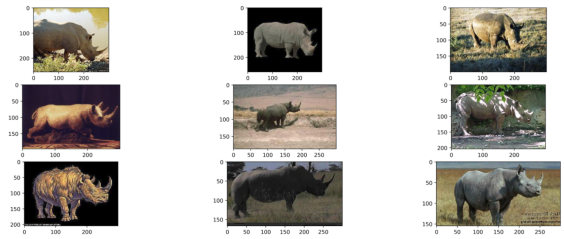


Figure 4: Nine images of class 'rhino' from sliced Caltech 101 data set, all images are RGB but the size are not concordant.

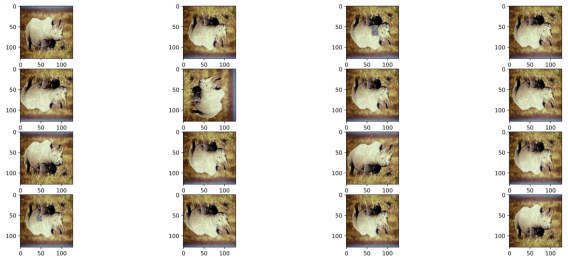


Figure 5: A sample of generating augmented images from one raw image from class 'rhino', all samples are re-sized to 128*128*3 and the probability of occurring one of six transforms mentioned previously are randomly chosen.

5. Experiments

In this section, the process of data processing details will be reported.

5.1. Caltech 101

Caltech 101[3] is pictures of objects belonging to 101 categories. About 40 to 800 images per category. Most categories have about 50 images. The data set was collected in September 2003 by Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato. The size of each image is roughly $300 * 200$ pixels.

Our experimental data set is a slice from Caltech 101, containing 5 species of animals (hedgehog, dolphin, elephant, rhino, cougar). The raw image is of RGB format and the size of them vary, the figure 4 are samples of raw images from 'rhino' class. Moreover, the number of images in each category of our sliced images ranging from 50-70, which is roughly a balanced raw data set.

5.2. Image Augmentation

Since the size of images are not concordant, we re-size all images to $128*128*3$ (which means 128 pixels in height, 128 pixels in width, 3 color channels) to make a universal size of images. But the data set after re-sizing is still too small for our image classification problem, there's only 293 images in all. This is not enough to make a good model, even the features (before feature extracting) of equal-size gray scale images is over 10,000.

Therefore, to augment our raw data set, we implement 6 different types of transforms on our raw data set, which adds more training and testing samples as well as more noise (because of addition of random-erasing). The details of 6 augmentations are as follows:

Transform 1: Flip

'Flip' contains 2 different sub-types, one is left-right flip of image(occurs uniformly, randomly to each image with probability 40%); the other one is top-bottom flip of image(occurs uniformly, randomly to each image with probability 80%).

Transform 2: Rotate

'Rotate transform' randomly rotate a image by 90 degrees(occurs uniformly, randomly to each image with probability 10%).

Transform 3: Elastic distortion

'Elastic distortion' makes distortions to an image while maintaining the images aspect ratio(occurs uniformly, randomly to each image with probability 40%, other properties includes: grid_width =4 , grid_height =4 , magnitude =5).

Transform 4: Shearing (Perspective shearing)

'Shearing' transform tilts an 2-d image along one of its sides. The can be in the x-axis or y-axis direction(occurs uniformly, randomly to each image with probability 30%, other properties includes: max_shear_left=10, max_shear_right=10).

Transform 5: Perspective transform(Perspective skewing)

'Perspective transform' involves transforming the image so that it appears that you are looking at the image from a different angle(occurs uniformly, randomly to each image with probability 30%, other properties includes: magnitude=0.2).

Transform 6: Random erasing

'Random erasing' involves transforming the images so that a fixed-area rectangular is substituted with noise,literally, it adds mosaic(occurs uniformly, randomly to each image with probability 10%, other properties includes: rectangle_area=0.2).

The figure5 is a specific sample generated after applying re-sizing and 6 transforms of random distributed probability.

5.3. Image pre-processing

Image pre-processing includes 3 parts, the first part is to transform all RGB images of size 128*128*3 to gray scale images with size 128*128(only show intensity of a image), the second part is to try sharpening the image to outline the edge, the last part is to implement Sobel operator mentioned

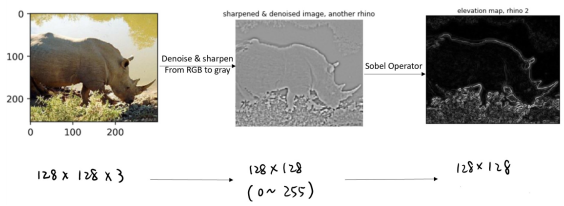


Figure 6: Pipeline of image pre-processing: (from left to right)gray scale image, image sharpen/denoise, image operated by sobel operator.

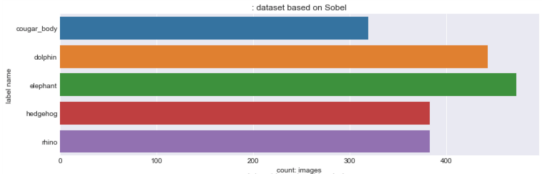


Figure 7: Image data set after re-sizing and augmentation(from probability and transforms set in the previous subsection), including 5 species still, all images are of 128*128*3

in the last section to detect edge in images. Particularly, in transforming RGB images to gray scale images, we apply the formula:

$$gray = 0.299 * R + 0.587 * G + 0.114 * B$$

In the part of image sharpening, we apply gaussian filter with $\sigma = 1$ and set the coefficient as 30(pick 50 if sharpen images after denoising). In summary, the elements in matrix of sharpened gray scale images are calculated by:

$$Sharpen = gray + 30 * (gray - gaussian(gray, \sigma = 1))$$

Also, similarly, step of denoising imgaes can also be applied, which can be added as an addition step before sharpening images:

$$Denoise = gaussian(gray, \sigma = 3)$$

But the step of denoise will blur the edge as well, so it is essential to apply 'sharpen' after denoising. The last step is to apply sobel operator mentioned in the previous section to detect edges in sharpened images.

The whole pipeline of this image pre-processing has been shown in figure6.

5.4. Experimental setup

Each experiment was carried out under identical conditions. For all categories of animals (recall there are 5 species), applying re-size and image augmentation on them, then the augmented images data set can be obtained and the augmented image data set is balanced.

In the experiment, we set the number of augmented image data set to be 2,000, that is, after image re-sizing and augmentation, there are 2,000 RGB images with size 128*128*3 of 5 species, the statistics of the new image data set is shown in figure 7.

The next step is to do feature extraction (edge detection) on the new image data set. Transforming all image data to gray scale images and sharpening images with gray scale intensity, we have 2,000 transformed data set within which images are all 128*128 and gray scaled. The last step in pipeline of image processing is to apply sobel operator on these 2,000 transformed images.

To begin our experiment, we extract training data set (including validation data set) from 2,000 transformed images, the size of training data set is 70%(in the first experiment) and 90%(in the second experiment) of whole transformed data set, and when extracting, 'stratify' variable is set to be the class of species.

After splitting transformed data set into training data set and testing data set. In the first experiment, we use the training data set of size 70% and implement K-NN, SVM, Random forest, Stacking, Adaboost and Voting classifier to fit our training data set. To state it in more details, we first use random search algorithm to find the best parameters based on average score of cross validation on training data set (with cv = 5), and predict the testing data set under best parameter successed from random search in this case. In the second experiment, we apply PCA to training data set(of 90% size) in advance and pick the first 30 principle components. In this case, we implement K-NN, SVM and Random forest on the training data set after PCA and pick best models by random search with average score of cross validation(with cv = 5). Notice that, PCA suppresses information, therefore, to obtain a good model, we extend the size of training data set.All other parameters remain the same during both experiments.

In evaluation of models, the main metrics are confusion matrix and receiver-operating-characteristics (ROC) and the area under ROC curve (AUC-ROC), in the latter case, a model performs as well as random guess has a AUC-ROC of 50%, the two metrics are applied to three best models in first and second experiment separately. In addition, to evaluate the robustness of our training data set(also, the robustness of models), we also apply repeated sampling of training data set and rebuild the models with best parameters in two experiments separately. It is expected that a good model should have a relatively similar performance when trained with difference training data set with same size. Notice that the variance in cross validation can partly reflect the robustness of models too, a good model should have a relative small variance when validated crossly.

5.5. Software

Python 3.7
Jupyter notebook

5.6. Hardware

Macbook Pro (2 GHz Intel Core i5/8GB RAM)
Surface Pro (8th Gen i5/8GB RAM/128GB SSD)
Surface Laptop (7th Gen i5/8GB RAM/256GB SSD)

6. Results and Discussion

As mentioned, we choose accuracy, confusion matrix, ROC and AUC-ROC to evaluate the results of the models, and perform repeated sampling to evaluate the robustness of the models, and we also discussion about some other aspects of the results. The structure of this section is as follows: first we discuss results of first experiment, then we discuss results from second experiment in parallel, we discuss results of two models together in the end of this section.

6.1. Experimental results from first experiment

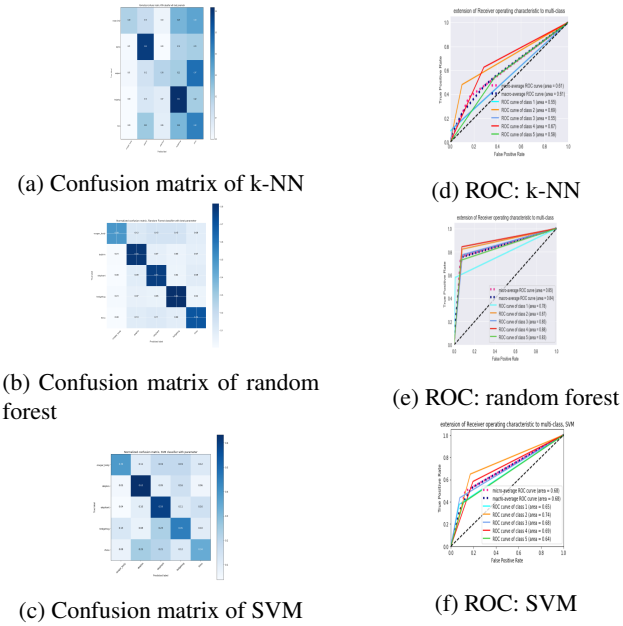
Following the procedures we depicted in previous sections, the results of first experiment can be obtained. First and foremost, we would like to present the process of picking best parameters through random search, involving variance analysis of models:

From table 1, roughly speaking, the model is robust and the model is properly fitted. And among them random forest model has a best performance with a relative low variance. The next step is to check the selected models' performance on testing data set (which is of size 30%).

The figure 8 shows the performance of confusion matrices on testing data set (of size 30%) of three best models selected based on mean cross-validation scores on training data set (of size 70%).

To investigate more into how the three best models perform, we apply ROC and calculate its corresponding area. Basically, when calculating ROC curve, we first binaries all 5 classes sequentially, that is treat one of 5 classes as positive label while treating the rest as negative label, then the multiclass classification problem is transformed into a binary classification problem, the greater the AUC-ROC is, the better the model is. An acceptable model should have all AUC-ROC bigger than 50%, which is the accuracy of a random guess ,regardless the chosen label.

The figure 8 illustrates ROC of three best models, two additional indices(macro and micro auc-roc, the former one is $Macro - ROC = \frac{\sum_{i=1:n_{class}} (\frac{TP_i}{TP_i+FP_i})}{n_class}$, the latter one is $Micro - ROC = \frac{\sum_{i=1:n_{class}} TP_i}{\sum_{i=1:n_{class}} (TP_i+FP_i)}$) are introduced to estimate the overall performance of a model. (From the top left to bottom right)k-NN with macro-auc-roc 0.61, random forest with macro-auc-roc 0.84, SVM with macro-auc-roc 0.68, random forest still has the best performance among them. Therefore, from the first experiment, it can be deduced that random forest has a best performance either in the cross-validation or in the real testing data set.



(a) Confusion matrix of k-NN

(d) ROC: k-NN

(b) Confusion matrix of random forest

(e) ROC: random forest

(c) Confusion matrix of SVM

(f) ROC: SVM

Figure 8: Confusion matrices of three best models selected based on mean cross-validation scores on training data set (of size 70%), (from top left to bottom left)k-NN(a), random forest(b), SVM(c), random forest has a best performance among them. ROC of three best models, two additional indices(macro and micro auc-roc, which are measurement of mean performance) are introduced to estimate the overall performance of a model. (From the top right to bottom right)k-NN(a) with macro-auc-roc 0.61, random forest(b) with macro-auc-roc 0.84, SVM(c) with macro-auc-roc 0.68, random forest still has the best performance among them

To dig more into the random forest model, we do 50 times random sampling (figure 9) to generate different training data set of size 70% repeatedly in order to has a more subjective assessment on the performance of the best random forest model we pick, the data is tested on training data set with cross validation (notice, the model here is properly fitted, no significant overfit nor underfit).

6.2. Experimental results from second experiment

The second experiments follows the same logic as the first experiment, but this time, we decompose the transformed data set into 30 principle components with PCA, that is, we reduce the original data set of size

Model	Parameters	Mean cross-validation accuracy	Standard deviation
k-NN	n_neighbors = 6, p = 2, 'ball tree',leaf_size = 30	44.6%	0.027
SVM	kernel = 'polynomial', gamma=0.1, C = 20	46.6%	0.026
Random forest	criterion='gini',max_feature = 5,min_sample_split=5	50.9%	0.026
Stacking	Not through random search	31%	0.01
Voting classifier	Not through random search	36%	0.03

Table 1: Results of best parameters from first experiment, based on 5-folded cross validation assessed on training data set with size 0.7. (From first column to the last)the model under cross-validation, best parameters under random search, best mean cross-validation score, standard deviation of models with best cross-validation score.Among all models, random forest has a best performance.

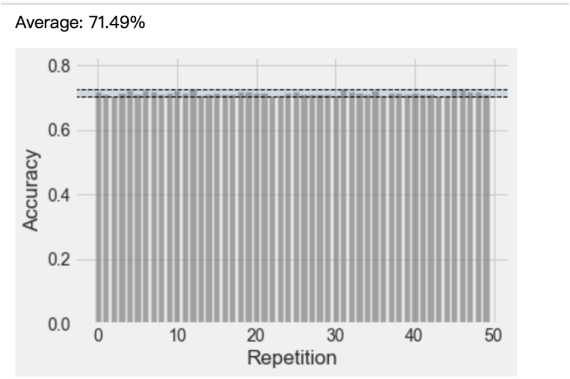
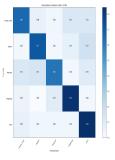
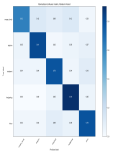


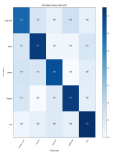
Figure 9: accuracy of training data set during 50 times repeated sampling with same size of training data set, the mean accuracy is 71.4%.



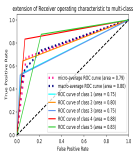
(a) cm of k-NN(PCA)



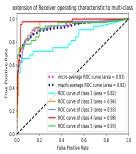
(b) cm of random forest(PCA)



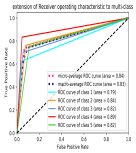
(c) cm of SVM(PCA)



(d) ROC: k-NN(PCA)



(e) ROC: random forest(PCA)



(f) ROC: SVM(PCA)

Figure 10: Confusion matrices of three best models selected based on mean cross-validation scores on training data set (of size 90%), (from top left to bottom left)k-NN(a), random forest(b), SVM(c), random forest has a best performance among them. ROC of three best models, two additional indices(macro and micro auc-roc, which are measurement of mean performance) are introduced to estimate the overall performance of a model. (From the top right to bottom right)k-NN(a) with macro-auc-roc 0.80, random forest(b) with macro-auc-roc 0.92, SVM(c) with macro-auc-roc 0.83, random forest still has the best performance among them

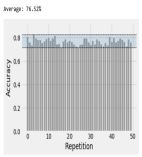
2000*(128*128) to size 2000*30, following a similar methods, with training data set of size 90%, we evaluate k-NN, random forest and SVM from its confusion matrix and ROC.

The figure 10 represents corresponding confusion matrix and ROC of K-NN, random forest and SVM based on PCA and random search, among them random forest has an accuracy of 77% on testing data set,SVM has an accuracy of 69% on testing data set, k-NN has an accuracy of 72% on testing data set. Based on this, random forest is still the best model in the second experiment with mean accuracy 76.52% in repeated sampling prediction.

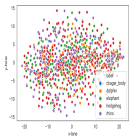
From figure 11, we investigate more in the performance of PCA and robustness in random forest.

6.3. Comparison of two experiments

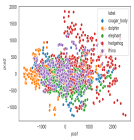
The two experiments both present that random forest plays a best role in dealing with this image classification among algorithms we applied. Moreover, we show that with a higher size of training data set, PCA + regular machine learning algorithms performs quite well within a short time of training and learning models. But from this, it cannot be concluded that PCA is essential in training models despite it does accelerate the process of training models, because PCA losses information completeness and usually requires a larger data set as a remedial measure. However, if considering this image classification problem from a practical



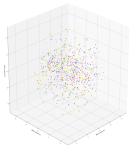
(a) Repeated sampling random forest)



(c) 2-d image, tSNE



(b) 2-d image, PCA



(d) 3-d image, tSNE

Figure 11: A comprehensive image: from top left to bottom right, (a) illustrates the robustness of random forest models obtained in the second experiment,which is repeatedly sampling and predicting, with a mean accuracy 76.52%, (b) illustrates the data point in 2-d with PCA, (c)(d) illustrate the data point in 2-d, 3-d separately with t-SNE technique.

view, PCA does need to be introduced into the process of building models.

6.4. Shortcoming

The model we build is only based on 5 species, the real-world model should be extended to a larger range of species (like major mammals); and our model are not equipped with ability of identifying fake animals from living animals, nor with ability of identifying dead animals from living animals. These two abilities are vital in real-world application.

7. Conclusions

Return to the problem we are dealing with, the situation of this algorithm being used is pre-classification of wild animal images, hence, this data set can have a large training data set support. Through the procedure of experiments, it is observed that random forest + PCA can solve the model of 5 species in a significant shorter time and still guarantee a high performance in prediction (92% macro AUC-ROC ,93% micro AUC-ROC and 76.52% mean accuracy in repeated sampling prediction), therefore, under this data set, the model is a relative good model, but it is still not an accuracy high enough to be applied in real-world to assist scientists and volunteers and it could be better by dealing shortcomings proposed in the last section.On the other hand, Caltech 101[3] is a really famous image classification data set. This problem has been the subject of many recent papers .[5][9][6] However, most of methods implemented on this data set are neural network. There are lots of papers discussing how different kind of neural network perform on this data set. There are few paper talking about how basic machine learning and classification tools can be implemented and how these methods can be improved to better fit the data. This is what our project mainly discuss.

8. Further discussion

We still have a lot of jobs to do in further develop our models, for example, we can endow the model with ability of identifying fake animals from living animals, or with ability of identifying dead animals from living animals. These two abilities are vital in real-world application. In addition, we still need to do more sophisticated analysis in balancing the efficiency of model and information completeness.

9. Acknowledgements

We are glad to acknowledge Prof. Sebastian Raschka, without whose instruction we will not be able to establish and finish this project, let alone entering this interesting world of machine learning. And we'd like to acknowledge Prof. Feifei Li and her team, without whose work we will not have a data set to train our models. Last but not least, we would like to acknowledge Overleaf and Github, they make cooperation more easier than ever.

10. Contributions

Zhangjie lyu set up and maintained a GitHub repository for collaborative work in a version-controlled environment.He implemented data augmentation and data prepocessing part.He also analyzed the result and wrote the experiment part of the report. Chen Xing implemented the first experiment of the project where she used KNN, random forest and SVM to fit the data.She also wrote the introduction and result part of the project. Lingfeng Zhu implemented the second experiment of the project where he used PCA, and tSNE method to reduce the dimension to fit the model better. He also wrote the proposed method part of the report.

References

[1] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.

[3] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

[4] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *Computer Vision,*

2009 IEEE 12th International Conference on, pages 237–244. IEEE, 2009.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European conference on computer vision*, pages 346–361. Springer, 2014.

[6] Q. Li, H. Zhang, J. Guo, B. Bhanu, and L. An. Reference-based scheme combined with k-svd for scene image categorization. *IEEE Signal Processing Letters*, 20(1):67–70, 2013.

[7] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[8] C.-F. Tsai, K. McGarry, and J. Tait. Image classification using hybrid neural networks. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 431–432. ACM, 2003.

[9] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.