Butterfly Classification with Machine Learning Methodologies for an Android Application

Lili Zhu
School of Engineering
University of Guelph
Guelph, Canada
lzhu03@uoguelph.ca

Petros Spachos
School of Engineering
University of Guelph
Guelph, Canada
petros@uoguelph.ca

***Abstract*—In this paper, we evaluated traditional machine learning, deep learning and transfer learning methodologies by training and testing on a butterfly dataset, and determined the optimal model for a developed Android application. The application can detect the category of a butterfly by either capturing a real-time picture of a butterfly or choosing one picture from gallery.**

***Index Terms*—Classification of Butterfly, Deep learning, Transfer Learning, TensorFlow Mobile.**

I. Introduction

The patterns of butterflies not only protect them from predators, but also are essential features for researchers and people who are interested in categorizing butterflies. However, it is arduous to classify butterflies based on biological patterns, such as shapes, wing colors and venations, which are complicated to distinguish [1]. Traditional butterfly identification techniques, such as the use of taxonomic keys [2], depend on manually identifying and classifying by highly trained individuals. Currently, researchers rely mostly on rather time-consuming processes that are inefficient in dealing with the distribution and diversity of butterflies. Therefore, to tackle the the so-called taxonomy crisis [3], there is a need for modern and automated technologies in dealing with species identification.

Traditional machine learning algorithms, such as K-Nearest Neighbor (KNN) and Support Vector Machine (SVM), perform classification well on small datasets. However, for more complicated datasets, deep learning outperforms traditional machine learning. Recently, transfer learning is becoming prevalent because pre-trained models can save computing resources which avoids developing deep networks.

Today both mobile phone users and the artificial intelligence industry expect to have systems implemented in mobile applications. Currently, existing deep learning models can be stored in servers and used by mobile devices which allows data transmittance. However, the process has raised concerns such as the latency in transmittance, which risks the privacy and security of data. Now TensorFlow Mobile and TensorFlow Lite can solve these problems.

This project aims to build several models and develop an Android application as a butterfly detector. After model selection, the most effective was implemented in an Android application to build a system to detect butterfly categories.

II. Related Works

SVM is one of the effective methods in traditional machine learning to address various problems such as classification and regression. Principal Component Analysis (PCA) combined with SVM is also prevailing as it enhances the training efficiency for the model. In [4], they adopted PCA and SVM on two face datasets namely the ATT and IFD database and reached an accuracy of 90.24% and 66.8% on each dataset. In [5], they used this method to detect the event that people fall on the floor and had an accuracy of 82.7%. In [6], they applied PCA and multi-kernel SVM to compare healthy controls and Alzheimer Disease patients with their brain region of interest, and achieved an accuracy of 84%.

As tasks get more complicated, numerous large datasets cannot be processed by traditional machine learning. As a result, the role of artificial neural networks is revealed. In [7], they developed a neural network structure for classifying a dataset including 740 species and 11,198 samples. They had a 91.65% accuracy with fish, 92.87% and 93.25% for plants and butterflies.

In the field of butterfly detection, a content-based image retrieval (CBIR) to detect butterfly species is proposed in [8]. The total accuracy is 32% for the first match and 53% for the first 3 match. In [9], they applied Extreme Learning Machine (ELM) to classify butterflies and compared the results with SVM method. The ELM method has the accuracy of 91.37%, which is slightly higher than SVM. In [10], they built a single neural network to classify butterfly according to their shapes of wings, and the accuracy is 80.3%. In [11], they aim to classify all butterfly species in China, which adopted three methods, ZF, VGG_CNN and VGG16, to compare the performance. The results showed that ZF has an accuracy of 59.8%, while VGG_CNN and VGG16 had 64.5% and 72.8%, respectively.

As for transfer learning, in [12], they utilized the DeepX toolkit (DXTK), which contains several pre-trained low-resource deep learning models for integration in mobile devices. In [13], they used a multi-task, part-based DCNN structure for feature detection and then deployed the structure on a mobile device. In [14], they built real-time CNN networks

based on transfer learning to detect the regions with Post-it®, and transformed these models to be used on smart phones. A Faster RCNN ResNet50 architecture achieved the highest mAP (Mean Average Precision) of 99.33% but the inference time is 20018 ms.

III. Methodologies

The comparison between traditional machine learning, deep learning and transfer learning methodologies on butterfly detection has not been the focus of any research so far. Therefore, we hope to improve the accuracy from previous studies and create a practical application for butterfly detection.

A. Traditional Machine Learning

SVM is one of the most important classical machine learning algorithms. A linear classifier with the largest interval in the feature space is the definition of the SVM algorithm. Its objective is to determine the maximum interval. Then, the problem turns into convex quadratic programming and the maximum interval is the solution to this problem. However, our task is to complete a ten-class classification, which cannot be achieved by linear classifier. We applied nonlinear SVM with Radial Basis Function kernel(RBF kernal) aka. Gaussian kernel. SVM can address nonlinear classification problems by mapping the inputs to a high dimensional space kernels.

Intergrating PCA with an SVM is widely adopted in pattern recognition as not only can it avoid the SVM kernal Gram to become overlarge but also improves computational efficiency for classification [15]. If the dataset is n-dimensional and there are m samples$(x_1, x_2, ..., x_m)$, PCA can reduce the dimensions of these m samples from n-dimensional to $n^{'}$-dimensional, and the m $n^{'}$-dimensional samples can represent the original dataset as much as possible. Therefore, we selected PCA/SVM method as one of the approaches to address this butterfly classification problem in this paper.

B. Deep Learning

The most important advancement of deep learning over traditional machine learning is that its performance improves with the increasing of the amount of data. A 4 convolutional layers model (4-Conv CNN) was built from scratch for this paper. This 4-Conv CNN has 4 Conv2D layers, 2 MaxPooling layer and 5 Dropout layers, and a fully connected layer is following. A Flatten layer is used to flatten the out-put to 1D shape because the classifier will finally output a 1D prediction (category). Two Dense layers have the fully connected function and another Dropout layer between them can discard 30% of the outputs. Because this task is a 10 categories classification, a final layer is combined with a Softmax activation in order to transform the outputs to 10 possibilities.

C. Transfer Learning

Transfer learning is to migrate the learned parameters in a model to a new model in order to help the new model training. Recently, the advanced VGG19 [16] is more widely used. One improvement of the VGG19 over other models is the use of several consecutive 3×3 convolution kernels instead of the larger convolution kernels. For a given receptive field (the local size of the input image related to the output), the use of stacked small convolution kernels is superior to the use of large convolution kernels, because multiple nonlinear layers can increase network depth to ensure more complex learning modes, and the cost is relatively small (less parameters). As a result, we adopted VGG19 as the base model in this paper. The convolutional base of VGG19 structure was frozen and we added our own fully connected layer on top of the base.

IV. Dataset and Application

A. Dataset

The dataset for this project is called Leeds Butterfly Dataset [17], and there are 832 images in total. This data set has ten species of butterflies with both images and textual descriptions.

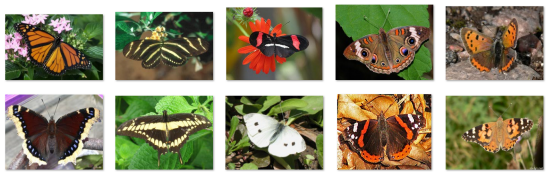


Fig. 1. Ten species of butterflies.

The textual descriptions of every specie were collected from the eNature online nature guide. The descriptions include the name, the normal size and a brief introduction of each specie.

B. Android Application

After the model selection, we built an Android application (app) which utilized the best model to detect the category of butterfly. We built the app in Android Studio. The Sdkversion of the environment is 27 and the mobile phone is API19.

The traditional way to deploy deep learning models is to save the model on servers. When client-side needs a prediction from the model, it should make call to server-side to get result. After data are processed on server-side, the result will be sent to client-side. However, server may work slowly and there is latency during data transmission. Additionally, users prefer not to send private data to servers because of cyber security. As a result, the demand for on-device intelligence capabilities is increasing. As a complement to TensorFlow ecosystem, TensorFlow Mobile and TensorFlow Lite offers researchers and developers the opportunity to run deep learning models on mobile devices locally with low latency, efficient runtimes, and accurate inference. Especially with the computational ability of GPU in nowadays' mobile devices, more complex deep learning models can take advantage of TensorFlow Lite, which allows massively parallelizable workloads and previously not available real-time applications to run fast on mobile devices [18]. Therefore, we chose to implement the model with the cutting-edge technique in this paper.

V. Experimental Results and Discussions

Models need further parameters optimization to achieve the best accuracy after being built. In this paper, all experiments were conducted on device with NVIDIA GeForce GTX 1080 Ti GPU and Intel(R) Core (TM) i7-8700K CPU.

A. Optimization

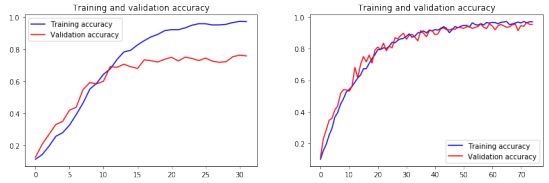
1) Data augmentation: The quality and quantity of datasets is always a concern to researchers and developers as this will affect the accuracy of algorithms. The common problems are overfitting and underfitting. The common way to slow down the problems is to increase the data quantity in a training dataset. In machine learning, we sometimes cannot increase the amount of training data because it is costly to find labeled data. However, the dataset can be expanded by rotating, cropping, shifting, transforming the images, and so on. In this way, we can generally improve the accuracy of the model.

Data augmentation has no obvious enhancement with SVM as the size and ten classes of the original dataset has already challenged the model. Table I shows the accuracy of the SVM/PCA method and it implies that the best accuracy 52.8% occurred when the top 20 components were extracted by PCA.

TABLE I
THE ACCURACY OF SVM PLUS PCA METHOD

Number of components	Acc(%)
10	46.8
15	49.2
20	52.8
25	43.2
30	44.8
35	44.4

When it comes to 4-Conv CNN model, we notice overfitting after training. With certain epochs of training, the validation accuracy started to become stable while the training accuracy still increases. The validation accuracy stopped to increase after 15 epochs of training and the testing accuracy ended up with 85.7% on 4-Conv CNN model. After data augmentation, it is clear that the 4-Conv CNN model overcame the early over-fitting problem. In transfer learning, the pre-trained VGG19 model also showed the improvement after data augmentation.



(a) before data augmentation (b) after data augmentation

Fig. 2. Before (a) and after (b) applying data augmentation on 4-Conv CNN model.

As performances of both 4-Conv CNN model and VGG19 model improved, we have determined that data augmentation indeed prevents over-fitting problem.

TABLE II
THE COMPARISON BETWEEN TRAINING BEFORE DATA AUGMENTATION AND AFTER DATA AUGMENTATION

Batchsize=128	Before	After
Model	Acc(%)	Acc(%)
4-Conv CNN	85.7	95.50
VGG19	94.0	95.97

2) Batch Size: When comparing the results of training with different batch sizes, it is noticeable that the final accuracy with 64 is higher than that with 128 on both groups.

TABLE III
THE COMPARISON BETWEEN DIFFERENT BATCH SIZE

	Batch size=128		Batch size=64	
Model	Epochs	Acc(%)	Epochs	Acc(%)
4-Conv CNN	82	95.50	90	96.63
VGG19	44	95.97	40	96.99

The reason why small batch sizes perform better is that small batch size brings a more stochastic factor to the learning so that the learning has more chances to leave the local minimum and keeps looking for the accurate gradient direction. Although training with large batch size needs less time to finish a training epoch, more epochs are needed to reach the same precision as the small batch size, or even could not reach the same precision. Small batch size will cause the loss function to fluctuate severely if the network is deeper and wider. In order to make the difference more obvious, we trained 4-Conv CNN with batch size of 16 and 256, and we can observe from Fig. 3 that when batch size is 256, it took more epochs to arrive at the similar accuracy as the other one.

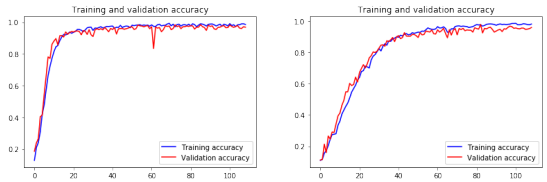


Fig. 3. Training 4-Conv CNN with batch size is 16 (left) and 256 (right).

3) Optimizer: With conducting experiments on 4-Conv CNN model, we note that the SGD needs more epochs to converge and the final accuracy is lower than the others.

TABLE IV
DIFFERENT OPTIMIZERS ON 4-CONV CNN AND VGG19 MODEL

Batch Size	4-Conv CNN		Pre-trained VGG19	
64	Epochs	Acc(%)	Epochs	Acc(%)
Adam	87	96.63	45	96.99
SGD	1262	91.10	522	96.51
Adam+SGD	127	97.72	146	98.53

Figure 4 illustrates that the fluctuations during training with SGD process are way more dramatic than training with Adam. The reason is that the mechanism kept adjusting itself to search

for the correct gradient direction. However, the training has not been converged even after 1200 epochs' training.

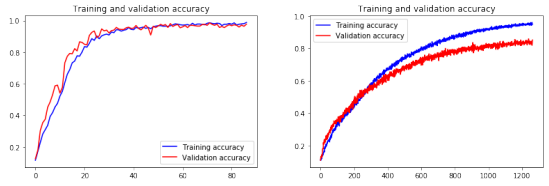


Fig. 4. Training 4-Conv CNN with Adam (left) and SGD (right).

Because of the limited time and machine resources, we did not test SGD on other models and we adopted Adam plus SGD as our default optimization method. This hybrid of Adam and SGD method means that we used Adam as the optimizer to train the model first because of its self-adaptive ability and fast speed, but we applied SGD with a relatively low learning rate (0.00001) to train the model at the late training phase in order to avoid missing the optimal solution.

4) Batch Normalization (BN): VGG model does not have BN layers because BN did not exist before, so we did not add BN to this pre-trained model. Also, it would change the pre-trained weights by subtracting the mean and dividing by the standard deviation for the activation layers [19]. Thus, we did experiments only on the 4-Conv CNN model. Although the testing accuracy after adding BN layers is slightly higher, it took fewer epochs to converge than training without BN since BN layer enables faster training with higher learning rates.

TABLE V
THE COMPARISON BETWEEN BEFORE AND AFTER ADDING BN LAYER

Batch size=64	Before BN		After BN	
Model	Acc(%)	Epochs	Acc(%)	Epochs
4-Conv CNN	97.72	114	98.44	91

B. Comparison between different methodologies

After several attempts to optimize all models, we summarized the best performance of each model. As it is shown in Table VI, deep learning outperforms traditional machine learning. Although the accuracy of the 4-Conv CNN model and the re-trained VGG19 model are very close, according to Occam's Razor, we chose 4-Conv CNN model to embed in Android App as its architecture is simpler than VGG19.

TABLE VI
THE COMPARISON AMONG MODELS

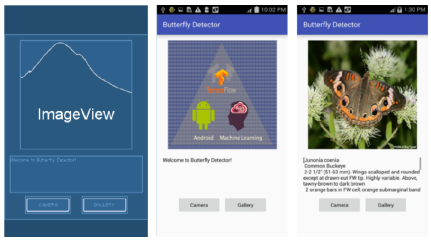
Model		Acc(%)
Traditional machine learning	SVM	52.8
Deep learning	4-Conv CNN	98.44
Transfer learning	VGG19	98.53

Because the structure of SVM is shallow and simple, the only way to enhance the performance is to integrate other algorithms into the existing models. This is why the main task in traditional machine learning area is to search for suitable algorithms, while the chief problem in deep learning is to adjust suitable parameters as the layers in deep learning are relatively easy to understand, but the process to combine different layers together and to adjust parameters is complicated and time-consuming. When it comes to large data sets and complex scenarios, deep learning methods dominates.

C. Android Application

1) Structure: In this Android project, we have three classes which are MainActivity, TensorFlowImageClassifier and Classifier. The deep learning model and label file were uploaded to the assets file. An image can be passed to the model loaded in TensorFlowImageClassifier, in where the label file is also loaded. After the model successfully accepts and predicts the data, the index of the label and the confidence of the result released from the output interface of the model are passed to the class Classifier. Then the predicted label and the description of the image will be returned to the textview defined in MainActivity, and the confidence of the predicted label will be shown in the textview at the same time.

2) Layout and Output: The layout includes an ImageView, an TextView and two buttons. An image can either be captured by camera or selected from gallery and it will be shown on the ImageView. Images from both camera and gallery will be processed and sent to the classifier in the same way. Then the returned label and description are on the TextView.



(a) layout (b) output

Fig. 5. The layout and output of the Android application .

VI. CONCLUSION

This paper proposed an Android application for detecting butterfly categories. We built and optimized three models and the best one is deployed to an Android mobile device. The results show that different models are sensitive to parameters. For SVM, the testing accuracy would not improve after data augmentation as the model cannot handle complex scenarios. But data augmentation is significant to 4-Conv CNN, while transfer learning is not sensitive to the size of dataset because the domain of said dataset is similar to their original. Therefore, choosing tuning different parameters requires patience and experience to serve the objective of the project.

At last we implemented the optimal model in Android application. The result shows that this app distinguishes butterflies with ease. Compared to save models on servers, this app's superiority is that no communication between server

and client, making this application function without network connectivity. This advantage is crucial for butterfly researchers working in the wild. From this perspective, we are convinced that implementation of deep learning on limited resources platforms, e.g., mobile, is beneficial to users and the industry.

REFERENCES

[1] M. Mayo and A. T. Watson, "Automatic species identification of live moths," *Knowledge-Based Systems*, vol. 20, no. 2, pp. 195 – 202, 2007, aI 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705106002012

[2] "Taxonomic keys," https://collectionseducation.org/identify-specimen/taxonomic-keys/, accessed: 2019-8-24.

[3] B. DAYRAT, "Towards integrative taxonomy," *Biological Journal of the Linnean Society*, vol. 85, no. 3, pp. 407–417, 06 2005. [Online]. Available: https://doi.org/10.1111/j.1095-8312.2005.00503.x

[4] A. Bouzalmat, J. Kharroubi, and A. Zarghili, *Comparative Study of PCA, ICA, LDA using SVM Classifier*, 2014, vol. 6(1).

[5] *PCA-SVM Algorithm for Classification of Skeletal Data-Based Eigen Postures*, 2016, vol. 6(5).

[6] S. Alam, M. Kang, J.-Y. Pyun, and G. Kwon, "Performance of classification based on pca, linear svm, and multi-kernel svm," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2016, pp. 987–989.

[7] A. Hernández-Serna and L. F. Jiménez-Segura, "Automatic identification of species with neural networks," in *PeerJ*, 2014.

[8] J. Wang, L. Ji, A. Liang, and D. Yuan, "The identification of butterfly families using content-based image retrieval," *Biosystems Engineering*, vol. 111, no. 1, pp. 24 – 32, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1537511011001784

[9] S. Iamsa-ata, P. Horataa, K. Sunata, and N. Thipayanga, "Improving butterfly family classification using past separating features extraction in extreme learning machine," *Proceedings of the 2nd International Conference on Intelligent Systems and Image Processing 2014*, 2014.

[10] S.-H. Kang, J.-H. Cho, and S.-H. Lee, "Identification of butterfly based on their shapes when viewed from different angles using an artificial neural network," *Journal of Asia-Pacific Entomology*, vol. 17, no. 2, pp. 143 – 149, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1226861513001210

[11] J. Xie, Q. Hou, Y. Shi, L. Peng, L. Jing, F. Zhuang, J. Zhang, X. Tang, and S. Xu, "The automatic identification of butterfly species," *CoRR*, vol. abs/1803.06626, 2018. [Online]. Available: http://arxiv.org/abs/1803.06626

[12] N. Lane, S. Bhattacharya, A. Mathur, C. Forlivesi, and F. Kawsar, "Dxtk: Enabling resource-efficient deep learning on mobile and embedded devices with the deepx toolkit," in *Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services*, ser. MobiCASE'16. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 98–107. [Online]. Available: https://doi.org/10.4108/eai.30-11-2016.2267463

[13] P. Samangouei and R. Chellappa, "Convolutional neural networks for attribute-based active authentication on mobile devices," in *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, Sept 2016, pp. 1–8.

[14] O. Alsing, "Mobile object detection using tensorflow lite and transfer learning," Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

[15]

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[17] J. Wang, K. Markert, and M. Everingham, "Learning models for object recognition from natural language descriptions," in *Proceedings of the British Machine Vision Conference*, 2009.

[18] "Tensorflow lite gpu delegate," https://www.tensorflow.org/lite/performance/gpu, accessed: 2019-6-30.

[19] "Batch normalization in neural networks," https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c, accessed: 2018-11-21.