Bats Classification and Detection through Transfer Learning

Supervisor: Prof. Richard Sinnott

Name: Yifu Tang

Student ID: 951685

Summer semester, 2019

Abstract

The severity of the coronavirus from Wuhan China has deeply affected many people around the world. The origin of the outbreak of the pneumonia is indicated to be related to the bats. Better recognition of bats can help people better study on it and thus enables people to know more about the potential harm it may bring. It is meaningful to detect and classify the bats we see. The fast development of deep learning has proved its efficiency in dealing with large amount of images and can be useful help us recognize the bats. The aim of this paper is to test and compare the effectiveness using popular convolutional neural network models including VGGNet, Inception and ResNet. The feasibility of adapting object detection is also performed using one of the most popular algorithm YoloV3. The accuracy reached 99.10% on the training set and 90.11% on the test set after fine-tuning the VGG16 model. The detection also provide good performance. An executable program is built to illustrate the work.

Keywords— Bats Detection&Classification, Transfer Learning, CNN, YoloV3

Contents

1 Introduction

Bats are the only mammals in nature that have wings and can have a stable flight. There are around one thousand kinds of bats in the world. Different from some pets or poultry, bats do not appear very often in people's daily life, which makes people not so familiar with it and can hardly figure out its breed. Bats brings benefit to human as well as some threats. Bat dung is mined from caves to fertilize as guano and some bat can protect crops by eating insect pests. The complex living environment determines that this species can bring unusual influences to human. Being the natural reservoirs of many pathogens, bats are considered to be the origin of rabies, SARS and coronavirus[1].

Deep learning can be useful in helping recognize bats. Computer vision problem has become an independent and important problem in deep learning. Image recognition is one of the main topics in computer vision. The application of image recognition has influenced so many areas in people's life like automated self-driven cars and face detection. Mature techniques enable us to extend the application to solve more similar problems. This project is going to adopt some cutting-edge methods in computer vision, to be exact, convolutional neural network models, to help to do bats detection and classification.

(a) Indiana Bat (b) Little Brown Bat

Figure 1: Some Bats are hard to distinguish

This paper chooses six major kinds of bats to study, which are Egyptian Fruit Bat, Giant Golden-Crowned Flying-Fox Bat, Indiana Bat, Kitti's Hog-Nosed Bat, Little Brown Bat, and Vampire Bat. A complete process from collecting, labeling and processing data to adapting transfer learning with various models is performed. This paper also tries to find the model that best fits the data by doing a detailed comparison and making fine-tuned adjustment of the parameter. In order to make this work visible, the predicting result is shown as an application that is user-friendly.

2 Related Work

2.1 Deep learning

Deep learning is part of the machine learning family. Different from traditional machine learning methods, deep learning especially refers to the methods based on the neural network[2]. The convolutional neural network makes good use of a large number of layers of the network that effectively extracted features from the data. It has been proved that deep learning approaches perform well in dealing with very large data sets. Deep learning has dominated recent years' contests in the field of image recognition. The most contest held by ImageNet, Large Scale Visual Recognition Challenge (ILSVRC) introduces inspiring models every year[3].

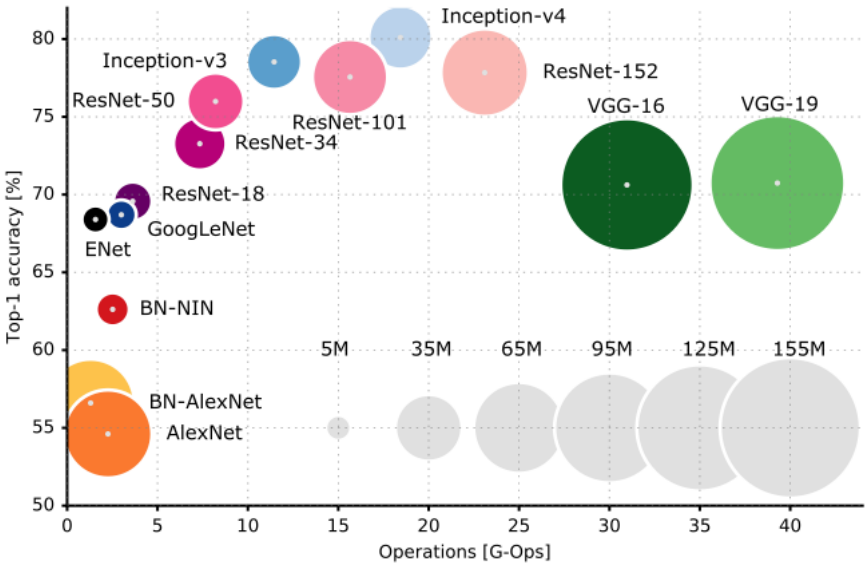

Figure 2: Deep Neural Networks

Training a neural network with many layers can be exhausting because the complexity if the structure determines that the number of parameters can be huge. And the input size should be large enough for these parameters to work. The contest of ILSVRC has more than 14 million images that are hand-annotated to train. Huge computing resources are required to load the image and train the model. It is infeasible for common users to start from the beginning. Normally used gradient descent technique can take so long time to approach to the optimal point and challenges always appear in this process like stuck into the local minima or plateaus[4]. Using the pre-trained model and do some minor adjustments based on it can greatly save time as well as make good use of the existed intelligence.

2.2 Transfer learning

Transfer learning enables knowledge learned from one problem to be applied to some other similar problems easily. It is a popular approach to deep learning. Pre-trained models are elaborately tuned and made good use of abundant computing resources on large enough datasets. It is cost-efficient to retain the major structure of the neural network as well as the weights that have been learned[5].

The advantage of transfer learning is that it best suits a project that has inefficient data. Existing CNN models like ResNet are trained by more than 1 million images and can provide lots of low-level and mid-level features of your dataset if you fit them into it[6][7]. Also, transfer learning uses more complex models that have already been proved to be effective and may improve the performance of the new problem. The process of transfer learning is relatively easier than building a network from the start. It requires fewer resources to train the model by freezing the majority of layers.

This project uses some state-of-art networks that performed will in past years' ImageNet challenge to perform transfer learning.

VGGNet

VGG is the abbreviation of the Visual Geometry Group. The model is introduced in 2014 by Simonyan and Zisserman[8]. VGG network is famous for its simplicity by using only 3*3 convolutional layers. This model wins second place in the competition of ILSVRC held by ImageNet. It uses max pooling to reduce the volume size. Two versions of models are VGG16 and VGG19 where the number stands for the number of weight layers in the network. There are two main disadvantages to training these networks. One is that it takes so long time to train and another is that the model's weights are quite large(138 million parameters for VGG16)

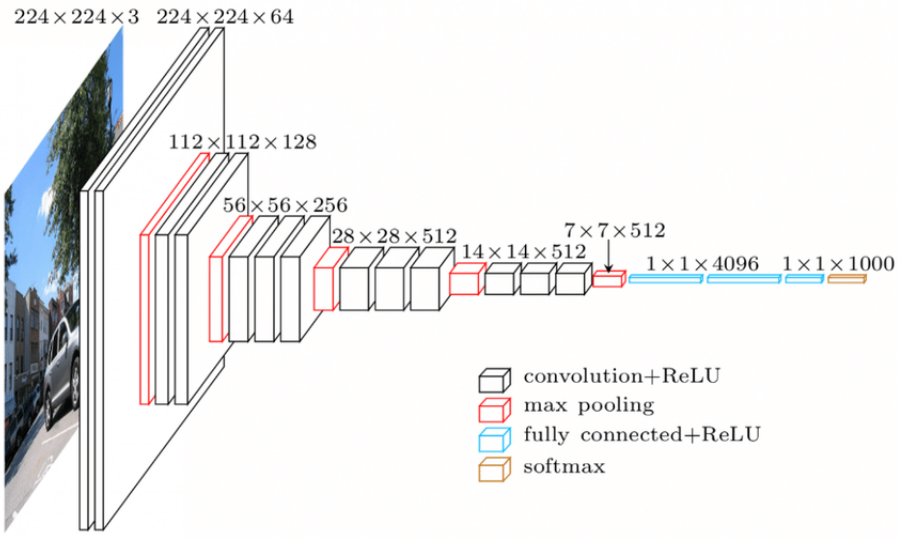

Figure 3: VGG16 Architecture

ResNet

Residual Neural Network is different from other traditional sequential network architectures. The evolution of CNN models before that was nothing but the stack of layers. Accuracy can get saturated with these methods. ResNet uses skip connections to build the deep network. Skip connection is also known as gated units or gated recurrent units. This structure enables the network to have lower complexity than the VGG network even if it has 152 layers.

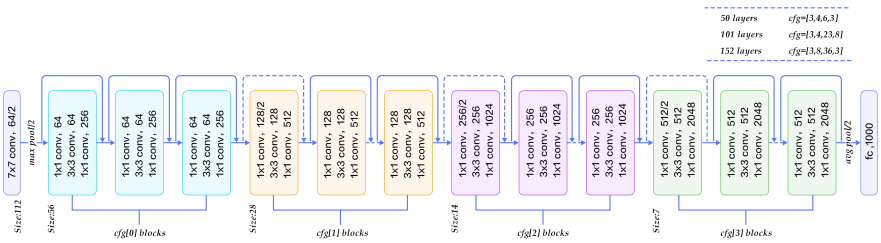

Figure 4: ResNet Architecture

Inception

The Inception module came from GoogleNet[9], who wins the ILSVRC 2014. The subsequent adjustment is simply named Inception Vx with the version number to be x. This model used the Network-in-Network method by introducing the ”Inception module”. The module extract features using 1*1, 3*3 and 5*5 convolutions in the same module and stacks the result before going to the next layer. The novelty is to use dense blocks hence the name of Inception. The model integrated batch normalization and RMSprop in the following

version[10].

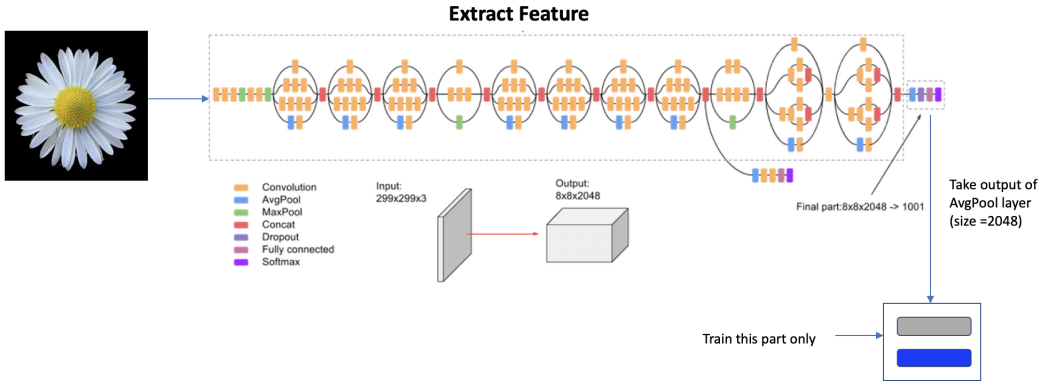

Figure 5: Inception Architecture

YoloV3

YOLO means "You Only Look Once". It is an object detection algorithm that is considered to be a very effective model in speed as well as accuracy. YOLOV3 is the third version of the algorithm. [11]. This third version uses a variant of the Darknet, which has 53 layers trained in the ImageNet. 53 more fully connected layers are added, making the architecture to be 106 layers. The predictions are made across three scales, where the last layer of it predicts the bounding box offsets and category. More meaningful semantic and finer-grained information are extracted by this method. It also employs k-means cluster for better performance finding the bounding box. YoloV3 employs cross-entropy loss function rather than square loss in the previous version. YoloV3 is tested to have better performance than SSD especially in terms of the speed and can better detect the small objects.

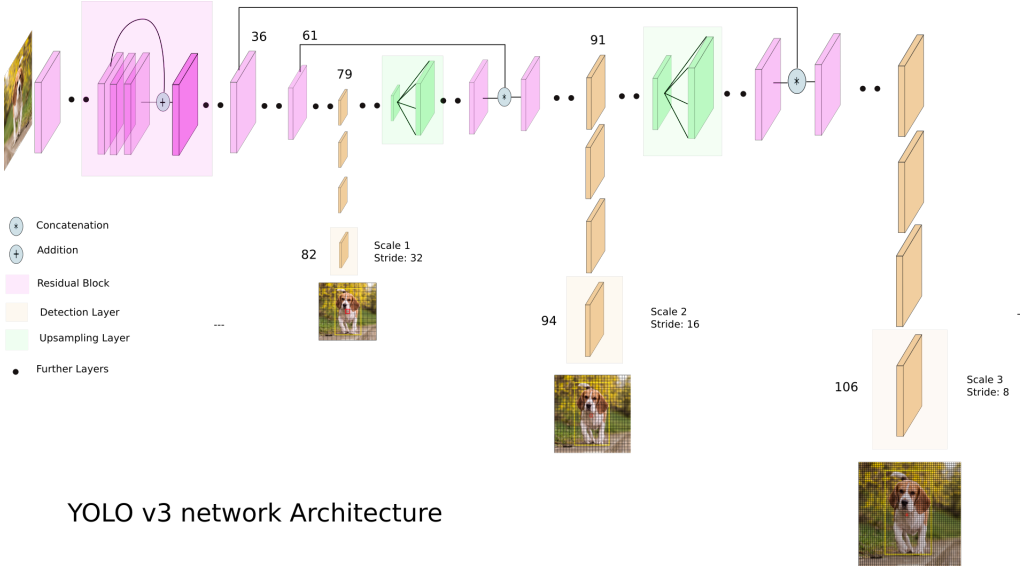

Figure 6: YoloV3 Architecture

3 Dataset

The Data is crawled from google image by an extension application Image Assistant. The crawler simply crawls the image on the current web page in its original size. The images are carefully searched and selected to avoid duplication between different classes. After crawling the image, it is necessary to manually check the quality of the images. For example, the meaning of the ”Flying Fox” is not only one kind of bats but also a sport playing on a zip line.

Figure 7: Ziplining

In order to perform the object detection, the images are labeled carefully by the labeling tool LabelImg. This process generates one .xml format file for each of the images and records the information of each object in the image and its category.

The size of the data is of 887 images in total. The data size for each category is shown as below:

Bats Breeds	Number of Images
Egyptian Fruit Bat	129
Giant Golden-Crowned Flying-Fox Bat	208
Indiana Bat	76
Kitti’s Hog-Nosed Bat	90
Little Brown Bat	233
Vampire Bat	151

Table 1: Unbalanced Data Size

It can be seen that the category of Giant Golden-Crowned Flying-Fox Bat and Little Brown Bat has way more samples than any other category. The unbalanced samples can hurt the

performance of both classification and object detection. Thus the samples are manually pruned. After balancing the datasets, there are 688 images[12]:

Bats Breeds	Number of Images
Egyptian Fruit Bat	129
Giant Golden-Crowned Flying-Fox Bat	105
Indiana Bat	76
Kitti's Hog-Nosed Bat	90
Little Brown Bat	138
Vampire Bat	151

Table 2: Balanced Data Size

4 Training

4.1 Data augmentation

For the computer vision problem solved by CNN, a large sample size is always desired. Generally, more data can result in better performance. While many problems do not have sufficient data, data augmentation is a necessary technique to adapt. Data augmentation refers to "create" more pictures based on the existed images[13]. There are various ways to generate samples such as flipping, cropping, rotating, color shifting and so on[14].

In this project, offline data augmentation is adapted. Different from online augmentation, where data are enriched before feeding into the neural network, offline data augmentation starts before training the model. Offline data augmentation is suitable for small datasets like that. The generated images are stored in the memory so it is not suitable for a larger dataset because of the pressure of hard disk resources and if the data requires to be transferred between servers, it is efficient to do so.

This project compares the performance between different classification models, so offline data augmentation is used to prepare the dataset in advance

4.2 Data split

The dataset is split into the training set and test set. In the training set, both image features and the corresponding label are included. The test set is used to evaluate the accuracy of the model of the last epoch by comparing the prediction and the actual label. The library sklearn is used to generate training data and test data. Since the dataset is small, 90 percent of the data are used for training and the rest are for testing.

4.3 Configuration

4.3.1 Batch size

Batch size is an important parameter that can influence the training speed and model performance greatly. The value of batch size refers to the number of samples that will be propagated through the network. According to different configurations of batch size, there are three flavors of the learning algorithm[15],

1. Batch Gradient Descent: Batch size equals to the number of examples in the training dataset.
2. Stochastic Gradient Descent: Batch size equals to one.
3. Minibatch Gradient Descent: Batch size is set between one and the total number of examples in the training dataset.

Smaller batch size has many advantages. Smaller batch makes it easier to fit the data into the memory, which is crucial when unable to fit the whole data. Mini-batch also trains faster because of the weight update after each propagation. Another effect is that the smaller batch size is noisy, which can provide some regularizing effect that can lower generalization error. But smaller batch can also make the gradient fluctuates and hurt the estimation[16].

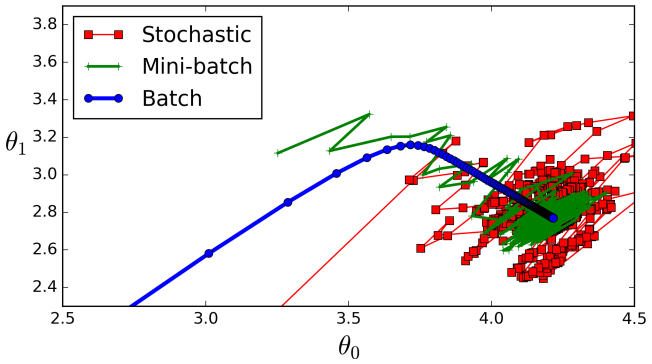

Figure 8: Batch Size Flavors

For classification models, we set batch size equals to 32, which is roughly tested to be a good balance between accuracy and speed. The following test also tunes this parameter with care.

4.3.2 Epoch

The number of the epoch is another hyperparameter that affects the training process. It defines how many times the learning algorithm will work through the entire dataset. During one epoch, there will be several batches according to the batch size flavor. Traditionally the

value of epoch is set to be large to let the complex neural network fully trained and make the loss small enough. But too large epoch number can also result in overfitting problem, that is, the model performs very well in the training set but loses the ability of generalization so fit bad on the test set.

For comparing models, 200 epochs are configured to save time. The number of the epoch can also be dynamically set to meet the user's needs.

4.3.3 Callbacks

Callbacks are a set of functions that can be used to control the procedure of the training. By customizing the callbacks, the users can observe the internal states and statistics of the training. In this project, a list of functions are established as components of the callbacks.

Early stopping is an approach to avoid the overfitting problem[17]. As mentioned above, too many epochs can result in overfitting. Early stopping is to monitor the model's performance dynamically and stop the training if the generalization ability is wakened. The module of callbacks in TensorFlow Keras is used and the function EarlyStopping is adapted. The patience is set to be 8 and the monitor is 'val_loss', meaning that if indicator 'val_loss' is not improved for 8 epochs, the training will stop.

The learning rate controls the size of the step that the weights adjust after each iteration. If the learning rate is set to be too large, the training process is unstable and can skip the optimal point and result in a sub-optimal result. If the learning rate is too small, it will take a long time for the algorithm to converge.

The configuration of the learning rate is important as it affects efficiency and accuracy. As the training goes on, the learning rate should change to fit different stages. For example, at the beginning of the training, the learning rate may be large so the algorithm can move forward the optimal fast. After training for a while, it is better to decrease the learning rate which enables the weights to have finer adjustment.

A learning rate scheduler is used to dynamically change the learning rate. In this project, the learning rate is set to be 0.001 for the first ten epochs. When the training process exceeds 10 epochs, the learning rate will slightly shrink by multiplying a negative power of e.

4.3.4 Optimizer

Setting an optimizer also support adjust learning rate dynamically. Keras provides some common optimizers to achieve adaptive learning rates like SGD, RMSProp, Adam and so

on[18].

SGD is Stochastic Gradient Descent[19], which is an iterative method to optimize the objective function, that is, the loss function. For big data problem, it is effective as it speeds up the iteration for at the cost of lower convergence rate.

RMSProp optimizer is like gradient descent with momentum, where the oscillations are restricted in the vertical direction[20]. Thus increasing the learning rate can make the algorithm take larger steps and converge faster in the horizontal direction.

Adam is the short of Adaptive Moment Estimation[21]. Inherit from the Adaptive Gradient Algorithm (AdaGrad) and RMSProp, Adam realizes the benefit of both and furthermore makes use of the exponential moving average of the gradient and the squares gradient. Adam is proved to be effective in most cases. The default configuration of its parameter is good enough so it is easy to use.

5 Classification

5.1 Model Comparison

This paper tested the performance VGG16, VGG19, ResNet50 and InceptionV3 models on the classification of bats. The naive transfer learning is conducted by loading the pre-trained model and add one dense layer with 16 hidden units and one softmax layer. The softmax layer is influenced by the number of categories which is 6 in this project. All these six models are having more than 80 percent accuracy in the training set and more than seventy percent accuracy in the test set. The results show that VGG16 performs best in this case with an accuracy of test data to be 83.43%. The InceptionV3 has the lowest accuracy of 70.37% All these models perform obvious over-fitting because of the lack of fine-tuning. But generally, the classification is very good. To solve the problem of over-fitting, a more elaborated adjustment should be conducted. And it is necessary to compare and test the proper value of hyperparameters.

Models	Training Accuracy	Test Accuracy
VGG16	94.77%	83.43%
VGG19	89.72%	78.36%
InceptionV3	81.64%	70.37%
ResNet50	86.05%	73.41%

Table 3: Transfer Learning for Classification

5.2 Model fine-tune

In order to improve the performance of the classification model, VGG16 is chosen as the baseline model since it has the best test set accuracy. Start from the origin of the VGG16 model, the library of TensorFlow Keras provides a convenient way to load the pre-trained model. The model contains a total of thirteen convolution layers which are consist of 3*3 convolution filters and max-pooling layers. Max pooling layers are used for downsampling. Two fully connected layers with 4096 hidden units in each layer are built after 13 layers and the last layer is is a dense layer with 1000 units. 1000 refers to the total category of images in the ImageNet dataset.

```
base_model = VGG16(weights='imagenet', include_top=False,
                   input_shape=(224, 224, 3))
base_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 9: VGG16 Structure

For the purpose of transfer learning, the last three layers are abandoned and can be customized to meet specific needs. The above 13 layers can be regarded as feature extractions if we freeze the convolution blocks and use the pre-trained weights. The extracted features are flattened as bottleneck features of shape 1*8192 and can be feed into the classifiers built by the user. In this bats classification project, there are 6 breeds, so the prediction layer is made to be dense of six with a softmax activation function.

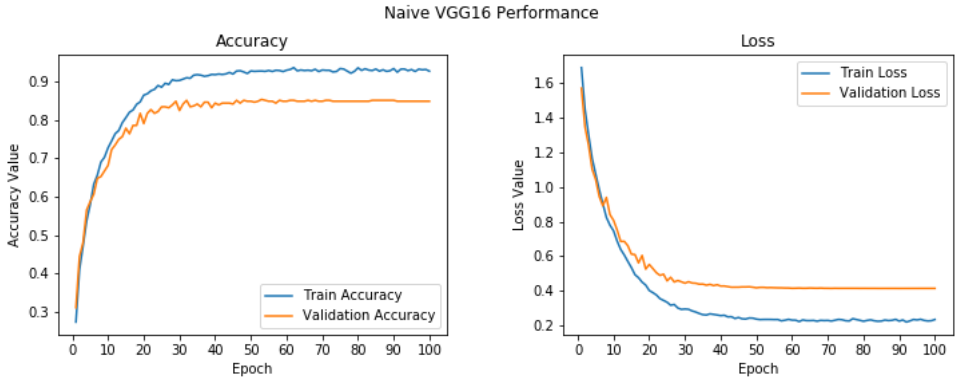

Figure 10: VGG16 Performance

The naive transfer learning model performs a problem of overfitting, which is common in deep learning. There are several ways to help to solve the problem.

Hidden Units

The number of hidden units determines the number of connections between the two layers. It is an important parameter as it directly determines how many weights to train in the CNN model. Using more hidden units will increase the weights that need to be trained, which can extract more features to increase accuracy. However, too many hidden units will bring the problem of overfitting. To test the performance of the different number of hidden units, one dense layer is added above the last classification layer. Five grades of unit size are tested, from 16 to 256.

Hidden units	Training Accuracy	Test Accuracy
16	81.61%	77.63%
32	86.27%	80.05%
64	90.87%	82.95%
128	95.25%	87.30%
256	96.86%	87.67%

Table 4: Hidden Units

The table shows that an increase of hidden units in the project can make the model performs better significantly. the largest hidden unit size is tested up to 256, but there is still potential to increase.

Batch Size

Based on the above model, setting the number of hidden units of the hidden layer before the last layer to be 256, a fine-tune of batch size is conducted. Five grades of batch size starting

from 4 to 64 are trained with all other parameters the same. Each model is trained for 40 epochs, which is tested to be a proper number from the early stopping experience. The figure shows that except for the batch size of 64, all other batch sizes converge similarly. 64 batch size model is not so accurate as of the others. The large batch size makes the gradient updates slower, so the modeled may be undertrained.Small batch size enables the weights to be updated more frequently. If the batch size is too small, the gradient can fluctuate drastically. In the validation set, the larger batch size is obviously smoother than the small batch size.

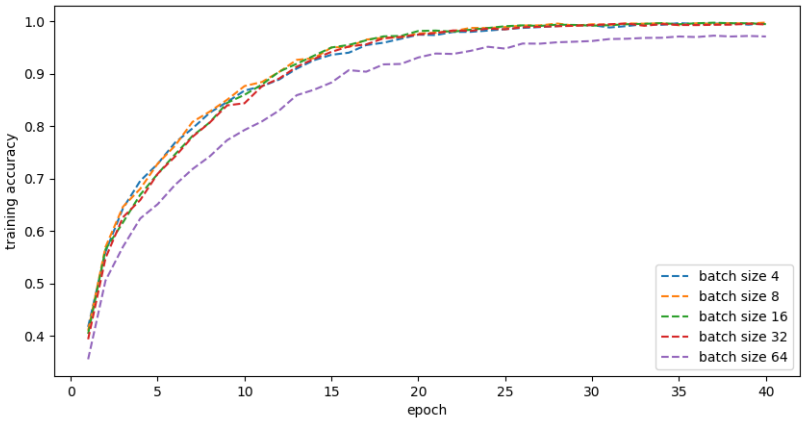

Figure 11: Batch Size-Training Accuracy

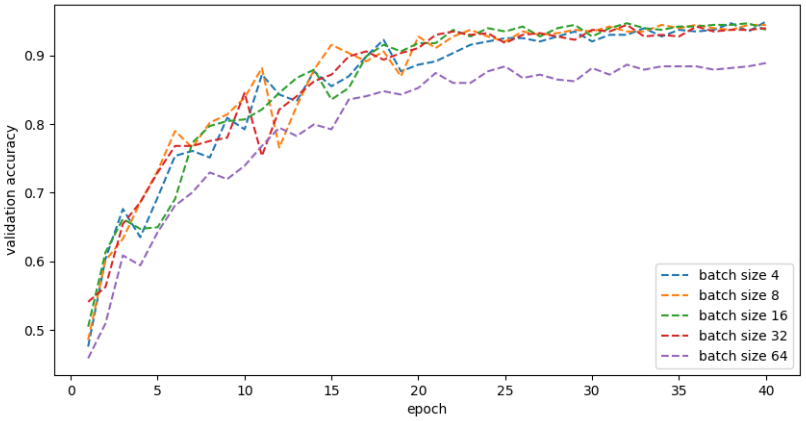

Figure 12: Batch Size-Testing Accuracy

Optimizer

This project tests three optimizers that are commonly used. Label 0 refers to the Adam optimizer, 1 is Stochastic Gradient Descent and 2 is the RMSprop optimizer. It is clear

that the Stochastic Gradient Descent optimizer falls into the local minimum. Stochastic Gradient Descent uses a batch size of 1 to train the model, making the gradient fluctuate too much and skip the actual optimal point. Adam and RMSprop optimizer finally converge to a similar accuracy while Adam optimizer converges faster and smoother than RMSprop.

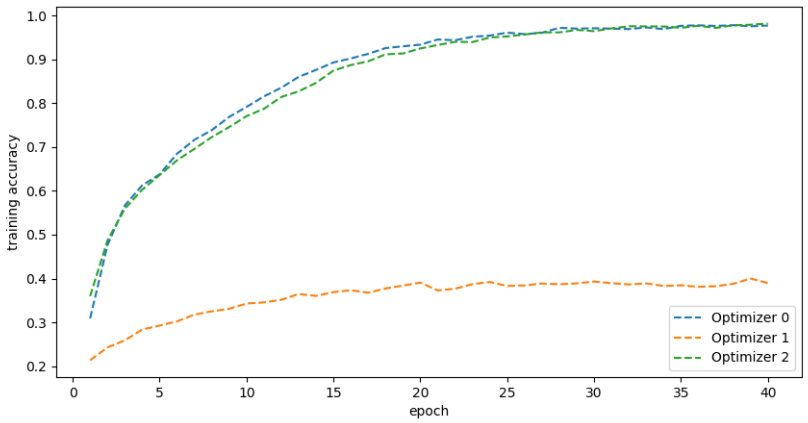

Figure 13: Optimizer-Training Accuracy

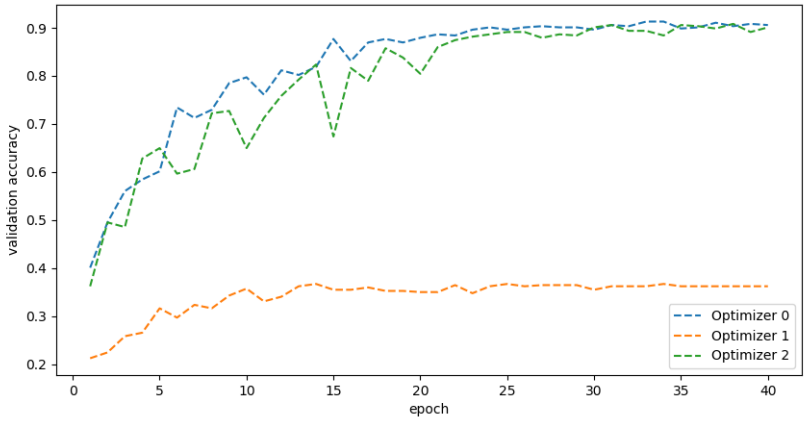

Figure 14: Optimizer-Testing Accuracy

Cross Validation

After fine-tuning the parameters, another problem needs to be dealt with is the overfitting problem. One of the most commonly used methods is cross-validation. Cross-validation is a resampling procedure that enables us to make good use of the whole data. In the Keras library, there are no integrated cross-validation functions as the Scikit-learn[22], so in this project, the hard code K-fold method is conducted. The general procedure of the k-fold is that:

1. Randomly shuffle the data
2. Split the data into several groups, where the number of the group is configured by the user
3. For each group of data, take it to be test set and the rest to be training set
4. Summarize the performance and evaluates the skill of the model

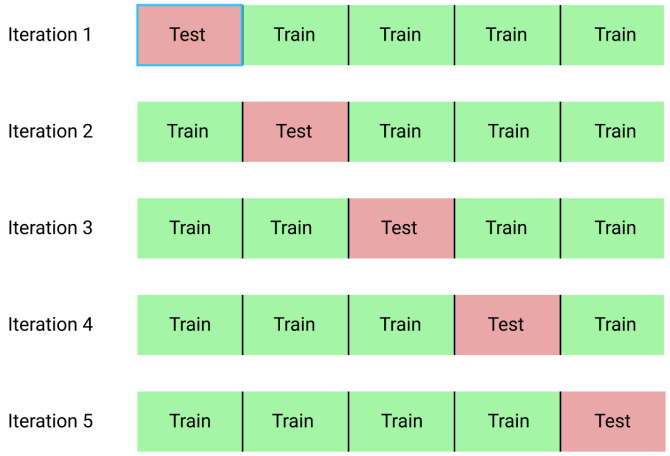

Figure 15: Five group K-fold

K-fold guarantees every single data can be training data as well as testing data, which largely eliminates the sampling bias. The drawbacks are the choice of parameter K. If the data is split into 3 groups the training size is about 66.7% of the whole data. For the small datasets, the sample for training can be insufficient. If setting K to be a large number, the testing set is of small size and the model needs to be trained for K-1 times which may be exhausted.

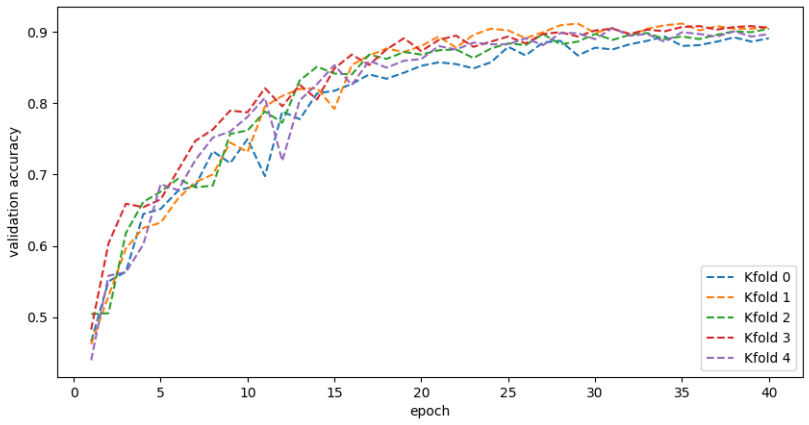

Figure 16: K-fold validation

Hidden units	Training Accuracy	Test Accuracy
K-fold 0	98.79%	89.12%
K-fold 1	99.52%	90.69%
K-fold 2	99.00%	90.45%
K-fold 3	98.91%	90.57%
K-fold 4	99.30%	89.71%
Average	99.10%	90.11%

Table 5: K-fold Validation

6 Object Detection GUI

To extend the topic and enhance the usability of this project, a YoloV3 weight for object detection is also trained. The labeled images are processed to meet the requirement of the darknet. The training is employed on the Google Cloud Platform with Tesla P4 Graph Process Unit. Both the fine-tuned VGG16 model for classification and the YoloV3 weights are integrated into a graphical user interface, where enables the user to choose an image and classify or detect bats with a simple click.

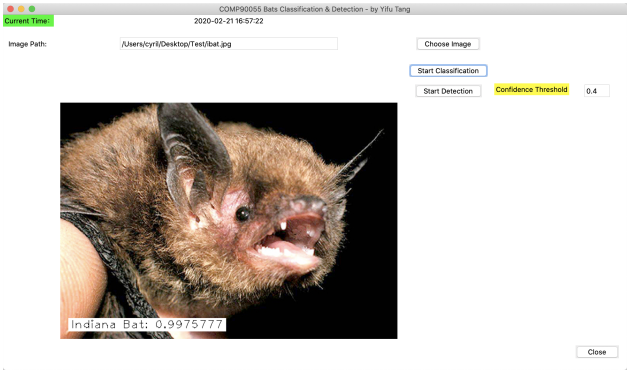

Figure 17: Classification

For detection, a confidence threshold is required. The model generally performs satisfying. The breed of the bat will show at the left bottom corner with the softmax confidence and for detection there will be a frame and confidence showed.

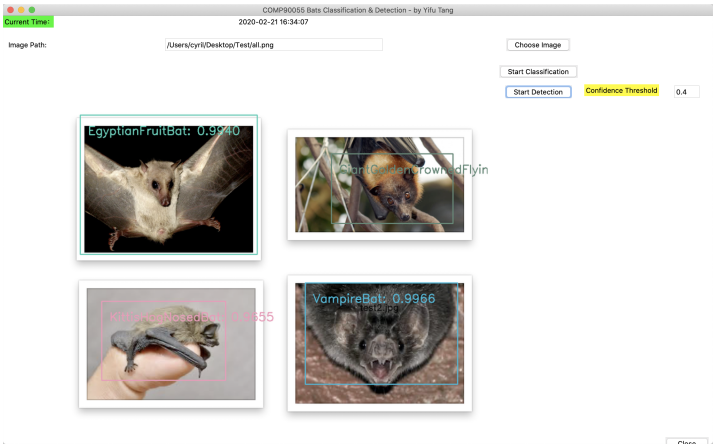

Figure 18: Object Detection

7 Challenges and Future Work

The Keras library provides an easy way to load pre-trained weights of many classical models for transfer learning. There are some problems when conducting transfer learning. For models that include Batch Normalization layers like ResNet50, extra conduction is needed to transfer the weight right. Batch Normalization layer is to standardize the weights of the last layer thus the average and the variance are calculated. In the training part of transfer learning, the BN layer inherited the average and variance from the weights of ImageNet, while for validation, these two values are calculated from the new dataset. The different characters between different domains make the models perform badly in the test set. An easy way to solve this problem is to use the average and variance from the ImageNet in the validation process as well. This method is not precise but avoids train the whole model.

This project is implemented in the range of six breeds of Bats without including any kind of other objects. So even if the image is not a bat at all, the model will still classify or detect it as one of the six breeds but with a low confidence level. For example, the false image of a batman is also classified and detected. Future work can include other objects like some birds to extend the functions.

Figure 19: Batman Detection

Most of the sample images contain clear characteristics of the bats' shape or face, which limits the predicting power of the model. If the input is part of a bat, for example, a wing or a leg, the model fails to work. To achieve a detailed detection, the quality of data is crucial. Future works can take care of collecting detailed images with characters of part of the bat.

8 Conclusion

This project uses transfer learning to explore the performance of classification and detection of bats. A complete procedure from data collection, data process to model loading and fine-tuning is employed. VGG16 outperforms VGG19, ResNet50 and InceptionV3 models as the naive feature extractor and an elaborate adjustment of hyper-parameters is implemented. A dense layer of 256 hidden units is added and the best model is tested to be trained with the batch size of 32 using Adam optimizer. The K-fold method is used to guarantee the randomness of the sampling dataset. The final model provides an accuracy of 99.10% in the training set and 90.11% on the test set. Besides classification, object detection is also implemented using Darknet and the trained weights of YoloV3 achieved good performance. The classification and detection models are integrated into a simple GUI, which provides the function of visualization. The model can still be improved with regard to detecting bats among different animals and recognize part of the bat. Possible improvements in the future are also discussed at the end.

References

[1] Peng Zhou, Xinglou Yang, Xian-Guang Wang, Ben Hu, Lei Zhang, Wei Zhang, Hao-Rui Si, Yan Zhu, Bei Li, Chao-Lin Huang, Hui-Dong Chen, Jing Chen, Yun Luo, Hua Guo, Ren-Di Jiang, Mei-Qin Liu, Ying Chen, Xu-Rui Shen, and Xi Wang. A pneumonia outbreak associated with a new coronavirus of probable bat origin. *Nature*, 02 2020.

[2] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, 61:85–117, 2015.

[3] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 153–160, Cambridge, MA, USA, 2006. MIT Press.

[5] Oscar Day and Taghi M. Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4:1–42, 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 06 2016.

[7] Minyoung Huh, Pulkit Agrawal, and Alexei Efros. What makes imagenet good for transfer learning? 08 2016.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. pages 1–9, 06 2015.

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and ZB Wojna. Rethinking the inception architecture for computer vision. 06 2016.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[12] Joffrey Leevy, Taghi Khoshgoftaar, Richard Bauder, and Naeem Seliya. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5, 12 2018.

[13] Vijay Vemuri. Pattern recognition and machine learning. *Journal of Information Technology Case and Application Research*, 21:1–4, 07 2019.

[14] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48, 2019.

[15] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, page 3–24, NLD, 2007. IOS Press.

[16] Leon Bottou. *Stochastic Gradient Descent Tricks*, volume 7700 of *Lecture Notes in Computer Science (LNCS)*, pages 430–445. Springer, neural networks, tricks of the trade, reloaded edition, January 2012.

[17] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Comput*, 7, 10 1998.

[18] Sebastian Ruder. An overview of gradient descent optimization algorithms. 09 2016.

[19] M. Taddy. *Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions*. McGraw-Hill Education, 2019.

[20] Christian Igel and Michael Hüsken. Improving the rprop learning algorithm, 2000.

[21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[22] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Edouard Duchesnay, and Gilles Louppe. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12, 01 2012.

Appendix

Source Code Link: Bats Classification & Detection
Dataset Link: Bats dataset
Video Demo of GUI: Youtube Video

Training Platform:

Google Cloud Platform: 8 standard vCPUs, 30 GB RAM, NVIDIA Tesla P4