Asian Giant Hornets Recognition Using Deep Convolutional Neural Network

Haochen Wang 1, a, *, Ruofeng Yang 2, b and Xu Li 1, c

1 School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China
2 School of Naval Architecture, Ocean & Civil Engineering, Shanghai Jiao Tong University, Shanghai, China

*, a Corresponding author e-mail: wanghc0409@gmail.com, b wanshuiyin@sjtu.edu.cn, c rgstone@sjtu.edu.cn

Abstract. We trained 2 deep neural networks to identify whether the images with high resolution in the dataset provided by COMAP contain any Asian giant hornet. We divide the classification problem into two subproblems: feature extraction, and image classification. In order to reduce the impact of sample imbalance, we apply image flipping and Borderline-SMOTE methods for data augmentation first, and then divide the data into the training set and the validation set (testing set). Next, we utilize auto-encoder to collect key features of images and the testing loss dips to 0.0274 after training. Next, we establish the Classification-Net to solve the identification problem based on the features just extracted, and the accuracy in testing set reaches 0.8030. Finally, we summarize the main features of having negative labels from three aspects: species characteristics, subject definition and background softness.

Keywords: Deep Learning, Image Classification, Auto-Encoder, Convolutional Neural Network.

1. Introduction

In the past decade, deep learning has been made a great progress in both computer vision [1], speech recognition [1], [3], and even in playing Atari games [4]. A wild range of neural network architectures have been utilized including fully connected neural networks, convolutional neural networks, and recurrent neural network to those problems. In the area of computer vision, clustering images into one of several predefined classes, called image classification, is a fundamental problem, and there have been a large number of scholars who have made great breakthroughs [5]-[8]. The first feature extracted from images was the characteristics of handwriting, which was considered as inputs to the classifier. The accuracy of the task always greatly depends on methods of feature extraction, and the structure of the classifier [5].

Recently, deep learning models, which process for feature extraction and transformation, pattern analysis, and classification with multilayer nonlinear information, have been shown to overcome these dilemmas. Among them, the convolutional neural networks (CNNs) are gradually becoming the mainstream architecture for image classification, semantic segmentation, and instance segmentation with the maturity of computing technology using GPUs.
Most of the CNNs take the original images as input directly and include several fully connected layers in the last few layers to classify the images, resulting in a significant increase in model computational complexity when the image size increases. Krizhevsky A. et al., design a huge deep neural network with 60 million parameters to solve the classification problem for the ImageNet dataset, which has over 15 million labeled high resolution images belonging to roughly 22,000 categories, and they trained the model on two NVIDIA GTX 580 3GB GPUs for several days [1]. However, there are some techniques that are able to convert high-dimensional data to low-dimensional codes, such as principal component analysis (PCA) [10], and factor analysis [11]. Moreover, Hinton G. E. and Salakhutdinov R. R. presents a method to reduce the dimension of data based on deep neural networks, which works much better than original methods, named auto-encoder [12].

In this paper, we first design an auto-encoder based on CNN to lower the dimension of input images, called Auto U-Net. After that, we build a neural network with convolutional layers and dense layers to solve the classification problem, taking the returns of Auto U-Net as inputs, outputting the probability that the image belongs to each category, named Classification-Net. The detailed structure of the models will be introduced in Section 3. We use the model for the determination of the Asian Giant Hornet, which is a binary classification problem. More information for the dataset can be seen in Section 2. In Section 4 and Section 5, we will introduce some tips in both preprocessing and training. Finally, there will be some discussions in Section 7.

2. Dataset Description
The dataset [13] contains 2134 reports of sightings with the observation date, notes, official classification (ground truth), latitude and longitude, and the corresponding images taken on the spot. COMAP organized a competition in February 2021, to find the most accurate algorithm for the identification of the Asian Giant Hornet. Due to the uneven quality of photos and similar characteristics of different species of bees, the image recognition process is extremely difficult.

The dataset consists of variable-resolution images, while our model requires a constant input dimensionality. Therefore, we down sampled the images to a fixed resolution of $3 \times 256 \times 256$, based on the RESIZE module in PYTORCH [14]. No any other preprocess procedures are applied.

In particular, the positive and negative samples in this dataset are extremely unbalanced. There are 2043 sightings with negative status, 19 with positive, 67 with unverified and 5 with unprocessed. Such imbalance of positive and negative samples will lead to various problems in model training, e.g., under fitting (the model probably tends to predict all the images are negative) and gradient disappearance. In cases where positive sample images are difficult to obtain, we choose to apply data balancing using Borderline-SMOTE [15], the details of which will be presented in Section 4.2.
3. Architecture for Neural Networks

We trained our Auto U-Net on the raw RGB values of the pixels, and the output of the model has a fixed resolution $48 \times 16 \times 16$, which is only 0.0625 of the original size. Figure 1 and Figure 2 show the structure of the Auto U-Net and the Classification-Net respectively. The Auto U-Net contains 4 convolutional blocks in total, and each block contains a convolutional layer with a $3 \times 3$ kernel and both of which padding and stride equals to 1, a down sampling layer, whose stride equals to 2, and a ReLU activation function.

If we input an image $x_1 \in \mathbb{R}^C \times \mathbb{R}^H \times \mathbb{R}^W$ into one convolutional block, the output of the encoder will be in shape $(2C, \frac{H}{2}, \frac{W}{2})$, i.e., the output image $x_2 \in \mathbb{R}^{2C} \times \mathbb{R}^{H/2} \times \mathbb{R}^{W/2}$. As for the decoder, it contains 4 deconvolutional blocks, and each block has one deconvolutional layer, one up sampling layer, and the ReLU activation function. As opposed to the convolutional block, when we take an image $x_1 \in \mathbb{R}^C \times \mathbb{R}^H \times \mathbb{R}^W$ as input, the output image $x_2 \in \mathbb{R}^{C/2} \times \mathbb{R}^{H/2} \times \mathbb{R}^{W/2}$.

For the Classification-Net, it has 2 convolutional blocks, 2 fully connected hidden layers with ReLU function, and finally a Softmax function. The output of the Classification-Net is a two-dimensional
vector $\mathbf{Y} = (\bar{Y}_0, \bar{Y}_1)$, indicating the input image has a probability of $\bar{Y}_0$ being negative and a probability of $\bar{Y}_1$ being positive. Obviously, we have $\bar{Y}_0 + \bar{Y}_1 = 1$.

Figure 3. The workflow of predicting whether an input image contains an Asian Giant Hornet.

After extract features of input images by the Auto U-Net, we utilize the Classification-Net for forecasting. The workflow is shown in Figure 3, when we have an image to be recognized, we first put it into the Auto U-Net, and take the output of encoder as the input of the Classification-Net, and then we will get the probability of positive or negative (they are equivalent because they always add up to one).

4. Reducing Overfitting
Our model has $???$ parameters in total, so overfitting may occur when training with so many parameters inevitably. Below, we introduce the two main ways to overcome overfitting.

4.1. Data Augmentation
In this paper, we first flip the positive images in three directions (vertical, horizontal, and diagonal) to form 76 ($18 \times 4$) images including the original image. Then we select 28 of them as the testing set, the rest as the training set, and randomly select 38 negative images as the testing set. However, the number of positive images in the training set (48) is still significantly less than the number of negative images, so we use Borderline-SMOTE to balance the data.

4.2. Data Balancing
As mentioned in Section 2, the number of negative samples is much larger than the number of positive samples in the original data, and we utilize Borderline-SMOTE to deal with it.

SMOTE (synthetic minority oversampling technique) is an improved oversampling algorithm based on random sampling, which is easy to implement but has some obvious disadvantages at the same time [16]. SMOTE treats all the minority samples equally and does not consider the class information of the nearest neighbor samples, it often leads to aliasing of samples, resulting in poor classification effect. Borderline-SMOTE is an improved over sampling algorithm based on SMOTE, which only uses a few class samples on the boundary to synthesize new samples, so as to improve the class distribution of samples [15].

According to [17], when dealing with the problem of binary classification, the ratio of positive and negative samples should not be less than $1:4$. Therefore, we sampled 697 observations from the original 48 positive training sets through Borderline-SMOTE for subsequent training.
4.3. Dropout

Apparently, simply combining the predictions of many different models is a very successful way to reduce testing errors, but it is too expensive for big neural networks. However, there is an efficient version of model combination that only costs about a factor of two during training, called dropout [18], which consists of setting to zero the output of each hidden neuron with a certain probability $\alpha$. The neurons be dropped out do not participate in the forward pass and do not contribute to back propagation. After trying plenty of different values of $\alpha$, we found that it works best when $\alpha = 0.25$.

5. Details of Training and Results

We trained our models using stochastic gradient descent with a batch size of ??? examples, and RMSProp of ???. We utilize a zero-mean Gaussian distribution with standard deviation 0.01 to initialize the weights of each layer, and the initial biases are zero. Moreover, we used a constant learning rate ??? for all layers.

**Auto U-Net.** We set the input image as the ground truth for the decoder, and set the loss function as MSE, which can be calculated as follows:

$$f(x^1, x^2) = \| x^1 - x^2 \|_2^2,$$  \hspace{1cm} (1)

where $x^1$ and $x^2$ are the output of decoder and the ground truth respectively. Figure 4 (left) shows the trend of training losses and testing losses with the increase of training iterations. The minimum testing loss during the training is 0.0274 after training 1605 iterations. Figure 4 (right) contains examples of input image and decoded image, output by the model with the smallest testing loss.

**Classification-Net.** We use the official classification (negative refers to 0 and positive refers to 1) to be the ground truth, and set a 0.25 dropout in the first dense layer. We utilize the Cross-Entropy loss while training, which can be calculated as follows:

$$H(p, q) = -\sum_x p(x) \log(q(x)),$$  \hspace{1cm} (2)

where $p$ and $q$ are two different probability density functions of the same random variable $x$, and $p$ represents the ground truth and $q$ is the current estimation. In our case, $x$ is an input image, and both $p$ and $q$ are the PDFs of a binary distribution, whose output implies the negative and positive probability of input $x$. Figure 5 (left) shows the trend of losses and the accuracy in testing set. Similar to the Auto U-Net, we regard the model with the maximum accuracy in testing set as the best model, and save its weights and biases. The maximum accuracy reaches 0.8030 after training 490 iterations. Figure 5 (right) contains 6 examples predictions sampled randomly, 5 of which are correct.
6. Discussion

To have a better insight, we choose 10 images with the highest negative probability shown in Figure 6. The predictions about these pictures are all correct, i.e., ground truths of images above are all negative.

To summarize, pictures containing following features are tend to be recognized as negative:

1. **Other species in the picture, not wasps, let alone the Asian giant hornet.** Obviously, there is an insect resembling a dragonfly in Figure 6a. Figure 6e is definitely not an Asian giant hornet, because the Asian giant hornet should have wings. Figure 6j contains something similar to a cocoon.

2. **The insect is too small.** For both Figure 6f, Figure 6d, and Figure 6h, the insects are too small to distinguish.

3. **The background is difficult to discriminate from the insect itself.** For Figure 6f and Figure 6g we can’t even see where the insect is at a glance because of its background.

7. Conclusion

In conclusion, for the image classification problem, we first clean the data and apply Borderline-SMOTE method to deal with the imbalanced training set and apply a series of methods to augment data. Next, we build the Auto U-Net model to extract features in images based on auto-encoder, of which the
minimum testing loss is to 0.0274. After that, we design the architecture of the Classification-Net mainly use convolutional layers and fully connected layers, of which the maximum testing accuracy is 0.8030. Finally, we summarize the features of negative images from three aspects: species characteristics, subject definition and background softness.

References
[1] Krizhevsky A., Sutskever I., Hinton G. E. “ImageNet Classification with Deep Convolutional Neural Networks”. Advances in Neural Information Processing Systems, vol.25, pp.1097-1105, 2012.
[2] Graves A., Mohamed A., Hinton G. “Speech Recognition with Deep Recurrent Neural Networks”. IEEE International Conference on Acoustics, Speech and Signal Processing, pp.6645-6649, 2013.
[3] Dahl G. E., Yu D., Deng L., et al. “Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition”. IEEE Transactions on Audio, Speech, and Language processing, vol.20, no.1, pp.30-42, 2011.
[4] Mnih V., Kavukcuoglu K., Silver D., et al. “Playing Atari with Deep Reinforcement Learning”. arXiv preprint arXiv:1312.5602, 2013.
[5] Rawat W. and Wang Z. “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review”. Neural Computation, vol.29, no.2, pp.2352-2449, 2017.
[6] Ciresan D. C., Meier U., Gambardella L. M., and Schmidhuber J. “Deep, Big, Simple Neural Nets for Handwritten Digit Recognition”. Neural Computation, vol.22, no.12, pp.3207-3220, 2010.
[7] Ciresan D. C., Meier U., Masci J., Maria Gambardella L., and Schmidhuber J. “Flexible, High Performance Convolutional Neural Networks for Image Classification”. 21st International Joint Conference on Artificial Intelligence, vol.1, pp.1237-1242, 2011.
[8] Ciresan D., Meier U., and Schmidhuber J. “Multi-Column Deep Neural Networks for Image Classification”. IEEE Conference on Computer Vision and Pattern Recognition, pp.3642-3649 2012.
[9] LeCun Y. “Generalization and Network Design Strategies”. Connectionism in perspective, vol.19, pp.143-155, 1989.
[10] Pearson K. L. III. “On Lines and Planes of Closest Fit to Systems of Points in Space”. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol.2, no.11, pp.559-572, 1901.
[11] Catell R. B. “Factor Analysis: an Introduction to Essentials II. The role of factor analysis in research”. Biometrics, vol.21, no.2, pp.405-33, 1965.
[12] Hinton G. E., Salakhutdinov R. R. “Reducing the Dimensionality of Data with Neural Networks”. Science, vol.313, no.5786, pp.504-507, 2006.
[13] Dataset for the Asian Giant Hornet provided by COMAP for MCM2021 problem C. [online] Available: http://www.mathismore.net/mcm/2021MCM_ProblemC_Files.rar.
[14] Documentation for torchvision. transforms. functional.resize in PYTORCH. [online] Available: https://pytorch.org/vision/stable/transforms.html?highlight=resize#torchvision.transforms.functional.resize.
[15] Han H., Wang W. Y., Mao B. H. “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning”. International conference on intelligent computing, pp.878-887, 2005.
[16] Chawla N. V., Bowyer K. W., Hall L. O., et al. “SMOTE: Synthetic Minority Over-Sampling Technique”. Journal of artificial intelligence research, vol.16, pp.321-357, 2002.
[17] Zheng M., Li T., Sun L., et al. “An Automatic Sampling Ratio Detection Method Based on Genetic Algorithm for Imbalanced Data Classification”. Knowledge-Based Systems, vol.216, pp.106800, 2021.
[18] Hinton G. E., Srivastava N., Krizhevsky A., et al. “Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. arXiv preprint arXiv:1207.0580, 2012.