Problem Chosen	2021 MCM/ICM Summary Sheet	Team Control Number
C		2126909

Asian Giant Hornet Detection Data Processing

Due to the discovery of Asian giant hornets in Vancouver Island, Canada in 2019, the presence of Asian giant hornets brought a great deal of panic since the potential serious impact of local honeybees and people. Therefore, the State of Washington started to create some platforms for people to report their sightings of these hornets. In this paper, we are asked to analyze those public reports of Asian giant hornets.

To begin with, we preprocess the given data and confirm that only a tiny amount of the sightings are real Asian giant hornets while the most of remaining sightings are other types of insects. Also, we deal with data lacking reports or images. Moreover, we do not see a clear pattern between neither detection date and the lab status nor detection place and the lab status. So we ignore those unrelated data instead only select data related to the lab status and merge them in pandas.

What's more, we classify the remaining data into four categories based on their lab status: "Positive ID" as 0,"Negative ID" as 1,"Unverified" as 2, and "Unprocessed" as 3.

Then, we utilize the theorem of TF-IDF and Multinomial Naive Bayes to filter out the most useful words in the report notes. Meanwhile, by using the most meaningful words in the positive and negative cases as indicators, we are able to build the NLP(natural language processing) Model.

After testing, it turns out that the NLP model's accuracy is not enough to filter out most of the negative cases. So we use the images as primary indicators to develop the Fine Tuning VGG16 Model. This model takes into account a wide variety of relevant theorems and technique such as convolution, padding and max pooling. We preprocess the image data to same size and use PCA(principal component analysis) to confirm that VGG16 modelling is workable on our image data set.

Last but not least, by applying the Fine Tuning VGG16 Model and choosing random pictures from the given data set, we are able to get correct predictions of the report lab status at 97 percent of the time. Moreover, you can update daily additional reports with light hardware support. If you get many positive cases confirmed by our model in a day, this would constitute evidence that the hornet has been eradicated in the report area and emergency operation is needed.

Keywords: *Deep Learning Model, Machine Learning Model, Fine Tuning VGG16 Model, Convolutional Neural Network*

Table of Contents

Contents

Memo

To: Washington State Department of Agriculture
From: Team #2126909
Date: 8 February 2021
Subject: Asian Giant Hornet Prediction Suggestion

Dear Governors,

We are reaching out to you because we can offer you a workable strategy to analyze public reports of Asian giant hornet.

Summary for Public Reports

Among the provided 4000+ cases, we only see 14 positive cases, but tons of negative and unverified cases. Also, detection dates and places do not have a clear pattern. Moreover, there is always a delay between detection date and lab result submission date. This delay is reasonable because it takes time and effort to perform lab investigation. So it is not reasonable to predict using the tiny sample of positive cases or detection dates and time data. But we can alternatively develop a model to determine whether the report is a negative case and filter those reports out.

Prediction Model for Negative Reports

With the given report data, we first categorize them into four categories based on their lab status. Then we extract the report notes as well as related images and ignore detection date and place data. For notes analysis, you can use the NLP Model but its accuracy score is not ideal. But the image analysis using the Fine Tuning VGG16 Model can produce highly accurate result under our testing.

Model development

To be specific about the model development, first we have been trying to build a simple NLP(Nature Language processing) Model so that we just need to input a document containing all sighting reports notes and get a result indicating which lab status it might be after further investigation. By using the idea of TF-IDF and Multinomial Naive Bayes, we are able to manipulate the words in the document. Then, the given document is transformed into an array containing only the keywords which are used to classify whether this sighting is a real report of Asian giant hornet.

However, after comparing the results filtered from the document and the confirmed cases of Asian giant hornet, we realize that this model is only correct

54 percentage of the time, which is certainly not accurate enough for the computer identifier. To solve the low accuracy problem, we decide to build another model using image recognition instead of text filtering. We utilize the concept of Fine Tuning VGG16 and CNN(Convolutional Neural Network) to reduce the dimension of the report pictures. After preprocessing of the input raw pictures from the reports, VGG16 could transform a two-dimensional picture into a one-dimensional array containing the features of the given picture. This is what we called Feature extraction. Meanwhile, VGG16 assembles models of neural networks with the features that we extracted. After those operations of turning the raw data into a informative array, a computer can easily interpret the array and use learned knowledge from neural networks to identify whether the given picture is a real Asian giant hornet. Random image testing has shown that the VGG16 model accuracy is about 97 percent, which is high enough for a classifier.

Recommended Actions

Since the image processing model(Fine Tuning VGG16 Model) would identify the negative reports accurately 97 percent of time, we highly recommend to require an image of the hornet for every report.

Wish our proposal can help you organize the reports and save effort in lab investigation. If you have any questions, please contact us and we are looking forward to hearing from you.

Yours,
Sincerely
Team #2126909 of modelers who love data science

1 Introduction

1.1 Problem statement

In September 2019, Asian giant hornet, also known as *Vespa mandarinia*, had spread over Vancouver Island, where the first confirmed sighting located. Popular media usually call this hornet "murder hornet"; because after rapid reproduction, this hornet would hunt the European honeybees, invade and destroy honeybees' nests. Moreover, *Vespa mandarinia* is the largest species of hornets in the world. Its 6mm stinger with potent venom could kill an adult in the case of multiple hornets stinging simultaneously. So in order to protect the honeybees, it is important to detect the hornet location precisely and use proper method to control its reproduction. Therefore, in response to increasing confirmed sightings of Asian giant hornet, the state of Washington has created a helpline and websites for suspecting reports of the sightings of these Asian giant hornets. Even though some of these reports have been confirmed that those sightings are *Vespa mandarinia*, most of these sightings have been determined as other hornet species. Since the volume of public reports is huge and only a tiny amount of those reports are truly positive cases for *Vespa mandarinia*, we are asked to interpret the data of those public reports and develop a model to prioritize them for additional lab investigation.

1.2 Goals

- Address and discuss whether or not the spread of this pest over time can be predicted, and with what level of precision
- Using the given data, develop a reliable model to filter out the mistaken reports for the hornet and leaving the most likely hornet reports for further investigation
- Discuss how our classification analyses leads to prioritizing investigation of the reports most likely to be positive sightings
- According to the above analysis, decide how often to update the data with additional reports
- Use the established model, decide whether the hornet has been eradicated in Washington State

2 Methodology

2.1 General definition of parameters/variables

First, we went through the given data set and classify the data set by Lab Status. There are four types of Lab status: "Positive ID","Negative ID","Unverified",

and "Unprocessed". So we categorize the data as follows:

Positive ID :0
Negative ID :1
Unverified :2
Unprocessed :3

2.2 General Assumptions

The spread of this pest over time can not be predicted. Based on our analysis for lab status over different time intervals, also shown in the following figure; the detection date of most of cases were reported in recent two years. The number of positive cases are negligible compared to that of negative cases. Meanwhile, dates do not have a clear pattern. As a result, we believe that the spread of this pest can not be predicted. However, we also find that there is a period between detection date and lab result submission date. By using our model, we could shorten this period so that once the image is uploaded, our model could automatically recognize whether it is an Asian Giant Hornet or not. Thus, we could "predict" the pest in another perspective.

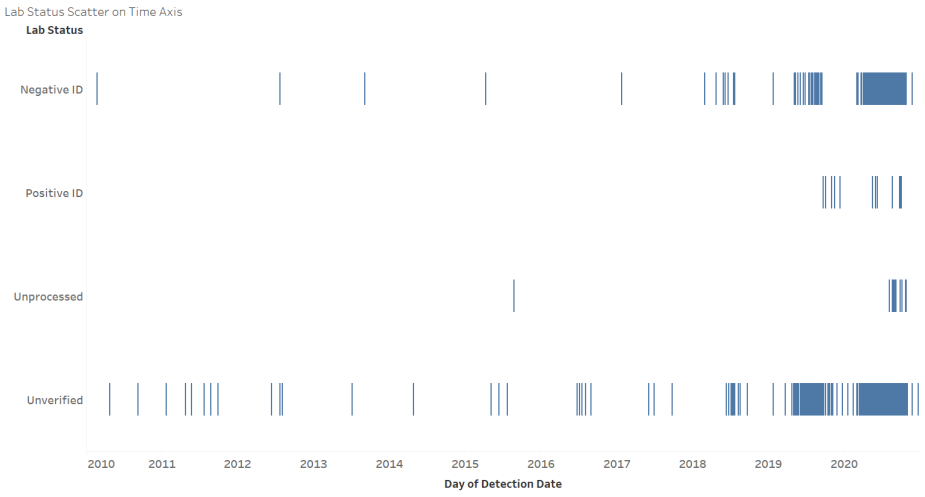

Figure 1: lab status scatter

Therefore, we decide to develop models to determine whether the report is a negative case. Then, by using our model we are able to filter out the ones that do not need additional lab investigation. In general, here are the main assumptions for our model:

- For the four categories, the detection place (Both Longitude and latitude of the report) does not affect the result because the natural environment near

Vancouver Island will not change. Moreover, the differences of longitude and latitude between reports are not essential to recognize. So we can treat longitude and latitude as constants. This assumption is reasonable and it could simplify our analysis.

- The detection date will not affect the result because people might not report the hornet detection immediately and they may see the dead body of the hornet instead of a live one. So the detection date is not synchronize with the live cycle of the hornet. Therefore, we do not use detection date as a factor in our models.

Notes NLP (Nature Language processing) Model: we assume the notes would be the only reliable source to determine whether the report is a negative case.
Fine-tuning VGG16 Model(Convolutional Network for Classification and Detection): We assume the pictures are more reliable than notes to determine the possible lab status.

2.3 Data Preprocessing

As the problem is given, "2021MCMProblemC_DataSet.xlsx" is a spreadsheet containing 4440 reports of suspecting sightings of Asian giant hornets. Manual screening is inefficient and its error rate is too high. Thus, our goal here is to use the computer so that it comprehend our problem and identify the data that we need: the real Asian giant hornet.

2.3.1 Notes NLP (Nature Language processing) Model

Since this is a data analysis problem, we need to introduce how we are going to treat incomplete data in the spreadsheet. Firstly, we import the given data set to two data-frames by using Pandas, a Python package. Then we classify the data into four categories based on their lab status(as described in 2.1):

	GlobalID	Detection Date	Notes	Lab Status	Lab Comments	Submission Date	Latitude	Longitude
0	{5AC8034E-5B46-4294-85F0-5B13117EBEFE}	2019-12-08 00:00:00	One dead wasp seen in Blaine, and suspect flyi...	0		2020-01-15	48.980994	-122.688503
1	{5EAD3364-2CA7-4A39-9A53-7F9DCF5D2041}	2019-10-30 00:00:00	Hornet specimen sent to WSU	0		2020-01-15	48.971949	-122.700941
2	{13B67BCB-AFCE-4100-AD2B-76EF178BA228}	2020-01-15 00:00:00	Massive loss of bees, decapitated. No hornet s...	2		2020-01-15	48.939200	-122.661300
3	{124B9BFA-7F7B-4B8E-8A56-42E067F0F72E}	2019-09-19 00:00:00	This was the colony that was found and destroy...	0	Thanks for adding this, and the great pictures!	2020-02-04	49.149394	-123.943134
4	{BBBA5BA0-CAFB-43D3-8F1D-FB2D9CF777E0}	2019-08-31 00:00:00	I was cleaning my gutters when I heard a snapp...	2	Thanks for this report. I can't verify it from...	2020-02-14	48.723779	-122.354431

Figure 2: TF-IDF Lab Status

Then we found some incomplete variables missing the notes or pictures in the given raw data. We just delete those data because they are nothing to con-

tribute to our model. We also delete all irrelevant data columns:"GlobalID", "Detection Date","Lab Comments","Submission Date","Latitude",and "Longitude". Then only the "Note" and "Lab status" column are left for NLP model analysis:

	Notes	Lab Status
0	One dead wasp seen in Blaine, and suspect flyi...	0
1	Hornet specimen sent to WSU	0
2	Massive loss of bees, decapitated. No hornet s...	2
3	This was the colony that was found and destroy...	0
4	I was cleaning my gutters when I heard a snapp...	2

Figure 3: TF-IDF Base

2.3.2 Fine-tuning VGG16 Model

First, we import the given data set and the image id data set to a data-frame in Pandas. Then we classify the data into four categories based on their lab status and delete the irrelevant columns as we did for the NLP Model. Since the Global ID links the the image with its report notes, we merge two data set together based on the Global ID. Moreover, since our model cannot work on pdf or video files, we delete files that are not in jpg or png type.

	Lab Status	FileName
829	1.0	ATT192_20200517_142140.jpg
2070	1.0	ATT1038_55590E40-99DE-4027-B20F-064B6E8C8EC7.jpg
2677	1.0	ATT1428_63AA55EC-DAEB-4934-A69A-D2B4090CBDB4.jpg
5505	1.0	ATT3242_20201012_141753.jpg
5850	0.0	B (233).jpg
...	...	...
5753	0.0	B (136).jpg
174	2.0	ATT74_image0 (27).jpg
346	1.0	ATT136_90CDFD8E-DC18-4DA3-A3C4-7F523755CDFD.jpg
3751	1.0	ATT2086_2B0A0898-1BF8-4C47-827C-08F2720CE482.jpg
3599	1.0	ATT2001_20200815_113742.jpg

Figure 4: VGG Deleted

From the NLP model, we see a large portion of the data set are negative or unverified cases and little positive cases.

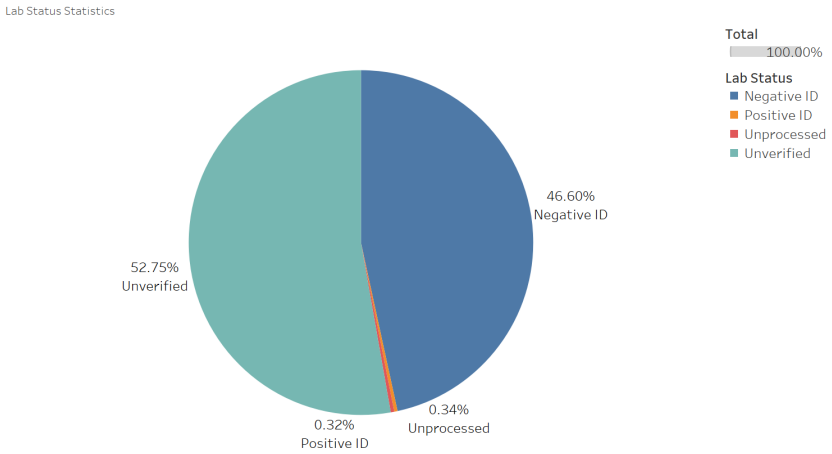

Figure 5: Percentage of Cases

So, we perform data augmentation to the positive cases. A larger data set is what we get, around 1000 positive images, by increasing the contrast and brightness of the pictures, rotating, and flipping the images. Then we import OpenCV to convert all the images to a RGB matrix and resize them to 150 $\times$ 150.

```
import cv2
```

```
IMG_SIZE = 150

def create_datasets(result2, img_size):
    imgs = []
    paths = []
    for path in tqdm(result2['FileName']):
        img = cv2.imread(path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (img_size,img_size))
        imgs.append(img)
        paths.append(path)
    imgs = np.array(imgs, dtype = 'float32')
    imgs = imgs / 255.0
    #df_ = pd.get_dummies(result['Lab Status'])
    df_ = np.array(result2['Lab Status'], dtype = 'int32')
    return imgs, df_, paths

train_imgs, train_df, train_paths = create_datasets(x_train, IMG_SIZE)
test_imgs, test_df, test_paths = create_datasets(x_test, IMG_SIZE)
pre_imgs, pre_df, pred_paths = create_datasets(result, IMG_SIZE)
```

Figure 6: VGG16 Import

In order to make sure that we can use VGG16 and CNN to process the data, we use principal component analysis(PCA) to analyze the features of the data as follows. From the analysis, we conclude that the features of positive cases are separated from those of negative cases, so using a fine-tuning VGG16 model to classify images is workable.

```
from sklearn import decomposition

pca = decomposition.PCA(n_components = 2)

X = train_features.reshape((n_train, x*y*z))
pca.fit(X)

C = pca.transform(X) # Représentation des individus dans les nouveaux axe
C1 = C[:,0]
C2 = C[:,1]
```

```
### Figures

plt.subplots(figsize=(10,10))

for i, class_name in enumerate(class_names):
    plt.scatter(C1[train_df == i][:1000], C2[train_df == i][:1000], label = class_name, alpha=0.8)
plt.legend()
plt.title("PCA Projection")
plt.show()
```

Figure 7: PCA Code

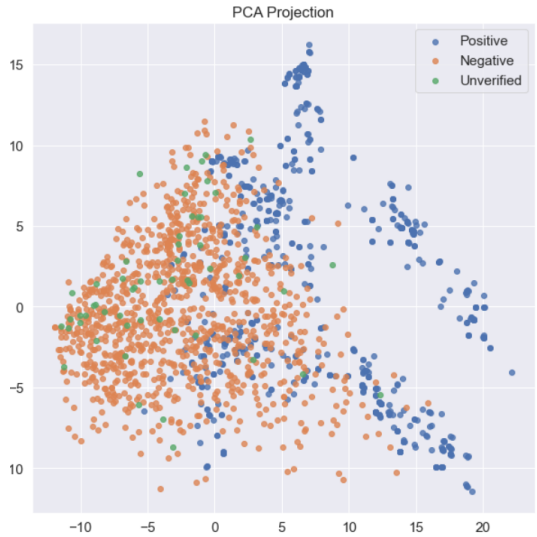

Figure 8: PCA Image

3 NLP Model

3.1 Theorem and Technique

3.1.1 TF-IDF(term frequency-inverse document frequency):

TF-IDF is a numerical statistic method that intended to evaluate the importance of a word in a corpus or data set. The importance of this word increases proportionally as with the frequency of its appearance in the data, but decreases with the frequency of its appearance in the corpus.

1. term frequency: the frequency that the given word appears in the given document. In other word, term frequency is the total number of times that the given word divided by the total number of all words in document. As figure 9 shown, if a word's term frequency is high in one article and rarely in other articles, it would be considered that this word has good classification ability.

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Figure 9: Term Frequency's Formula

2. inverse document frequency: frequency representing the measure of universal importance of a word, showing how much information does this word provide. It is the logically inverse function of the percentage of document containing the given word. On the other hand, inverse document frequency is the number of document containing the given word divided by the total number of documents, and then take the logarithm of that quotient. As figure 10 shown, if there are fewer documents containing the given word, that is, the higher the given word's inverse document frequency is, then the given word has good classification ability.

$$\mathrm{idf_i} = \lg \frac{|D|}{|\{j : t_i \in d_j\}|}$$

Figure 10: Inverse Document Frequency's Formula

3.1.2 Multinomial Naive Bayes:

Naive Bayes classifier use the Bayes' theorem firstly to calculate marginal probability and then the distribution of the conditional probability. As the figure 11 shown, *P(Y|X)* is conditional probability: given that event B is true, the likelihood event A happens; *P(X|Y)* is also conditional probability: given that event A is true, the likelihood event B happens. *P(X)* and *P(Y)* are probability of observing separate events, event A and event B, respectively.

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)}$$

Figure 11: Bayes' Theorem

In a multi-series events, a multinomial could represent the frequency that a certain event happen, based on a sample or feature vector. *P(Y)* becomes *P(C=c)*, which is the number of documents could be considered as category c divided by the total number of documents. *P(X)* becomes *P(w|c)*, which is the times that word x appear in the documents under category c divided by the total number of words that belong to category c. Therefore, the series of events x would be transform into a histogram, where x_i is the number of times that event i occur in a specific timezone. Then, the likelihood of histogram x happens:

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

Figure 12: Multinomial Naive Bayes

By the utilization with TF-IDF weights instead of traditional raw term frequency, we could generate a naive Bayes classifier that is competitive with Machine Learning.

3.2 Nature Language processing Model

As above methodology described, we have already preprocess the data that supplanted the lab status with numbers and deleted all the NaN functions. By using the NLTK user-package and setting the stop keyword, we could extract the specific characters from the articles to a new table containing only the variables and words that we need. As figure 13 shown, among filtered words in the given data set, we have already delete all meaningless words such as 'am', 'is' and 'are'.

	Notes	Lab Status
0	one dead wasp see blaine, suspect fly nearby	0
1	hornet specimen sent wsu	0
2	massive loss bees, decapitated. hornet specime...	2
3	colony found destroyed nanaimo, bc sep 18, 201...	0
4	cleaning gutter heard snap noise come wall hou...	2

Figure 13: Filtered Document

(a) Positive Keywords (b) Negative Keywords

Figure 14: Key words from positive and negative cases

After the filter part, we could use the TF-IDF method that explained above to restructure the data. After importing TF-IDF vectorizer and applying it on the filtered data, the data set has been transformed into a matrix containing series of 1 time n variables. It could be understood as two sets: train set and test set, as shown below:

Last but not least, we use machine learning algorithm, Multinomial Naive

```
Tfidf_train: (3552, 54160)
Tfidf_test: (888, 54160)
```

Figure 15: Train and Test Set

Bayes, to fit the Multinomial Naive Bayes for TF-IDF features so that the machine could understand the model. Therefore, this accuracy score is what we get from the NLP model:

```
mnb_tfidf_score : 0.5472972972972973
```

Figure 16: MNB Result

4 Fine-tuning VGG16 Model

4.1 Theorem and Technique

4.1.1 VGG16

- **How does computer comprehend the data of a picture**
  If we try to understand and view a picture from the computer aspect, we would consider a picture with the RGB color channels that constructed by red, green and blue. These three channels could be represented as two-dimensional array where its pixel value is represented by a number that counts from 0 to 255.

- **Convolution**
  Convolution is a process that utilizing the convolution core to scan the pixel matrix of each layer by step. The value that we get from the scanning would be multiplied by the number of corresponding position in convolution core. Thus, we would get a smaller matrix containing the key value of above layer. Convolution core is equivalent to a filter that extracts the features of the above layer.

- **Padding and Pooling**
  During the process of convolution, it is important that the value in the center might be extracted several times repeatedly. Meanwhile, the number of extraction for boundary value is relatively small. Thus, we would add a surrounding value of 0 in original layer during the process of convolution so that we could use the boundary value efficiently. This is what we called padding. Pooling is equivalent to dimensional reduction. What we used is max pooling since the extracted value that we get from convolution would have similar information which are able to substitute each other. Easily to say, max pooling is the process of picking the max value in the area where values inside are alternative. Pooling is able to continuously reduce the number of parameters and complexity of calculation.

- **Fully Connected**
  For a random layer n and its above layer *n-1*, every node in *n-1* layer is connected with every node in n layer. In other word, while doing the calculation part for every node in n layer, the input of activation is the weight of all nodes in *n-1* layer.

- **Softmax**
  Softmax squashes all outputs of each unit to be between 0 and 1; therefore, the output of softmax function would be equivalent to the distribution of categorical probability.

- **How does VGG16 works**
  VGG16 is a convolutional neural network 16-layer CNN model that achieves the greatest accuracy scores in ImageNet. As shown above, we are able

to use the convolution, max pooling, fully connected and softmax to compress a two-dimensional image into a one-dimensional array containing readable data for computer. Therefore, we tuned some of the layers of VGG16 and an example model is shown below.

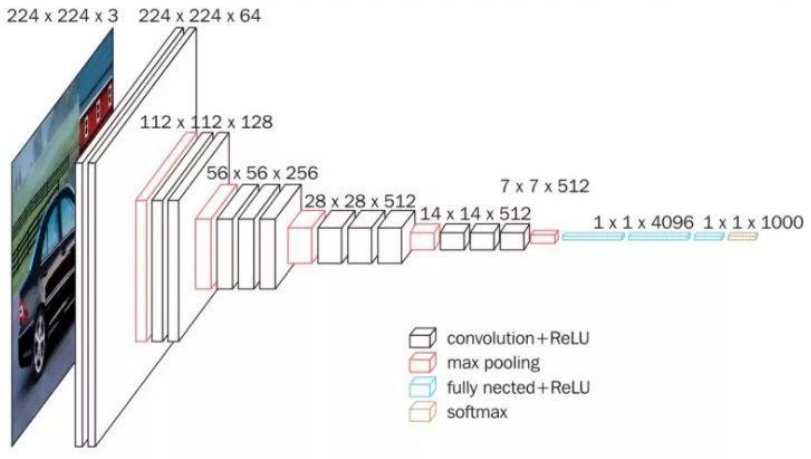

Figure 17: composition principle of VGG16

Based on our PCA projection analyse result, we recognize that it is an effective way to use a fine tuning VGG16 that trained on ImageNet to classify our images. Firstly, we perform transfer learning to extract features of convolution layer for test set and train set by VGG16 model.
We use sparse categorical cross-entropy as our loss function. And our new model use the 4th layer count backwards of VGG16 as our new input layer and the size is 9, 9, 512. We also add a dropout layer to make sure the model does not over fit. Then, we use max-pooling, pool size 2 * 2, to half the length and width. After flatten the two dimension array to one dimensional array, it goes to two full connected layer: one using ReLU as activation function and other one using softmax.
In conclusion, our new model has around 7.7 million total parameters and 7.7 million trainable parameters.

```
: new_model.summary()

Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 9, 9, 512)]       0
_________________________________________________________________
block5_conv1 (Conv2D)        multiple                  2359808
_________________________________________________________________
block5_conv2 (Conv2D)        multiple                  2359808
_________________________________________________________________
block5_conv3 (Conv2D)        multiple                  2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   multiple                  0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 2, 64)          294976
_________________________________________________________________
dropout_2 (Dropout)          (None, 2, 2, 64)          0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                3250
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 306
=================================================================
Total params: 7,377,956
Trainable params: 7,377,956
Non-trainable params: 0
_________________________________________________________________
```

Figure 18: Parameters of VGG16

4.2 Test Results and Model evaluation

To train our model, we give the batch size a value of 128 and 10 epochs. The training process takes about thirty seconds with a computer that contains the following device: CPU: Intel i5-7500, RAM:8GB DDR4, NVIDIA GeForce GTX 1060 6GB. The training process is shown in figure 19.

```
Epoch 1/10
19/19 [==============================] - 4s 126ms/step - loss: 2.1604 - accuracy:
0.6152 - val_loss: 0.3887 - val_accuracy: 0.8682
Epoch 2/10
19/19 [==============================] - 2s 89ms/step - loss: 0.3629 - accuracy:
0.8626 - val_loss: 0.2501 - val_accuracy: 0.9161
Epoch 3/10
19/19 [==============================] - 2s 89ms/step - loss: 0.2099 - accuracy:
0.9341 - val_loss: 0.2474 - val_accuracy: 0.9247
Epoch 4/10
19/19 [==============================] - 2s 91ms/step - loss: 0.1801 - accuracy:
0.9544 - val_loss: 0.1346 - val_accuracy: 0.9709
Epoch 5/10
19/19 [==============================] - 2s 91ms/step - loss: 0.1188 - accuracy:
0.9695 - val_loss: 0.0965 - val_accuracy: 0.9743
Epoch 6/10
19/19 [==============================] - 2s 90ms/step - loss: 0.1053 - accuracy:
0.9741 - val_loss: 0.0920 - val_accuracy: 0.9812
Epoch 7/10
19/19 [==============================] - 2s 96ms/step - loss: 0.0847 - accuracy:
0.9781 - val_loss: 0.1034 - val_accuracy: 0.9726
Epoch 8/10
19/19 [==============================] - 2s 89ms/step - loss: 0.0741 - accuracy:
0.9813 - val_loss: 0.1215 - val_accuracy: 0.9692
Epoch 9/10
19/19 [==============================] - 2s 88ms/step - loss: 0.0600 - accuracy:
0.9820 - val_loss: 0.0894 - val_accuracy: 0.9795
Epoch 10/10
19/19 [==============================] - 2s 94ms/step - loss: 0.0361 - accuracy:
0.9858 - val_loss: 0.1001 - val_accuracy: 0.9795
```

Figure 19: Training Process

The training accuracy versus value accuracy history as well as the training loss versus value loss history are shown in figure 20.

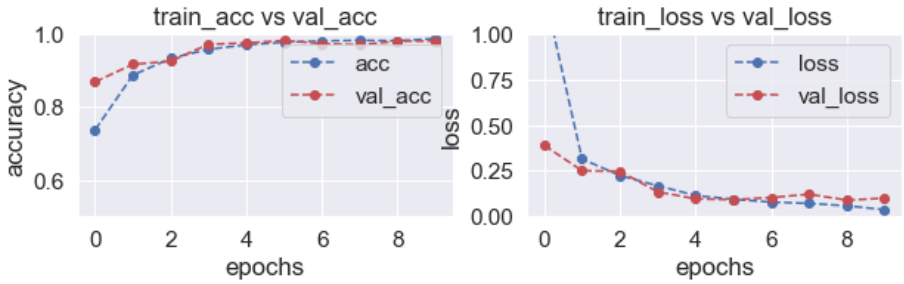

Figure 20: Accuracy history

To evaluate a model, we use an idea called confusion matrix(figure 21).

Figure 21: Confusion Matrix

There are four states: true positive, true negative, false positive and false negative. Thus, we are able to use the confusion matrix to calculate the accuracy score and the equations is in (figure 22).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + +TN + FN}$$

Figure 22: Accuracy Score

As a result, our fine-tuning VGG16 model reaches an accuracy score of 0.97(figure 23).

```
In [54]: from sklearn.metrics import accuracy_score
         predictions = new_model.predict(test_features)
         pred_labels = np.argmax(predictions, axis = 1)
         print("Accuracy : {}".format(accuracy_score(test_df, pred_labels)))

         Accuracy : 0.9768555466879489
```

Figure 23: Accuracy Result

After filtering the data from the two given data set, we find eleven unprocessed cases with recognizable images(figure 24).

	Lab Status	FileName
5598	3.0	ATT3295_42311DF4-A9FC-48E0-80A0-44FB7F9D1BD0.jpg
5602	3.0	ATT3296_Screenshot_20201023-035213_Blink.jpg
5603	3.0	ATT3297_Screenshot_20201023-035325_Blink.jpg
5604	3.0	ATT3298_Screenshot_20201023-035341_Blink.jpg
5605	3.0	ATT3299_Screenshot_20201023-035247_Blink.jpg
5606	3.0	ATT3300_Screenshot_20201023-035401_Blink.jpg
5610	3.0	ATT3301_42D408C0-95F0-4121-B0CF-D3AF977F25B0.png
5612	3.0	ATT3302_tmp-cam-4811883243317525499.jpg
5613	3.0	ATT3303_5F5E645F-0249-4CF9-84E2-9E7BCD1CE348.jpg
5614	3.0	ATT3304_3F3414C0-1943-4ABC-BA74-241D6C0EAA35.jpg
5615	3.0	ATT3305_AE3995C6-528B-4F16-BDF3-A6F5DBFFA534.jpg

Figure 24: Unprocessed Cases

Figure 25: Unverified Cases

We use our model to predict these images and it returns a result that all the images are not the hornet we want(negative) except one with file name "ATT3302_tmp-cam-4811883243317525499.jpg"(unverified). (figure 25).
Also, we predict some images of our test set and get a reasonable final result.(figure 26)

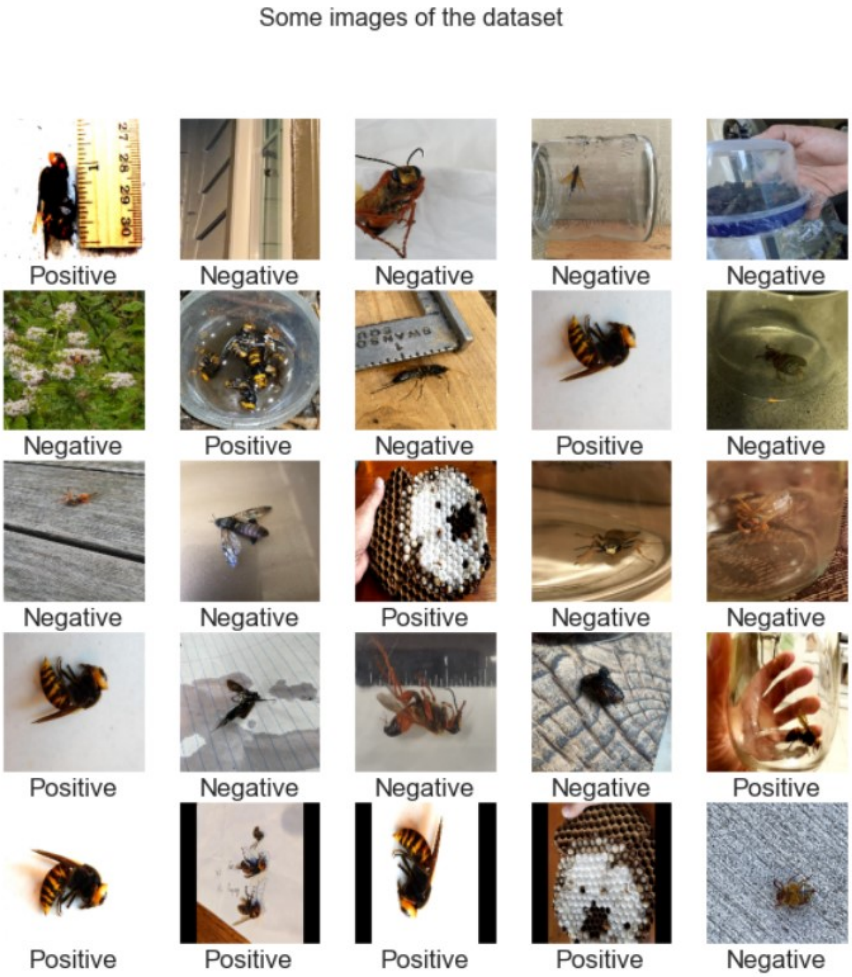

Figure 26: Final Results

Therefore, we suggest that we could update our model given additional reports once a day. Because our model is a small neural network model, it only needs light hardware support. The training speed on some intro level GPUs is fast enough. Also, adding new cases could help our model increase its accuracy.

5 Conclusions

We tried to develop a data processing model to help the governors filter the public hornet reports. We only need the most possible positive hornet reports to go to further lab investigation. After performing data analysis and modeling, we successfully complete this task. First, we use report notes to decide whether it is a negative report(ones are not Asian giant hornets and do not need lab investigation) and construct the NLP model. However, the NLP model only have 54 percent of accuracy. So alternatively, we manage to use report images to determine whether it is a negative report and develop a Fine Tuning VGG16 Model. It turns out that analyzing images using the Fine Tuning VGG16 Model works well on the given data set with an accuracy score of 0.97.

5.1 Strengths

- The model could predict the images at 0.97 accuracy score, which is a precise enough to recognize most of the images.
- The model is a small model compared with other CV models, which saves training time when updating new data set.
- The model could predict cases from not only image perspective but also language perspective.

5.2 Weaknesses

- If we do not have image or language data, we cannot analyze the report.
- We neglect data about detection place and time, which might help determine whether the hornet could live there at the detection date using biological knowledge of Asian giant hornets.

Finally, using our NLP and Fine Tuning VGG16 model and updating daily, if one get many positive reports confirmed in a day, there must be a hornet devastation happening in the detection place.

6 References

1. Baker, Mike. "'Murder Hornets' in the U.S.: The Rush to Stop the Asian Giant Hornet." The New York Times, 2 May 2020. NYTimes.com, https://www.nytimes.com/2020/05/02/us/asian-giant-hornet-washington.html.

2. "Asian Giant Hornet." Wikipedia, 7 Feb. 2021. Wikipedia, https://en.wikipedia.org/w/index.php?title=Asian_giant_hornet&oldid=1005313445.

3. Miller, George A. "WordNet: A Lexical Database for English." Communications of the ACM, vol. 38, no. 11, Nov. 1995, pp. 39–41. DOI.org (Crossref), doi:10.1145/219717.219748.

4. Su, Jiang, et al. Large Scale Text Classification Using Semi-Supervised Multinomial Naive Bayes. p. 8.

5. Culjak, I., et al. "A Brief Introduction to OpenCV." 2012 Proceedings of the 35th International Convention MIPRO, 2012, pp. 1725–30.

6. Yang, Fan, et al. "How to Reduce Dimension with PCA and Random Projections?" ArXiv:2005.00511 [Math, Stat], May 2020. arXiv.org,http://arxiv.org/abs/2005.00511.

7. Xia, Denan, et al. "Insect Detection and Classification Based on an Improved Convolutional Neural Network." Sensors, vol. 18, no. 12, 12, Multidisciplinary Digital Publishing Institute, Dec. 2018, p. 4169. www.mdpi.com,doi:10.3390/s18124169.

8. Brownlee, Jason. "How to Develop an Ensemble of Deep Learning Models in Keras." Machine Learning Mastery, 20 Dec. 2018, https://machinelearningmastery.com/model-averaging-ensemble-for-deep-learning-neural-networks/.

9. PhD, Aisha Sikder. "Building Kriging Models in R." Medium, 9 Oct. 2020, https://towardsdatascience.com/building-kriging-models-in-r-b94d7c9750d8.

10. VGG16 - Convolutional Network for Classification and Detection. 20 Nov. 2018, urlhttps://neurohive.io/en/popular-networks/vgg16/.

7 Appendices

Here are part of the python code to help you understand the model: Confusion matrix and VGG16 import:

```
from sklearn.metrics import confusion_matrix
import seaborn as sn; sn.set(font_scale=1.4)
class_names = ['Positive', 'Negative', 'Unverified']
CM = confusion_matrix(test_df, pred_labels)
ax = plt.axes()
sn.heatmap(CM, annot=True,
           annot_kws={"size": 10},
           xticklabels=class_names,
           yticklabels=class_names, ax = ax)
ax.set_title('Confusion matrix')
plt.show()


# In[42]:


from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input

model = VGG16(weights='imagenet', include_top=False)


# In[43]:


train_features = model.predict(train_imgs)
test_features = model.predict(test_imgs)


# In[44]:


n_train, x, y, z = train_features.shape
n_test, x, y, z = test_features.shape
numFeatures = x * y * z


# In[45]:


from sklearn import decomposition

pca = decomposition.PCA(n_components = 2)

X = train_features.reshape((n_train, x*y*z))
pca.fit(X)

C = pca.transform(X) # Repr sentation des individus dans les
    nouveaux axe
C1 = C[:,0]
C2 = C[:,1]
```

```
1052
# In[46]:
1054

1056 ### Figures

1058 plt.subplots(figsize=(10,10))

1060 for i, class_name in enumerate(class_names):
        plt.scatter(C1[train_df == i][:1000], C2[train_df == i][:1000],
         label = class_name, alpha=0.8)
1062 plt.legend()
     plt.title("PCA Projection")
1064 plt.show()

1066
# In[47]:
1068

1070 from keras.models import Model

1072 model = VGG16(weights='imagenet', include_top=False)
     model = Model(inputs=model.inputs, outputs=model.layers[-5].output)
1074

1076 # In[48]:

1078
     train_features = model.predict(train_imgs)
1080 test_features = model.predict(test_imgs)

1082
# In[49]:
1084

1086 from keras.layers import Input, Dense, Conv2D, Activation ,
        MaxPooling2D, Flatten, Dropout

1088 model2 = VGG16(weights='imagenet', include_top=False)

1090 input_shape = model2.layers[-4].get_input_shape_at(0) # get the
        input shape of desired layer
     layer_input = Input(shape = (9, 9, 512)) # a new input tensor to be
         able to feed the desired layer
1092 # https://stackoverflow.com/questions/52800025/keras-give-input-to-
        intermediate-layer-and-get-final-output

1094 x = layer_input
     for layer in model2.layers[-4::1]:
1096     x = layer(x)

1098 x = Conv2D(64, (3, 3), activation='relu')(x)
     x = Dropout(0.1)(x)
1100 x = MaxPooling2D(pool_size=(2, 2))(x)
     x = Flatten()(x)
1102 x = Dense(50,activation='relu')(x)
```

```
x = Dense(6,activation='softmax')(x)

# create the model
new_model = Model(layer_input, x)


# In[50]:


new_model.compile(optimizer = 'adam', loss = '
    sparse_categorical_crossentropy', metrics=['accuracy'])


# In[51]:


new_model.summary()


# In[52]:


history = new_model.fit(train_features, train_df, batch_size=128,
    epochs=10, validation_split = 0.2)


# In[53]:


plot_accuracy_loss(history)


# In[54]:


from sklearn.metrics import accuracy_score
predictions = new_model.predict(test_features)
pred_labels = np.argmax(predictions, axis = 1)
print("Accuracy : {}".format(accuracy_score(test_df, pred_labels)))


# In[55]:


pred_features = model.predict(pre_imgs)
our_predictions = new_model.predict(pred_features)
```

VGG16.py