Final Report: Happy Whale - Whale and Dolphin Identification
Identify whales and dolphins by unique characteristics

Borui Li
Department of Computer Science
Simon Fraser University
8888 University Dr W
liborui0320@gmail.com

Bowei Pan
Department of Computer Science
Simon Fraser University
Simon Fraser University
bpa27@sfu.ca

Wenxiang He
Department of Computer Science
Simon Fraser University
8888 University Dr W
wha61@sfu.ca

Abstract

Our project topic is "Happy whale - whale and dolphin identification". The purpose of out project is automatically distinguish between whales and dolphins through an automatic picture recognition system. Our project can help marine researchers automatically distinguish between dolphins and whales, so they do not need to spend a lot of time manually distinguishing pictures. We found that the automatic identification of whales and dolphins is similar to the automatic identification of human being that has been achieved so far,and the same principle can be used to achieve the purpose.

1 Intro

Many years ago, marine scientists manually identified pictures. This not only takes a lot of time, but also leaves a lot of precious data unused. The purpose of our project is to help marine scientists automatically identify these images, which can save a lot of time.

2 Data analysis

In this step, we first download the data we need from kaggle, and then analyze them.

2.1 Data(pictures) install

Initial setting (Download the dataset to colab form kaggle) Here we use small size of image in order to reduce storage burden Each picture's size is 128*128.

```
! pip install -q kaggle
!mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
! kaggle datasets download -d rdizzl3/jpeg-happywhale-128x128
! unzip jpeg-happywhale-128x128.zip
```

Figure1: Data download

2.2 TensorFlow framework

The framework we use for this project is called tensor flow, it is a very popular framework nowadays for AI learning.

```
%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

Figure2: TensorFlow

2.3 Create the programming environment.

Import some libraries that we will use in our project.

```
# These library are for data manipulation
import numpy as np
import pandas as pd

# These library are for working with directories
import os
from glob import glob
from tqdm import tqdm

# These library are for Visualization(making the plot for evaluation part)
import matplotlib.pyplot as plt
import plotly.express as px

# These Library are for converting Label Encoding
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

# These library are for building model
from tensorflow.keras import layers
import tensorflow.keras.backend as K
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.imagenet_utils import preprocess_input
from tensorflow.keras.layers import AveragePooling2D, MaxPooling2D, Dropout
from tensorflow.keras.layers import Input, Dense, Activation, BatchNormalization, Flatten, Conv2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau


import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Figure3: Useful libraries

2.4: data loading and initial data analysis.

We will plot a pie chart for all the species available in the training data with Ploty lib

https://www.kaggle.com/code/meetnagadia/happywhale-2022-using-cnn

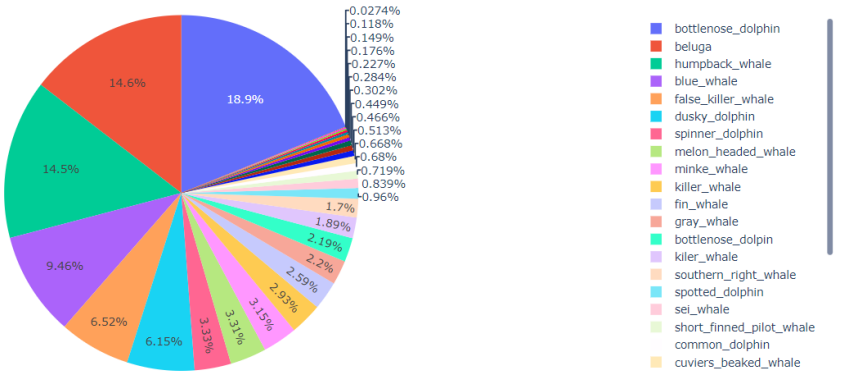

Figure4: Pie chart of data

The pie chart above shows the classification of the various whales of 51033 images.

We can also divide all data set into two parts: the whales and dolphins, and draw a pie plot reflecting the proportion of these two species in the training set

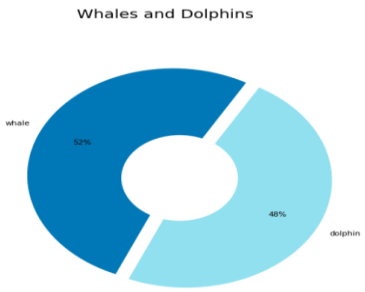

Figure5: Pie chart of data

2.5: Data processing

Prevent image overfitting by flipping, scaling, panning.

3 Model building

In this project, we will use the CNN framework developed by TensorFlow. First, we need to initialize this CNN framework according to our requirements, set some important parameters such as adding convolutional layers, pooling layers, spreading layers, and selecting activation functions.

3.1 The overall construction of the input layer, hidden layer and output layer for model

We will start building the CNN model, where we choose our processed data set as our input layer, for hidden layer, we will use convolutional layer with the maximum pooling layer pattern and repeat it three times, which is: cov1 -> maxpooling1 -> cov2 -> maxpooling2 -> Dropout1->cov3->maxpooling3->Dropout2. Between second layer and third layer, we use a dropout layer. which randomly invalidate some nodes to allow our neural network to generalize better and avoid overfitting the test data [1]

```
model = Sequential(
        [
          Conv2D(filters=32,kernel_size=(3,3),input_shape=(64,64,3),activation='relu'),
          MaxPooling2D(pool_size=(2,2)),

          Conv2D(filters=32,kernel_size=(3,3),activation='relu'),
          MaxPooling2D(pool_size=(2,2)),
          Dropout(0.25),

          Conv2D(filters=32,kernel_size=(3,3),activation='relu'),
          MaxPooling2D(pool_size=(2,2)),
          Dropout(0.25),

          Conv2D(filters=32,kernel_size=(3,3),activation='relu'),
          MaxPooling2D(pool_size=(2,2)),
          Dropout(0.25),

          Dense(128,  activation='relu'),
          Flatten(),
          Dense(max(smaller_processed_training_set.classes)+1,activation='softmax')

        ]
)
```

Figure6: overall code for model

After that, we use dense layer to link all nodes from the hidden layer, transform them using the activation function and pass them into the flattening layer as a new input layer[2].

Next, we will convert the n*n matrix into an n*1 vector in the spreading layer and then link the sense layer as the output of our entire CNN.

Here, we trained marine animal with a number of 5152.

4 Data training and evaluation

Based on the CNN model we created; we use it for our training data set. We also add function to get instant updated accuracy and loss during training.

4.1 Data training

In the process of training, we created 50 epochs, during each epoch we train 250 pictures.

Figure7: Epoch and epoch size

4.2 Instant updated accuracy and loss

We monitor the accuracy and loss rate of our model instantly during the training process, and collect their data after each epoch.

Table 1: Sample table title

Epoch 1/50	250/250 [==============================] - 24s 91ms/step - loss: 8.4173 - accuracy: 0.0068 - val_loss: 8.3887 - val_accuracy: 0.0090
Epoch 2/50	250/250 [==============================] - 21s 84ms/step - loss: 8.0734 - accuracy: 0.0077 - val_loss: 8.6202 - val_accuracy: 0.0090
Epoch 3/50	Output 250/250 [==============================] - 21s 83ms/step - loss: 7.9245 - accuracy: 0.0079 - val_loss: 8.7265 - val_accuracy: 0.0090
Epoch 4/50	250/250 [==============================] - 22s 86ms/step - loss: 7.7477 - accuracy: 0.0085 - val_loss: 8.8733 - val_accuracy: 0.0095
…	…
Epoch 48/50	250/250 [==============================] - 21s 83ms/step - loss: 4.4106 - accuracy: 0.1166 - val_loss: 15.6401 - val_accuracy: 0.0055
Epoch 49/50	250/250 [==============================] - 21s 83ms/step - loss: 4.3811 - accuracy: 0.1206 - val_loss: 15.4442 - val_accuracy: 0.0060
Epoch 50/50	250/250 [==============================] - 21s 83ms/step - loss: 4.3566 - accuracy: 0.1214 - val_loss: 15.7554 - val_accuracy: 0.0065

4.3 Figures and evaluation

The plots below show the change of accuracy and loss through training process.

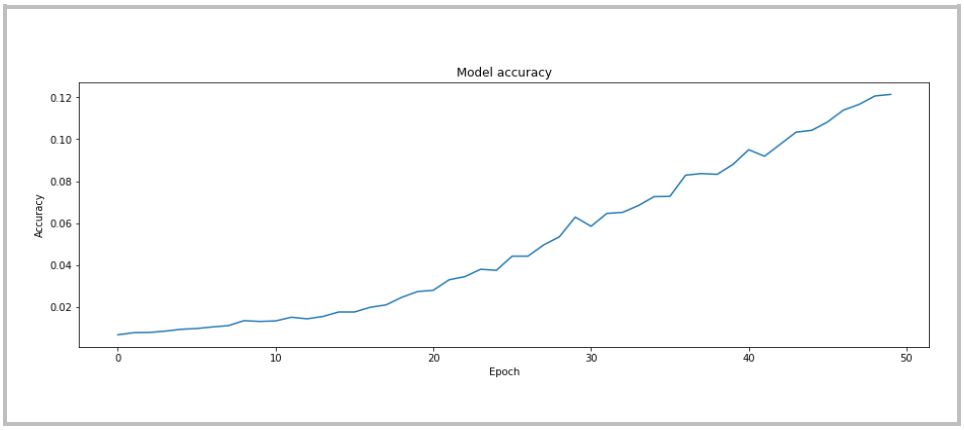

Figure8: Accuracy plot

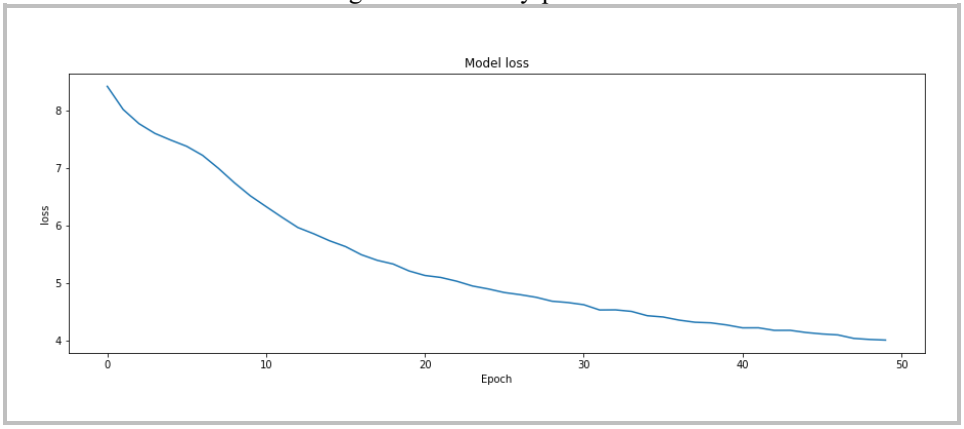

Figure9: Loss plot

As show in plots, after 50 epochs, our accuracy gradually increases from 0.0068 to 0.1214, and loss decreases from 8.4173 to 4.3566.

It takes approximately 1000 seconds to process the train set (10000 pictures), and the plots shows that our model is learning and being trained efficiently.

5 What can do to improve, pros and cons

Though we have trained through 10000 pictures and increase the accuracy, our accuracy is still quite low. We can improve through some methods.

- Change the activation function, this can help us solve vanishing gradient problem
- Add more hidden layer, this can increase the accuracy of model, but this also increase the depth of learning, so we also need to expand the size our dataset to prevent the overfitting.
- Increase the size of train data set, but this also increase our training load, which in turn extends our training time.

6 Conclusion

The above is the implementation of the technology we have done to identify whales and

dolphins through CNN， this technology increases the efficiency of identifying pictures, reduces the workload of scientists.

It can not only be used for the identification of whales and dolphins, but also can be widely applied to the identification of various marine lives. This provides technical support for the research and conservation of marine life, helps to better understand and manage the impact on the changing ocean, and helps to overcome the increasing human influence on the ocean.

7. Contribution

Borui Li did data processing and code implementation

Bowei Pan did final report abstract + part 1 – 2 and references

Wenxiang He did final report part 3-6

References

Code reference: https://www.kaggle.com/code/derrickmwiti/simple-convolutional-neural-network https://www.kaggle.com/code/meetnagadia/happywhale-2022-using-cnn

[1] Jason Brownlee (2016) Dropout Regularization in Deep Learning Models With Keras

(https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/)

[2] https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/