AI_BIRDER: USING ARTIFICIAL INTELLIGENCE AND DEEP LEARNING TO CREATE A MOBILE APPLICATION THAT AUTOMATES BIRD CLASSIFICATION

Charles Tian[1], Yu Sun[2]

[1]University High School, 4771 Campus Dr., Irvine, CA 92612
[2]California State Polytechnic University, Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

Birds are everywhere around us and are easy to spot. However, for many beginner birders, identifying the birds is a hard task [8]. There are many apps that help the birder to identify the birds, but they are often too complicated and require good internet to give a result. A better app is needed so that birders can identify birds while not depending on internet connection.

My app, AI_Bider, is mainly built in android studio using flutter and firebase, and the AI engine is coded with TensorFlow and trained with images from the internet [9]. To test my AI engine, I made six different prototypes, each having a different number of times that the code will train from the dataset of pictures. I then selected 5 birds that are in my dataset and found 5 pictures on the internet for each of them, which I then uploaded to the app. My app will then give me 3 bird species that most closely resemble the image, as well as the app's confidence in its choices, which are listed as percentages [6]. I recorded the percentages of accuracy for each picture. After taking the average percentage of all the models, I selected the most successful model, which had an average percent of accuracy of 79%.

KEYWORDS

Machine Learning, AI platform, Computer vision

1. INTRODUCTION

There are countless birders in the world, and many of them struggle with identifying the bird that they saw, which is the most important part of bird-watching. Many birders resorted to using field guides or existing birding apps like Merlin or Audubon birding Apps. However, the former methods are highly inefficient. A field guide requires the birder to memorize the features of the bird while having to flip through all the pages to find the bird. This method is highly impractical because not every birder has a field guide and the experience needed to efficiently use it. The other methods, the apps, are highly unreliable as well for the apps rely on the users to memorize details of the bird - color, size, tail shape, the type of the bird, its activity, habitat, voice, and wing shape - to accurately identify a bird. To an inexperienced birder, this could be a grueling task because a new birder has not likely been exposed to the different behaviors of birds, not to mention being able to spot body shapes and colors for a fast-moving target in the air. Learning from and improving on these common flaws of the former methods, I am going to create an app that can accurately identify birds using Artificial Intelligence, while only needing a picture of the bird from the user. The benefits of my app are that my app is going to be highly accessible to all birders for my app will be free to download from the google play store or the app store for IOS.

And also my app will be easy to use and successful because of the implementation of AI identification. Overall, my app has the potential to be very popular among birders and can be used instead of previous identification methods.

Using Artificial Intelligence to identify birds has been attempted by scientists before by Scientists from Kimberley, South Africa(Ferreira et al, 2020), and the Mutah University of Karak, Jordan(Al-Showarah and Al-qbailat, 2021) [3]. They used deep learning to accurately identify small birds and general birds, respectively. Ferriera's group used convolutional neural networks(CNN), a type of deep learning that automatically analyzes data like color, shape, and sizes(Ferreira et al, 2020). Al-Showarah and Al-qbailat used a type of CNN called VGG-19(l-Showarah and Al-qbailat, 2021). VGG-19 is a type of CNN that is pre-trained and can identify and distinguish different traits. This feature enables VGG-19 to identify objects like birds better than the conventional CNN models. Al-Showarah and Al-qbailat also used principal component analysis(PCA) to decrease the dimensional of the AI code while minimizing data loss [11]. The dimensional of a dataset is how many input variables or features that dataset has [10]. PCA, therefore, increases the efficiency of the code by significantly cutting down the time for the engine to train and produce an accurate model [12]. Both groups had a significant number of bird pictures that groups used as data for the AI engine. Al-Showarah and Al-qbailat used images from a database to train the model, whereas Ferriera's group captured bird pictures in the wild, setting up cameras near bird feeders. Although the latter method might produce more reliable data, the formal method is significantly more accessible and can be used to gather data for birds that are not native to the coders' regions. Using CNN, both groups obtained highly accurate AI models. However, their engines are not too accessible to general birders for the engine is trained and stored on the device. Therefore I designed an AI model and connected it to an app that users can access easily.

In order to make an APP, I used flutter, Firebase, and TensorFlow [7]. All of my code was coded in Android Studio. To make the user interface and the different pages of my app, I used flutter. Flutter is a software development toolkit(SDK) developed by Google [13]. Inside the Flutter framework, everything is coded with Dart, a coding language that is similar to java(Amadeo, 2018) [1]. For the back-end part of the APP - accessing the pictures that need to be displayed on the APP, getting the name of the birds that are displayed on the result page, logging-in information, etc. - I used Firebase. Firebase is a database service system that provides data storage and a server for hosting(Lardinois, 2014) [2]. The AI code is inspired by the CNN used by the previous scientists; I used TensorFlow, which is a library that contains materials needed for making my AI model. One major difference between my APP and the AI models of previous scientists is that I am able to display the results of my AI model on my APP, which is more appealing to general users. My app also allows the user to accurately identify birds while not worrying about internet connection. I have ensured this feature in my app by downloading a pre-trained AI model. This way the user can both use a functional model and not worry about training the model on their devices, for the model is ready from the moment the user downloads the app.

To prove my result and test my prototype, I mainly tested my AI model. This is because the main user interface and the "APP" part can not really be tested, rather the APP either works or doesn't. To test my AI Model, I tested my model's accuracy when identifying birds. I first selected five random birds that are in my dataset and found five random images from the internet for each of the five birds. Then I uploaded the images to my app and recorded the result in my notebook. To improve my model, I experimented with the number of times my model retrained itself. In my code, this amount of time is determined by a variable called epochs. And by changing the value of epochs, the model will train itself accordingly. I later experienced different numbers for the value of epochs and selected the model with the best result. After selecting the best model, I cleaned the data in the training code. Cleaning the code means that I went through some of the

training images in my dataset and deleted the ones that either has a disruptive background that hinders the machine from recognizing the bird. For example, a picture might have the bird covered by a leaf or a shadow, this alteration will decrease the effectiveness of the AI model for the picture has the wrong color(from the shadow) and the incorrect shape(from the leaf). Inaccurate pictures(ie. the wrong bird is displayed) are also crucial to be removed or else the model will be reading through all the images while training for the wrong bird.

The rest of the paper is organized as follows: Section 2 gives the details of the challenges that I met during the experiment and designing the sample; Section 3 focuses on the details of my solutions and prototype corresponding to the challenges that I mentioned in Section 2; Section 4 presents the relevant details about the experiment I did, following by presenting the related work in Section 5. Finally, Section 6 gives the concluding remarks, as well as points out the future work of this project.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Creating the App using Flutter and Firebase

The first challenge I faced was creating the app using flutter and Firebase [5]. Making an app is challenging because the process requires the coder to be familiar with a lot of keywords. Flutter has a lot of specific keywords that are not often used in python or java, like widgets, scaffold, app bar, state, floating, etc. I coded with Java and python before; although dart, the coding language for Flutter, is similar to Java, switching to Dart and learning all the new keywords took a lot of practice and time. Connecting my app to Firebase is another issue, for I have never used Firebase before. Firebase is an important part of my project, serving as the back-end portion of the project. Oftentimes I had to search up online guides and resources to search for each keyword and what they do. I also had a problem accessing an emulator to test out my APP, for I have a Ryzen CPU and it at first did not support the android emulator provided by android studio. But that problem was solved by some BIOS updates.

2.2. Making an Accurate AI model using TensorFlow

The second challenge I faced was making an accurate AI model using TensorFlow [14]. I have created AI engines with python, namely python PANDAS [15]. However, python PANDAS does not work as well with android studio and flutter compared to TensorFlow. But creating an AI Model was not easy. Just like making the app, there are a lot of keywords like labels, subsets, etc. The AI model consists of many different sections, there is an initializing part where the images are reshaped for preparation, a training part, and a validation part. And I had to refer to online guides to creating AI Engines with TensorFlow. Other than making an AI Model, I had trouble making it accurate, for my first prototype was only 53 percent accurate on average. I started to test out different methods to improve my model. I experimented with different numbers of times that the code retrains itself using the dataset. Another thing that I tried was cleaning the dataset by eliminating any inaccurate pictures and adding pictures that directly describe the bird.

2.3. Finding a Large Amount of Accurate data

The last challenge I faced was finding a large amount of accurate data. The image in the dataset is a crucial part of making the AI model. The accuracy of the AI model significantly depends on how accurate the training images are. At first, I used a dataset online that contained around one

hundred and fifty pictures per bird, and there is a total of three hundred and twenty-five birds. However, there are way more birds on earth than three hundred and twenty-five, so I must find a way to bulk copy images from the internet for each bird. To solve this problem, I used a bulk image downloader that will find and download however many pictures I told it to download from Bing. With the bulk downloader, I did not need to worry about finding images for my data. However, I still needed to go through the pictures to make sure they are accurate and are qualified to be used for my model. Unfortunately, there is not a more efficient way to check the quality of the images than manually going over them.

3. SOLUTION

The Overview of AI_Birder is presented in Figure 1. The user first creates an account and arrives at the main page. On the main page, the user can browse through the birds that the other users of AI_Birder have found. The birds are presented in a list with their name and a picture of the bird. And if the user clicks on the picture or the name, the APP will open up a new page showing all the pictures of the birds that have been uploaded by the users. If the user wants to identify a bird, they can return to the main page and either take a picture(pressing the camera icon button in the bottom right) or upload an image from the device(clicking on the image icon button in the top right). By clicking on either of the two buttons, the user can either select an image or take a picture. After that, the user will arrive at the AI identification page, where the AI engine will display the name of three bird species that are closest to the image uploaded. After the engine returns three bird species, the user can select the correct bird name. The image will then be added to the list of pictures for that bird species, and if the bird does not have any pictures yet, the APP will create a new section with a list of images for the new bird. And if the bird is not in the database, the user can enter the correct name of the bird, and there will be a new bird species added to the database. The main components of the APP are the login/logout and create account page, the main page, and the AI identification page with the AI engine.

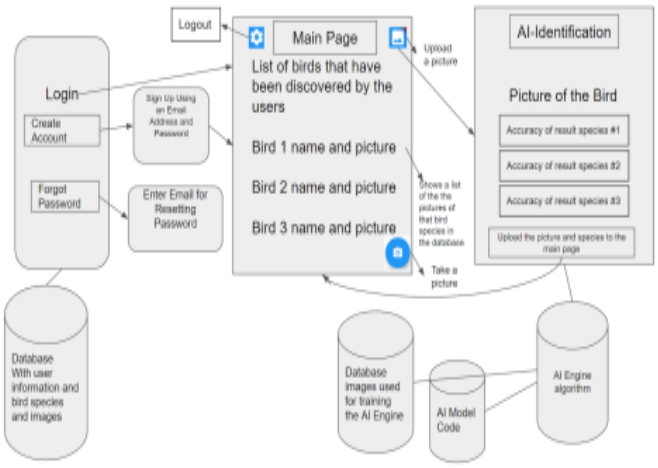

Figure 1. Overview of AI_Birder

To start off, Figure 2 shows the code that creates the Login page. First, I created a widget, or an object, called loginBody. This loginBody returns a container that contains text boxes, padding, buttons, etc. Padding are used to leave some space on the left and right sides of the app to make the page look aesthetically pleasing. CostomTextFields are also used for the user to input their email address and their password. I also implemented an elevated button at the end of the code. When the user clicks this button, the button takes the value of the email and the password to set

the user. The code also checks the email and the password beforehand to make sure that the user entered a valid email.

Figures 3 and 4 show the main page and the code that gets all the images and names from Firebase, respectively. In order for the code to get all the images and names, the images and names have to be stored somewhere. For AI_Birder, the names and images are stored in Firebase. Figure 5 shows the dataset that contains the images and names of each discovered bird. The "cover" photo of the bird species and its name is stored in the bottom right field, where there is an URL to the image and the name as a string. The rest of the photos are stored in the collection called "data " in the top right. Those photos are displayed when the user clicks on one of the bird species(a new page will open up with a list of the pictures of that species). The code is able to get the images from Firebase by returning a scaffold with a streambuilder [4]. The streambuilder builds an object based on the latest updated screenshot(state) of the stream, in this case, the stream of images. The streambuilder returns a listview of a column of images so that the user can see all the pictures of the bird. Another component of AI_Birder is the AI engine. The engine is made by implementing TensorFlow. The engine mainly trains itself by using images in a database. The engine uses the images in the training folder to train itself, then it uses the images in the validation folder to test its accuracy. The AI engine is modeled after MobileNetV2, which is a pre-trained CNN(see Figure 6). And from there, the model adds onto MobileNetV2 by recognizing patterns in bird pictures. This adding-on is called transfer learning, where, compared to traditional machine learning, the model creates a new neural network based on an existing one. This feature makes transfer learning more efficient and more effective than traditional machine learning types. After implementing MobileNetV2, multiple layers - Conv2D(which finds the patterns), Dropout(prevents overfitting), GlobalAveragePooling2D(returning the average output of information learned from previous layers), and Dense(receives all the information from previous layers) - train the AI engine. With all the training done, the code returns a label(all the birds' names) and a model, which is the trained AI engine. Those two things can be connected to the APP by uploading them to the code. After uploading, the APP is now functional with an AI engine.

```
Widget loginBody(){
  var targetIndex = 1;
  var forgotPwdIndex = 2;
  return Container(
    padding: EdgeInsets.symmetric(horizontal: 50),
    child: Column(
      children: [
        customTextField(
          controller: emailController,
          hintText: 'Email',
          inputType: InputType.email,
          ),
        customTextField(
          controller: passwordController,
          obscureText: true,
          autocorrect: false,
          hintText: 'Password',
          inputType: InputType.password,
          ),
        SizedBox(height: 20),
        customElevatedButton(
            onPressed: (){
              widget._fireAuth.signInWithEmailAndPassword(email: emailController.text, password: passwordController.text).then((user) => widget.setUser(user: user));
            },
            text: 'Log In',
```

Figure 2. Code for creating the Login page

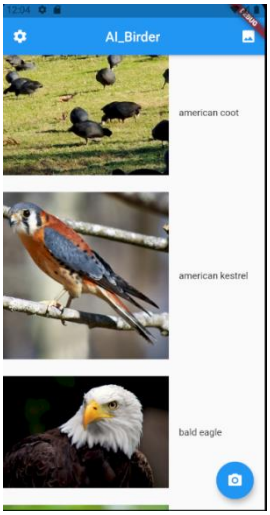

Figure 3. The main page of AI_Birder

```
Widget build(BuildContext context) {
  /* getting a snapshot of the species collection from our firebase database
   and passing it as a stream of the type query snapshot */
  final Stream<QuerySnapshot> _birdStream = FirebaseFirestore.instance.collection('species').doc(widget.birdType).collection('data').snapshots();

  return Scaffold(
    appBar: AppBar(
      centerTitle: true,
      title: Text(widget.birdType),
    ),  // AppBar
    // creates a screen from a stream of data - our specifically is the snapshot of the species collection
    body: StreamBuilder<QuerySnapshot>(
        stream: _birdStream,
        // snapshot - the current data within our collection
        builder: (context, snapshot) {
          if (snapshot.hasError) {
            return Text('Some error has occurred');
          }
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Text("Loading");
          }

          return ListView(
            padding: EdgeInsets.only(top: 10),
            children: snapshot.data!.docs.map((document) {
              var docData = document.data() as Map<String, dynamic>;
              var downloadUrl = docData['image-url'];
              return Column(
                children: [
                  Image.network(downloadUrl, width: 350,),
                  SizedBox(height: 10,)
                ],
              );  // Column
            }).toList(),

          );  // ListView
        }),  // StreamBuilder
  );  // Scaffold
```

Figure 4. Code for getting the names and the pictures of each bird species

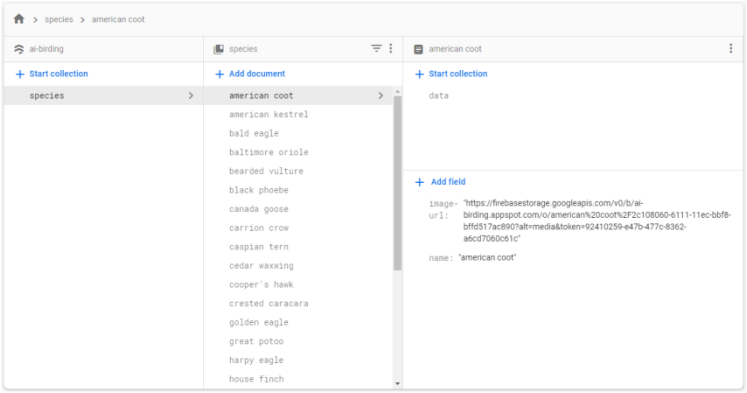

Figure 5. The Firebase database page

```
labels = '\n'.join(sorted(training_generator.class_indices.keys()))

# open the file 'labels.txt' and writing labels to it
with open('labels.txt', 'w') as f:
    f.write(labels)

# the size of our image (224, 224, 3)
img_shape = (img_dimension, img_dimension, 3)

# create a base model using MobileNetV2
base_model = tf.keras.applications.MobileNetV2(
    input_shape=img_shape,
    include_top=False  # used to prevent training previous layers and only adding on new ones
)

base_model.trainable = False
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Conv2D(32, 3, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.GlobalAveragePooling2D(),
    #change the number of folders
    tf.keras.layers.Dense(327, activation='softmax')
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Figure 6. Code for training and creating the labels and model for the AI Engine

4. EXPERIMENT

4.1. Experiment 1

To test my prototype, I mainly tested the accuracy of my AI engine, and I experimented with my prototype to find the best model. I experimented with a different amount of times that my code retrains itself using the dataset images, and I also experimented with a different number of training pictures for a species of bird.

The first experiment I did was testing the accuracy of my AI Model and experimenting with different amounts of times that my model retrains itself using the pictures in the dataset. That amount of time is determined by a variable called “Epochs”, so I changed that variable throughout the experiment. To test my AI model, I randomly selected five birds that are in my dataset and found 5 random pictures from the internet. I then uploaded the images to my app and recorded the result and took the average for each bird and for everything. Because my final average is the result of 25 random pictures, the average in the result reliably reflects the accuracy of my AI model.

Table 1 and Figure 7 illustrate the resulting percent of accuracy of all the models with different epoch values. By experimenting with different numbers of epochs, I discovered the model with Epochs = 11(the code will read through and train with all the pictures in the dataset 11 times) was the most successful. My data shows that the AI engine doesn’t become more accurate as it retrains itself more times. Because my AI engine was highly inaccurate - with a result of 52.2% average accuracy - when it retrained itself 19 times. This performance was due to the model overfitting, meaning that the model has trained itself too many times that it can only accurately identify the images in the training dataset. On the other hand, the model with only 9 epochs was also inaccurate because the engine did not train sufficiently. The model with epochs = 11 had an average percent of accuracy of 79 percent, proving itself to be the best model. I will be using the model with epochs = 11 for future experiments and tests since it yielded the best result.

Table 1. Average Percent of Accuracy for Models with Different Epochs value

Epochs vs accuracy(average)	Average Percent of Accuracy(%)
9	65.6
11	79
13	55.2
15	63.8
17	72.32
19	52.2

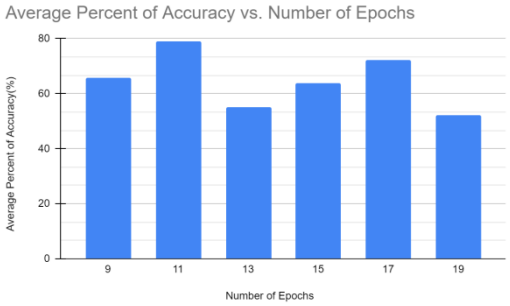

Figure 7. Average Percent of Accuracy for Models with Different Epochs value

Table 2. Data for Model with Epochs

Bird Species	Average Percent of accuracy
Red-tailed Hawk	83.8
Northern Shoveler	83.2
Mourning Dove	57
Black Phoebe	92
Bald Eagle	78.8
Average	79

4.2. Experiment 2

The next thing I tested was how my training images affected the accuracy of my models. Training and learning from the dataset's training pictures are the most important parts of making an AI engine, and having good training images is key to creating a successful AI engine. When testing my engine, I discovered that most of the inaccurate identifications resulted from pictures with birds flying, while the pictures with birds perching (staying still) yielded high accuracy. To improve my AI engine, I selected 5 random birds and found 5 random pictures of them flying. I then uploaded different numbers of pictures - 100, 200, 300- of the chosen birds flying to their respective database. Another set of pictures that my engine had trouble identifying was female and juvenile birds. Birds often have distinct differences in plumage between the male and female, and between adults and juveniles. To improve my AI engine, I uploaded different numbers of pictures - 50, 100, and 200(total) - of female birds to my training dataset and retrained my AI engine using the new images. I then recorded the percent of accuracy using online images of female birds. I also selected 5 random birds and 5 random images for each bird as the first

experiment. By adding female bird pictures, I hoped to make my AI engine more well-rounded so that the user can identify female birds as well as male birds.

Table 3 and Figure 8 display the increased accuracy of my AI engine after I added the images to the training dataset. The average accuracy increased from 60.7% to 86.6%. Therefore my experiment proves that I can improve my engine's effectiveness by adding more images of flying birds. Table 4 and Figure 9 display the resultant percentages of accuracy depending on different numbers of pictures of female birds. My data shows that by adding pictures of female birds to the training dataset, the AI engine can identify the birds more accurately, namely the female birds; almost doubling the accuracy from 32.4% to 64.5%. In conclusion, I can increase the effectiveness of my APP by adding more pictures of female and juvenile birds, because by doing that, the APP can not only identify the male birds but also their female and juvenile counterparts. Making sure the AI engine can identify the female and juvenile birds is crucial for there are countless female and juvenile birds in the world; not training the AI engine with appropriate images will cause the user to misidentify the bird, defeating the whole purpose of the APP.

Table 3. Average Percent of Accuracy vs. Number of Pictures of Flying Birds for Each Species

	Barn Owl	California Condor	Blue Heron	Common Grackle	Common Loon	Average
50 pictures	0.00%	28.0%	75.8%	28.2%	30.2%	60.7%
100 pictures	47.6%	67.0%	58.4%	63.4%	67.2%	64.5%
200 pictures	16.8%	78.4%	77.8%	63.0%	88.8%	86.6%

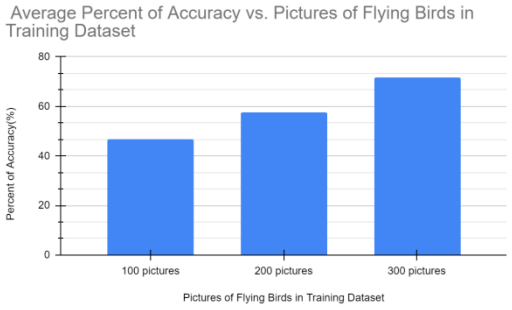

Figure 8. Average Percent of Accuracy vs. Pictures of Flying Birds in the Training Dataset

Table 4. Average Percent of Accuracy vs. Number of Pictures of Female Birds for Each Species

	Indigo Bunting	Mallard	Northern Cardinal	Northern Flicker	House Sparrow	Average
50 pictures	0.00%	28.0%	75.8%	28.2%	30.2%	32.4%
100 pictures	47.6%	67.0%	58.4%	63.4%	67.2%	60.2%
200 pictures	16.8%	78.4%	77.8%	63.0%	88.8%	64.5%

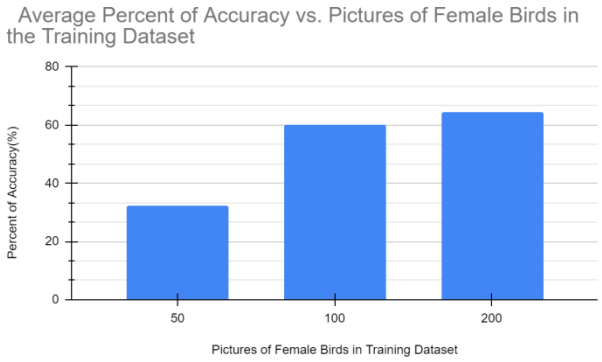

Figure 9. Average Percent of Accuracy vs. Pictures of Female Birds in the Training Dataset

After doing experiments and testing my AI engine. I found out that the model with epochs = 11 is the most accurate, with an average percent accuracy of 79.0%. Which is highly accurate for identifying random images. Using flutter, I was able to create a working APP that is now published on google play for android users and the APP store for IOS users. The AI-identification part of the APP is also completely functional even without the internet; this feature is achieved by implementing a pre-trained AI Model instead of on-device training. This method means that the users do not have to train their model when they first download the APP since the APP already has a model trained previously by me. This method not only circumvents internet requirements but also saves the user time and phone storage(the training database images take up a lot of space) from training the model themselves. My AI engine can also be easily improved by adding pictures of flying birds and female/juvenile birds. Although it may seem like a lot of pictures, my bulk-image downloader allows me to find hundreds and thousands of quality images within seconds. Therefore expanding and improving my database is simple and doable.

5. RELATED WORK

Steve Branson from Cal Tech, Grant Van Horn from UC San Diego, and his group categorized and identified bird species using Pose Normalized Deep Convolution Nets(Branson, Van Horn et al, 2014). Branson and his group identified birds using Pose categorization. Pose categorization works by first mapping out key features of the training image(an accurate picture of a species of bird) using nodes, basically mapping out where the beak, eyes, and crests are. And then when an image is uploaded, the algorithm marks the key features of the bird in the new images using nodes and compares them to the nodes of the training image to determine the bird species. Compared to Branson's method, AI Birder identifies birds by comparing the image and coloration of the uploaded image to the training image without assigning nodes. While pose categorization also produces accurate results, it is more complex compared to image and color recognition. And some images lack the normal features of the bird, for example, a picture with only the bird's back. In this case, color recognition would be more accurate and the safer option compared to pose categorization. Color and image recognition is also easier and more efficient to train for the developer doesn't have to assign nodes to the special features of each bird, instead, they just need to train the AI engine using accurate images of that bird.

Elias Sprengel and his group from ETH, Zürich, classified bird species using audio identification (Sprengel et al, n.d.). The group used convolutional neural networks for speech recognition to identify bird species. Unlike the work of Sprengel's group, AI_Birder only identifies birds using image and color recognition. Although identification using audio recognition poses challenges,

implementing it can be a great addition to AI_Birder. This is because often times the birder can easily hear birds concealed in bushes and leaves but fail to spot them to take a picture.

Andréia Marini, Jacques Facon, and Alessandro L. Koerich from the Pontifical Catholic University of Paraná identified birds based on color features(Marini, Facon, Koerich, 2013). The group's method is to eliminate the background of the bird picture to accurately identify the bird. Unlike Marini's group, AI_Birder takes both the bird and its background into consideration when identifying the bird species. Although getting rid of the background of the bird is a great idea and seems ideal, many bird species are very close in plumage and differ only in the type of environment they live in. Therefore, for some birds, considering the background while identifying the bird can help in securing higher accuracy.

6. CONCLUSIONS

In order to make birding more efficient and simple, I created AI_Birder, an APP that identifies birds without WiFi. Flutter was used to create the user interface and front-end part of the APP, and Firebase served as the back-end part of the APP and stores things like bird species and bird images. For the AI engine, I used Tensorflow, an AI library that contains invaluable resources and materials for creating AI models and algorithms. Using MobileNetV2, a convolutional neural network that is highly efficient for mobile applications, I created an AI engine that can identify birds without an internet connection. To test my AI engine, I selected five random birds from my database, found five random pictures for each bird, and uploaded the pictures to AI_Bird. I recorded all the results and the percent of accuracy for each bird. To improve my AI engine, I experimented with my code's "epoch" value, which determines how many times my AI engine retrains itself using all the pictures in the training database. After experimenting with different epoch values, I selected the most accurate model with the epoch value equal to 11, which has an average accuracy of 79 percent. Furthermore, I improved my AI engine by improving the accuracy of identifying female, juvenile, and flying birds(birds in motion). By adding more pictures of female, juvenile, and flying birds, my engine's accuracy increased significantly from an average of 60.7% to 86.6% for flying birds, and from 32.4% to 62.5% for female/juvenile birds. However, for some bird species that look very similar species, AI_Birder often returns lower accuracy. But this issue can be easily solved by adding more pictures of that species and also cleaning and checking the current pictures in the database. For example, juvenile bald eagles look very similar to golden eagles. After I deleted incorrect pictures of both species in the database, the accuracy of AI_Birder identifying bald eagles increased from an average of 43.2% to 78.8%. Other than improving on the current birds, I also need to constantly update AI_Birder for additional birds that users recently discovers. Another limitation to AI_Birder is that because birds are all over the world, and some species only live in some parts of the world. But because some birds look similar but live in different places in the world, AI_Birder might display some birds living outside the user's range. So implementing a system where the APP has a mode that displays birds that are only near the user's location will be beneficial to improving the user's experience. Audio-based identification is also a useful addition to the app so that birders can identify birds based on the calls they hear. Finally, the addition of displaying fun facts/educational messages for each bird on the main page is also a great addition. However, the main purpose of AI_Birder is to give the user the correct bird quickly. After, the user can use other online information and sources to research the bird further. Nonetheless, showing fun facts is still helpful for the user.

REFERENCES

[1] Al-Showarah, Suleyman A., and Sohyb T. Al-qbailat. "Birds Identification System using Deep Learning." International Journal of Advanced Computer Science and Applications 12.4 (2021).

[2] Wu, Wenhao. "React Native vs Flutter, Cross-platforms mobile application frameworks." (2018).

[3] Pough, Richard Hooper. Audubon Bird Guide: Small Land Birds of Eastern & Central North America from Southern Texas to Central Greenland. Doubleday, 1949.

[4] Ferreira, André C., et al. "Deep learning-based methods for individual recognition in small birds." Methods in Ecology and Evolution 11.9 (2020): 1072-1085.

[5] Alsalemi, Abdullah, et al. "Real-time communication network using firebase cloud IoT platform for ECMO simulation." 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2017.

[6] Lasseck, Mario. "Audio-based Bird Species Identification with Deep Convolutional Neural Networks." CLEF (working notes) 2125 (2018).

[7] Lee, Juhyun, et al. "On-device neural net inference with mobile gpus." arXiv preprint arXiv:1907.01989 (2019).

[8] Prokop, Pavol, and Rastislav Rodák. "Ability of Slovakian pupils to identify birds." Eurasia Journal of Mathematics, Science and Technology Education 5.2 (2009): 127-133.

[9] van Lent, Michael, and John Laird. "Developing an artificial intelligence engine." Proceedings of the game developers Conference. 1999.

[10] Van Der Maaten, Laurens, Eric Postma, and Jaap Van den Herik. "Dimensionality reduction: a comparative." J Mach Learn Res 10.66-71 (2009): 13.

[11] Abdi, Hervé, and Lynne J. Williams. "Principal component analysis." Wiley interdisciplinary reviews: computational statistics 2.4 (2010): 433-459.

[12] Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." Chemometrics and intelligent laboratory systems 2.1-3 (1987): 37-52.

[13] Vanfretti, Luigi, et al. "A software development toolkit for real-time synchrophasor applications." 2013 IEEE Grenoble Conference. IEEE, 2013.

[14] Birnbaum, Lawrence, Margot Flowers, and Rod McGuire. "Towards an AI model of argumentation." Proceedings of the First AAAI Conference on Artificial Intelligence. 1980.

[15] McKinney, Wes. "pandas: a foundational Python library for data analysis and statistics." Python for high performance and scientific computing 14.9 (2011): 1-9.