# Project 3
# Image Classifier: Warblers


- Danielle Dejean
- Baruch Gottesman
- Kaidon Kennedy
- Carolyn Scheese
- Harpreet (Monty) Singh



# Project Overview & Goals

- Use image classifier to identify different birds.
- Create an interactive app which users can use to predict the species and include a “fun fact” about that bird.


# Data Sets

- Basic Data Set – Kaggle
- Winged Wonders: The Birds-200. Kaggle. https://www.vision.caltech.edu/datasets/cub_200_2011/
  - 200 Species of birds in North America. Approx 60 images per species. 1100 + files.
  - Images of each species of bird was in its own folder.
  - Selected 25 files of Warblers
- Fun Facts
  - Scraped the internet – Wikipedia and Bing

# Data Collection, Clean Up &  Exploration


Eds., Birds of Europe, Russia, China and Japan (Princeton 2007)

Michele, Jules, “The Bird” (Oxford 1876)

Cuvier, Baron, “Animal Kingdom”  (Edinburgh 1835)

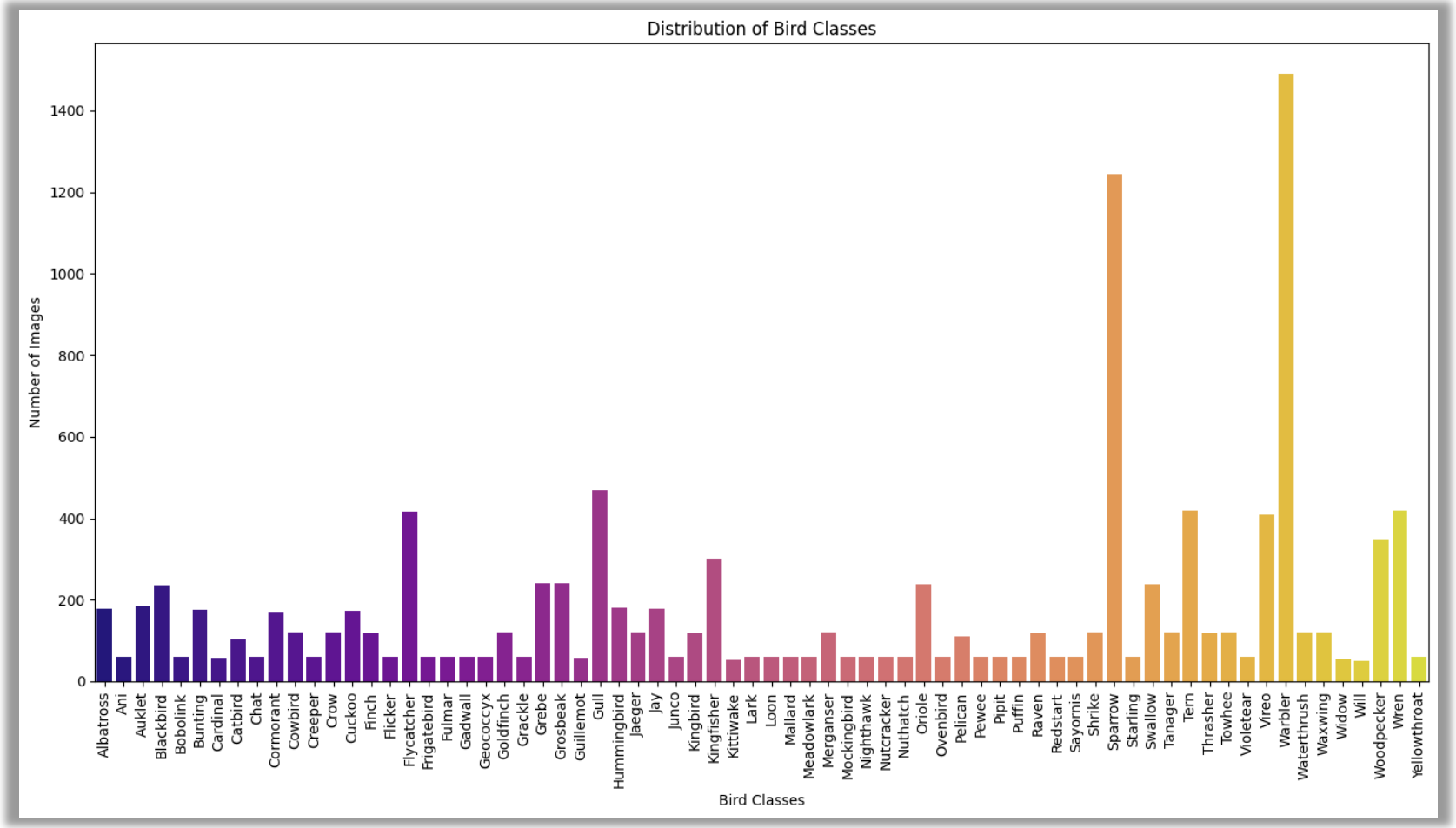
Blanchan, Neltje, Bird Neihghbors (Doubleday 1904)

Audobon Print (1886)

For convenience, we reproduce the parties' respective marks below:

# Data Collection, Clean Up & Exploration

*Abercrombie & Fitch Trading Co. v. Gubbala*, Opp. Proceeding 91255288 (TTAB 2023)

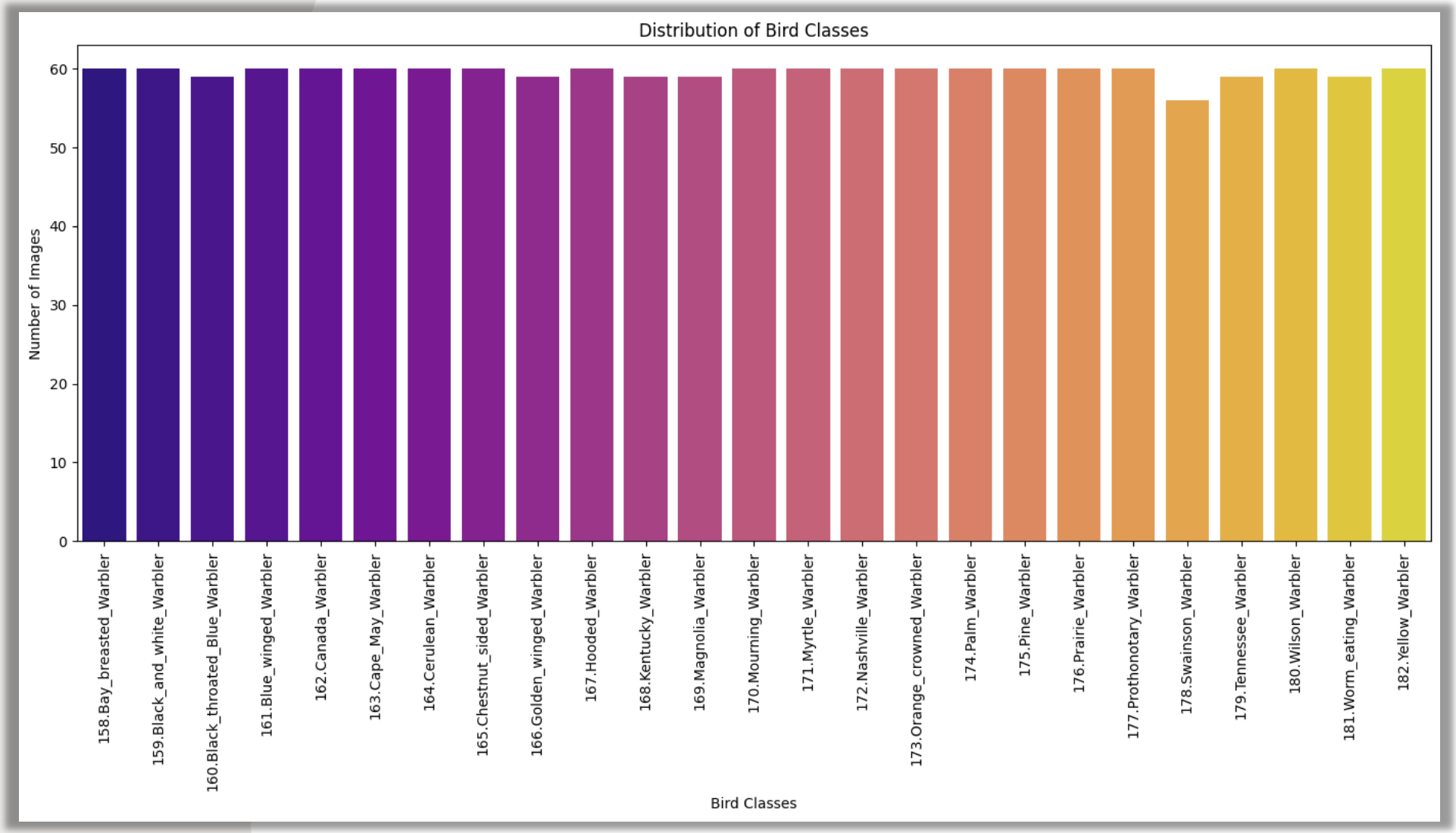


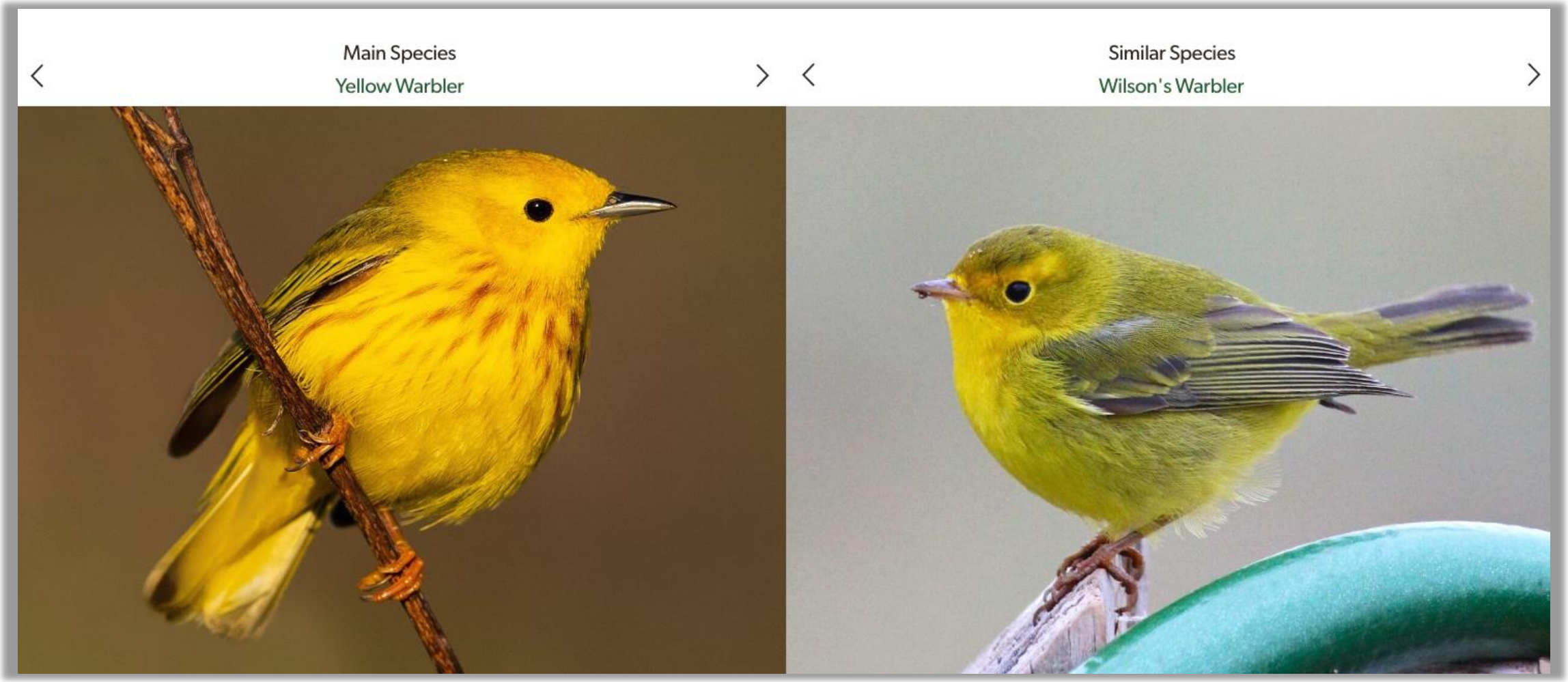

# Data Collection, Clean Up & Exploration

- * 11,788 Total Source Pictures
- * Unevenly distributed among types of birds

# Data Collection, Clean Up & Exploration

- 1,000+ Pictures
- Very similar birds, make them poor candidates for “classification”






# Data Collection, Clean Up &  Exploration

# Approach

Played with source data so we had one folder with all “Warbler” pictures designated image_##

Created CSV that contained the bird identification associated with each picture

Work within Google Colab/Google Drive to take advantage of virtual machines (T-4 High Ram)

# Approach

* Normalized to 224 x 224

```
[ ]  # Get all the sizes into a list, then convert to a set
     sizes = set([img.size for img in images])
     sizes
     # Use a for loop to resize all images to 224
     target_size = (224, 224)

     resized_images = [img.resize(target_size, resample = Image.LANCZOS) for img in images]
     resized_images[4]
```

* Augmented dataset with 5 transformed pictures

```
[ ]  # Apply augmentation to the whole training dataset
     # Define the augmentation pipeline

     data_augmentation = tf.keras.Sequential([
         tf.keras.layers.RandomRotation(0.1),          # Rotate by 10%
         tf.keras.layers.RandomTranslation(0.15, 0.15), # Shift horizontally and vertically by 15%
         tf.keras.layers.RandomZoom(0.1),              # Zoom in/out by 10%
         tf.keras.layers.RandomFlip('horizontal')      # Random horizontal flip
     ])

     # Create variables to hold the X and y training data
     X_train_aug = []
     y_train_aug = []
```

* Played with “patience” of non-improvement

```
[ ]  lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
         initial_learning_rate=1e-4,  # Start with a smaller learning rate
         decay_steps=10000,
         decay_rate=0.95,
         staircase=True
     )
     # Define the EarlyStopping callback
     early_stopping = EarlyStopping(
         monitor='val_loss',          # Monitors validation loss
         patience=20,                  # Stops if no improvement in 20 epochs
         restore_best_weights=True    # Restores best weights if stopped early
     )
     from tensorflow.keras.callbacks import ReduceLROnPlateau
     reduce_lr = ReduceLROnPlateau(
         monitor='val_loss',
         factor=0.5,
         patience=2,  # Reduce LR if no improvement for 2 epochs
         min_lr=1e-6
     )
```

* Played with number of layers

```
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    Dropout(0.3),

    layers.Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    Dropout(0.4),

    layers.Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    Dropout(0.4),

    # New additional layer block
    layers.Conv2D(256, (3, 3), activation='relu'),
    BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    Dropout(0.4),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    layers.Dense(25, activation='softmax')
])
```

# Side Quest: Develop “Fun Fact” Feature

Goal: * Develop Agent that can Collate “Fun Facts” and other supplemental info

* Ostensible Deliverable: More Complete Information about Identified Species

* Actual Deliverable: Practice work with Agents, APIs and LLMs

# Side Quest: Develop “Fun Fact” Feature

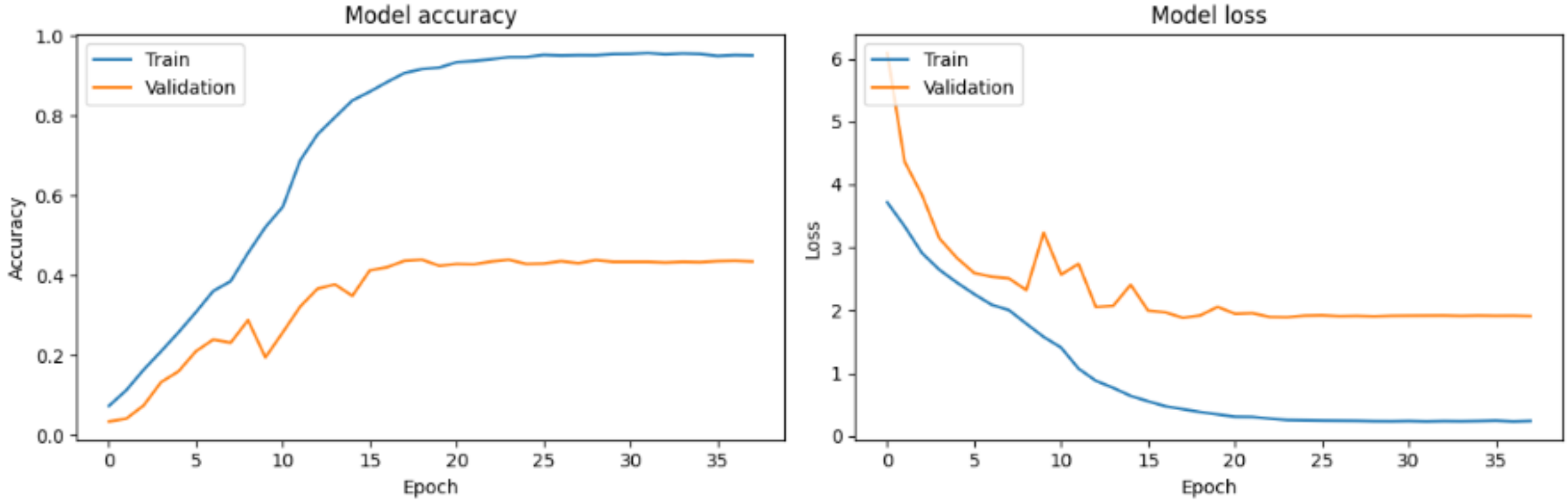
**Approach:**

*** Agent identifies Wikipedia Pages**

*** Agent obtains all info from Wiki**
***(Wikipedia User Agent)***

*** Agent searches for the species on Bing.com and returns four results**
***(Microsoft Azure Bing API Client)***

*** Send all text to Groq and ask it to develop five “Fun Facts”**
***(Groq API and have LLM extract data)***




# Analysis

- Our model uses layers to extract features from images, applies pooling to focus on important details, and includes batch normalization and dropout to make learning stable and avoid overfitting. It ends with dense layers to classify the images into categories.
- Our convolutional neural network capped off with a 53% validation accuracy.

# Analysis

- We chalk this up to a few of the birds having a lot of similar features and colors making it hard for the model to differentiate a few species. In turn dropping the overall accuracy of the model.

# Results & Conclusions

In conclusion we webscraped 3-5 fun facts for all 25 species of warbler.

We created a model with close to 60% overall accuracy in which you can identify 25 different species of Warbler.

Some warblers are identified more accurately.


Cape May Warbler

Chestnut Sided Warbler

# Steps:
# Developing an App

# Gradio Image Classifier for 25 Warbler Species

- Import libraries
- Load trained Keras model
- Read in “Fun Facts”
- Define a function to process the image & make predictions
- Create prediction
- Map predicted class to the species
- Create Gadio interface
- Launch the app

# Gradio Code

Import Libraries

```
#Import libraries
import gradio as gr
import numpy as np
import pandas as pd
from PIL import Image
from keras.models import load_model
from keras.preprocessing import image
```

Import Keras Model

```
# Load the trained Keras model
model = load_model('warbler_model.keras', compile=False)
```

Read in Fun Facts/ Define Function

Make Predictions & Map to Species

```
# read in fun facts into a df
fun_df = pd.read_csv("Fun Facts about Warblers.csv", usecols=["Species Name", "Fun Fact 3"])

# Define a function to process the image and make predictions
def predict_bird_species(img):
    # Preprocess the image for the model
    img = img.resize((224, 224))  # Resize to the input size of the model
    img_array = image.img_to_array(img)  # Convert image to array
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    img_array /= 255.0  # Normalize the image

    # Make predictions
    predictions = model.predict(img_array)
    predicted_class = np.argmax(predictions, axis=1)  # Get the class with the highest probability

    # Map the predicted class to the species name
    species_names = ["Bay_breasted_Warbler", "Black_and_white_Warbler", "Black_throated_Blue_Warbler", "Blue_winged_Warbler", "Canada_Warbler", \
                     "Cape_May_Warbler", "Cerulean_Warbler", "Chestnut_sided_Warbler", "Golden_winged_Warbler", "Hooded_Warbler", "Kentucky_Warbler", \
                     "Magnolia_Warbler", "Mourning_Warbler", "Myrtle_Warbler", "Nashville_Warbler", "Orange_crowned_Warbler", "Palm_Warbler", "Pine_Warbler
                     "Prairie_Warbler", "Prothonotary_Warbler", "Swainson_Warbler", "Tennessee_Warbler", "Wilson_Warbler", "Worm_eating_Warbler", \
                     "Yellow_Warbler"]
    fun_fact = fun_df.loc[fun_df["Species Name"] == species_names[predicted_class[0]], "Fun Fact 3"].values[0][3:]
    bird_image_path = f"single_image_warblers/{species_names[predicted_class[0]]}.jpg"
    bird_image = Image.open(bird_image_path)
    return species_names[predicted_class[0]], bird_image, fun_fact  # Return species name and the original image
```

Create Gradio Interface

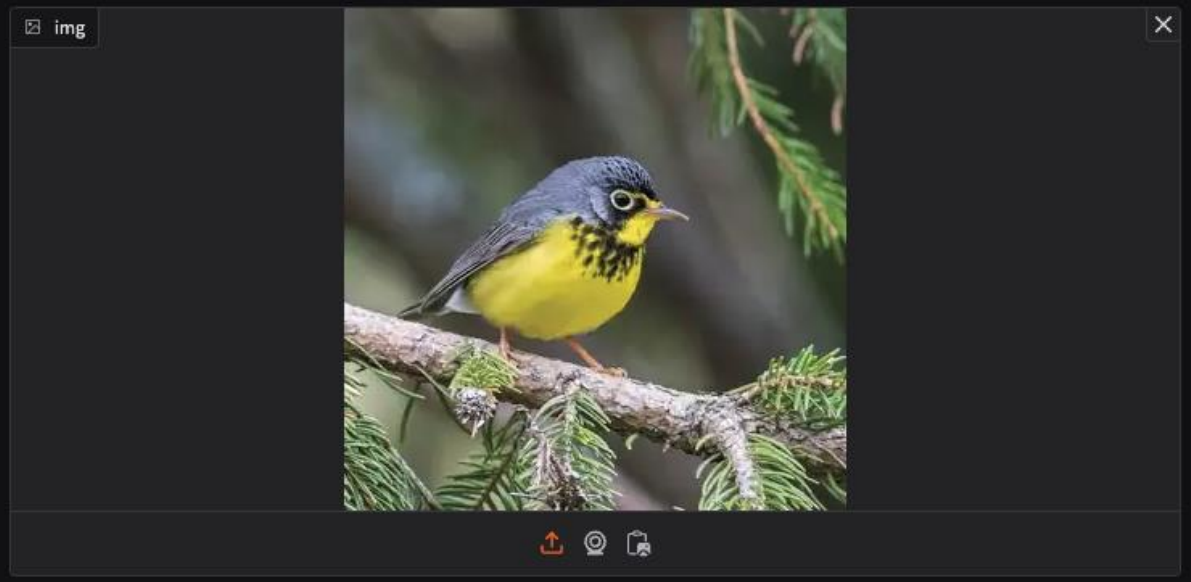
```
# Create the Gradio interface
iface = gr.Interface(
    fn=predict_bird_species,
    inputs=gr.Image(type="pil"),  # Input component for image upload
    outputs=[gr.Label(), gr.Image(type="pil"), gr.Label()],  # Output for species name and the image
    title="Bird Species Prediction",
    description="Upload a picture of a bird to predict its species."
)
```

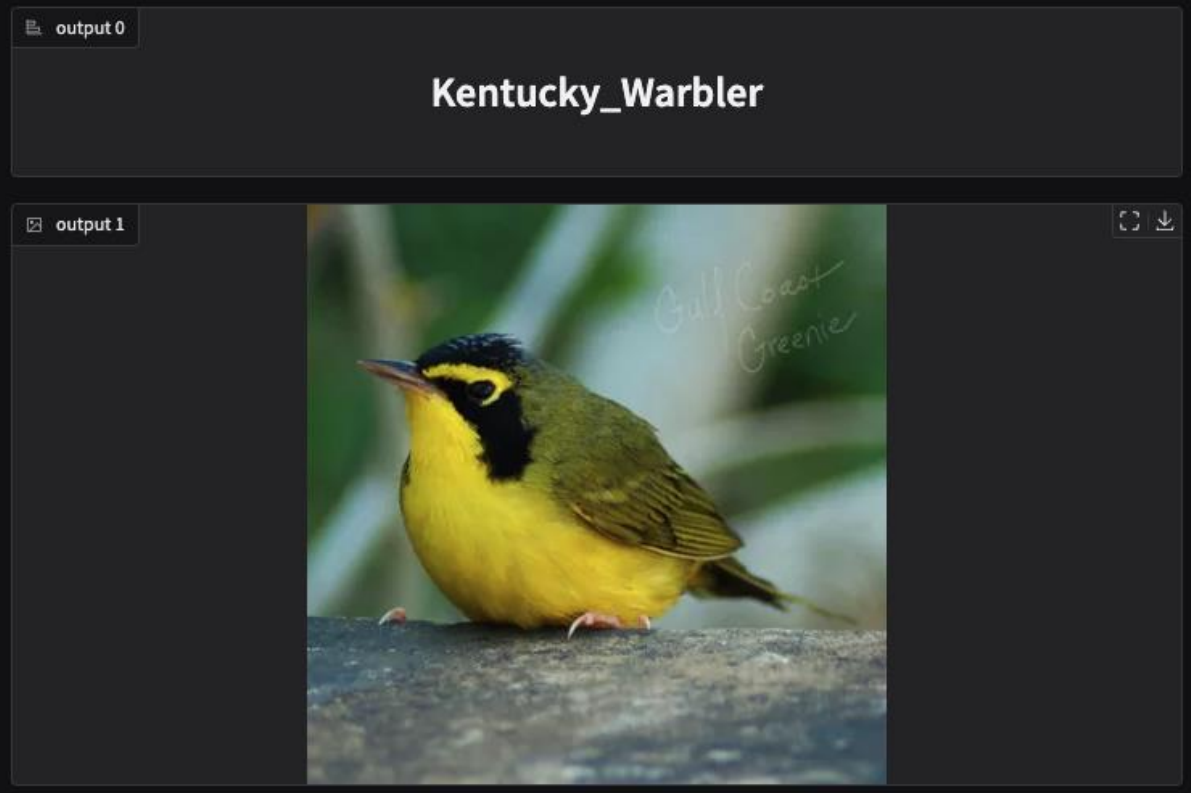
Launch the App

```
# Launch the app
iface.launch(share=True)
```

# Bird Species Prediction

Upload a picture of a bird to predict its species.




Gradio demo

# Steps:
# Developing an App

# Streamlit Image Classifier for 25 Warbler Species

- Import libraries
- Load trained Keras model
- Read in “Fun Facts”, create fn to filter out empty/NAN rows
- Configure page layout
- Map predictions
- Map predicted class to the species
- Display results in Streamlit
- Call fun fact summary NLP function

# Streamlit app – Main.py

## Text Cleanup

```python
# Function to clean up the introductory text and placeholder text in the DataFrame
def clean_facts(df):
    for col in [f'Fun Fact {i}' for i in range(1, 6)]:
        # Replace introductory text and placeholder with "No fact available"
        df[col] = df[col].apply(lambda x: "No fact available" if pd.isna(x) or
                                (isinstance(x, str) and (x.startswith(intro_prefix) or placeholder_text in x)) else x)
    return df

# Apply the cleaning function to the DataFrame
warbler_facts_df = clean_facts(warbler_facts_df)

# Function to get all fun facts for a species from the DataFrame
def get_fun_facts(species_name):
    fact_row = warbler_facts_df[warbler_facts_df['Species Name'] == species_name]
    if not fact_row.empty:
        facts = [fact_row[f'Fun Fact {i}'].values[0] for i in range(1, 6)]  # Get facts from columns Fun Fact 1 to Fun Fact 5
        return facts
    else:
        return ["No facts available for this warbler."]
```

## Filter Empty/NANs

```python
    # Display the fun facts
    fun_facts = get_fun_facts(cleaned_species_name)

    # Filter out invalid facts and handle NaNs
    valid_facts = [
        fact.split('. ', 1)[1] if isinstance(fact, str) and '. ' in fact else fact
        for fact in fun_facts
        if isinstance(fact, str) and fact != "No fact available"
    ]

    # Display the valid fun facts or a message if none are available
    if valid_facts:
        for i, fact in enumerate(valid_facts, start=1):
            st.markdown(f"### 🌟 Fun Fact {i}: {fact}")
    else:
        st.markdown("### 🌟 No fun facts available for this warbler.")
```

## Configure Page Layout

```python
# Page configuration
st.set_page_config(page_title="Warbler Classifier", page_icon="🔎", layout="centered")

# Title and prompt for users
st.title('Warbler Classification Application')
st.markdown("### 📸 Please upload a warbler image for classification.")
st.divider()

# Sidebar with information
st.sidebar.title("About This App")
st.sidebar.write("This app classifies warbler images and provides interesting facts about them.")
st.sidebar.markdown("### Instructions")
st.sidebar.write("1. Upload an image of a warbler.\n2. View the predicted species and learn some information about them!")
```

## Call Fun Fact summary fn

```python
    # Display the fun facts summary
    with st.spinner("Summarizing fun facts..."):
        summary = summarize_facts(cleaned_species_name, warbler_facts_df)
    st.markdown(f"### 🌟 Summary of Fun Facts:")
    st.write(summary)
```

# Streamlit app – summary_pipeline.py

## Create pipeline to pre-trained model

```
8  # Pretrained summarization pipeline
9  summarizer = pipeline("summarization", model="facebook/bart-large-cnn")
```

## Fix incomplete summaries

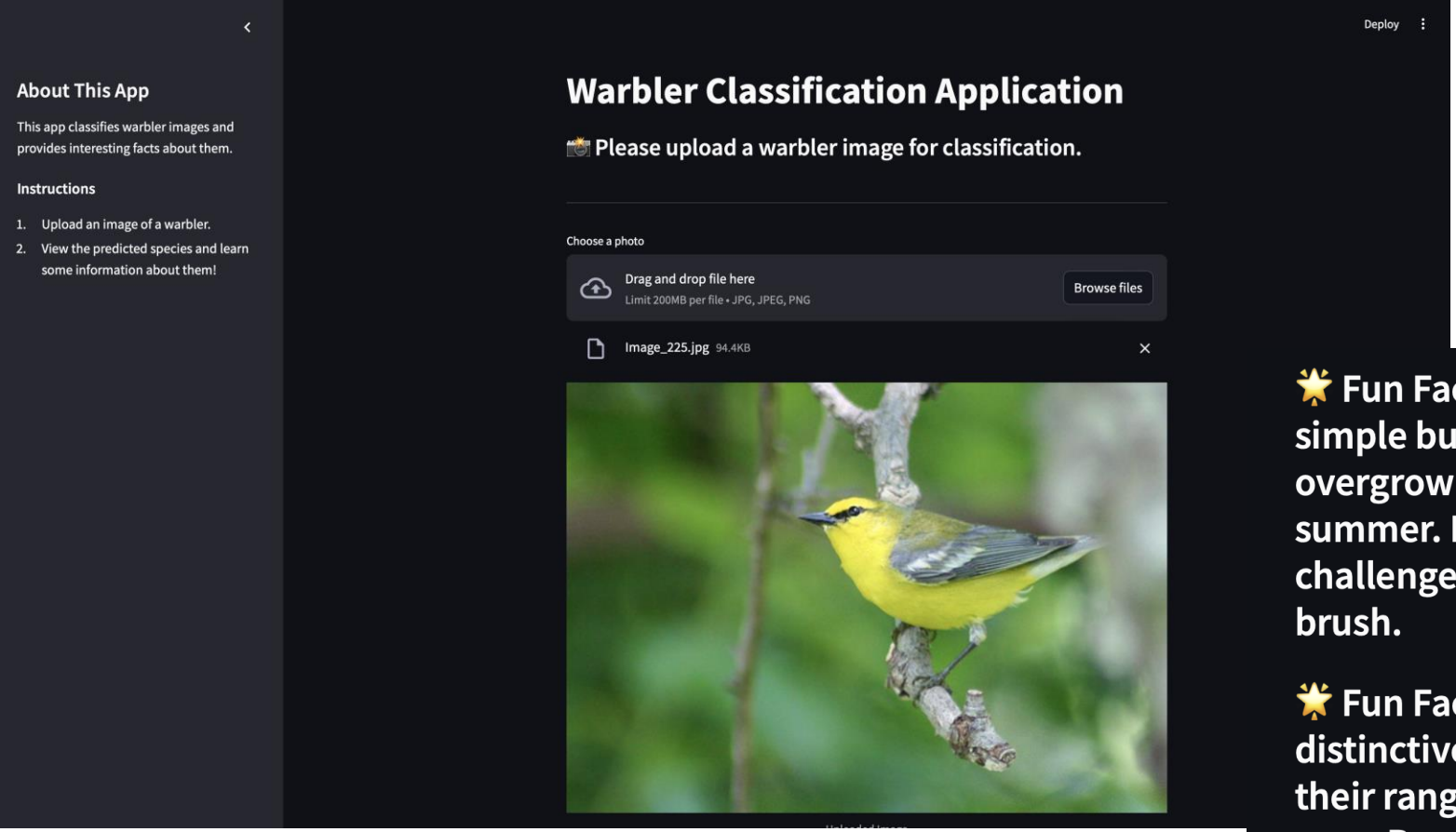
```
32 # Function to fix incomplete summaries
33 def fix_incomplete_summary(summary):
34     """
35     Checks for common incomplete sentence patterns and fixes or annotates them.
36     """
37     if summary.endswith(('in the', 'and', 'such as', 'with')):
38         return summary + " Additional context may be required."
39     return summary
```

## Create fn to summarize fun facts

```
60 # Summarize fun facts for a given species
61 def summarize_facts(species_name, warbler_facts_df):
62     text = prepare_text_for_summary(species_name, warbler_facts_df)
63     if text == "No fun facts available to summarize.":
64         return text
65
66     # Perform summarization
67     summary = summarizer(text, max_length=70, min_length=30, do_sample=True, temperature=0.7)
68
69     # Apply post-processing to the summary
70     cleaned_summary = fix_incomplete_summary(summary[0]['summary_text'])
71     refined_summary = refine_summary(cleaned_summary, warbler_facts_df, species_name)
72     return refined_summary
```

## Focus summary on ecology/behavior

```
41 # Function to refine the summary for key ecological or behavioral aspects
42 def refine_summary(summary, facts_df, species_name):
43     # Retrieve key ecological and behavioral facts from the dataset
44     fact_row = facts_df[facts_df['Species Name'] == species_name]
45     if not fact_row.empty:
46         key_facts = [
47             fact_row[f'Fun Fact {i}'].values[0]
48             for i in range(1, 6)
49             if "habitat" in fact_row[f'Fun Fact {i}'].values[0].lower()
50         ]
51     else:
52         key_facts = []
53
54     # If no ecological/behavioral facts are in the summary, append one
55     if key_facts and not any("habitat" in summary.lower() for fact in key_facts
56         summary += f" Additionally, {key_facts[0]}"  # Add the first relevant f
57
58     return summary
```




Streamlit demo

🌟 **Fun Fact 4: Blue-winged warblers are known for their simple buzzy song, which is often heard in brushy overgrown fields and thickets in the East during the summer. Despite not being especially shy, they can be a challenge to observe due to their active foraging in dense brush.**

🌟 **Fun Fact 5: The blue-winged warbler forms two distinctive hybrids with the golden-winged warbler where their ranges overlap in the Great Lakes and New England area: Brewster's warbler and Lawrence's warbler. These hybrids exhibit unique plumage patterns that combine characteristics of both parent species.**

### 🌟 Summary of Fun Facts:

The blue-winged warbler's scientific name was changed in 2010 to correct an error made by Carl Linnaeus in the 18th century. They have a sharply pointed bill that serves as an effective tool for gleaning leaves and buds as they hunt for small insects. Additionally, 2. Blue-winged warblers have benefited from landscape changes over the last 150 years as forest clearcuts and agricultural developments have created more scrubby and cut-over habitats, which they prefer.

# Challenges Encountered

**Computational Power:**

- The model required more epochs, layers, or larger images to achieve higher accuracy, which demanded significant computational power.

**Google Colab Limitations:**

- Ran into time limits with Google Colab's GPU, leading to interruptions in training.

**Streamlit Application Challenges:**

- Ensuring classes were ordered correctly to prevent incorrect predictions due to indexing errors.
- Importing and displaying fun facts accurately for the corresponding species.

**Large Dataset Processing:**

- Managing a dataset of 11,788 images was time-intensive at each stage (e.g., uploading to Google Drive, processing in Colab), even with a virtual T4 and high RAM, which added delays throughout the workflow.

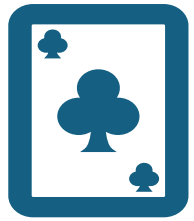
# If We Had More Time

## Improving Model Accuracy:

Increase bird types and iterations for better results; consider identifying high-level categories (e.g., “warbler” vs. “duck”) to simplify classification.

Experiment with more image sizes, additional layers, and pretrained models (like VGG16 and EfficientNet B0).

Address overfitting issues for more robust performance.

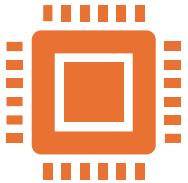


## Data and Fun Facts:

Correct fun facts formatting, expand fun facts, and ensure they match each species.

Retrieve endangered status information via the IUCN or Birdlife API when available.

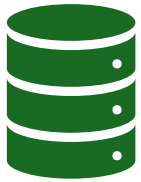
# If We Had More Time(cont.)



## Technical Resources:

Use more GPUs to speed up training in Google Colab.

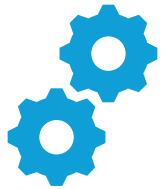
Run models on higher-tier virtual machines for better performance.



## Dataset Management:

Use optimized data pipelines to handle large datasets more efficiently.

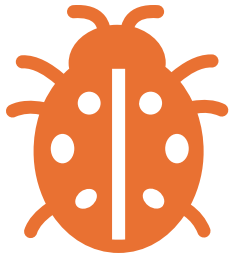
Explore data augmentation techniques to expand the dataset without needing additional images.



## Model Optimization:

Experiment with reducing model complexity to improve efficiency without sacrificing accuracy.

# If We Had More Time(cont.)



## Build out an Infobot:

We would look into adding an infobot to the Streamlit trained on more warbler data and general bird data for an option to learn more.

## Enhancements:

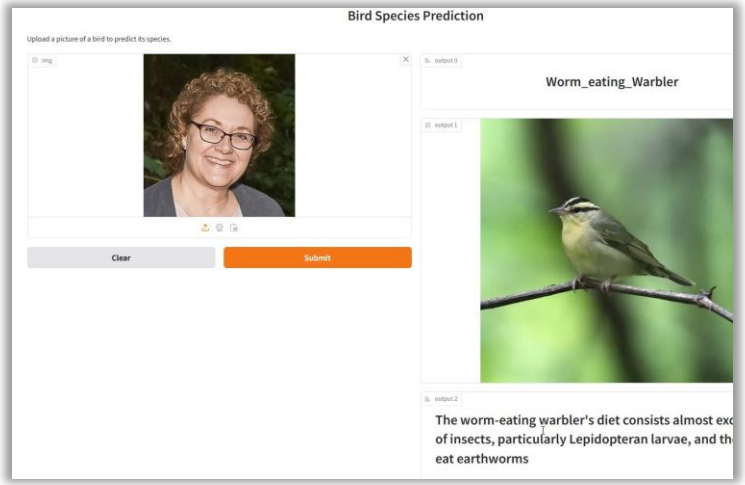
Include text to speech as part of the app

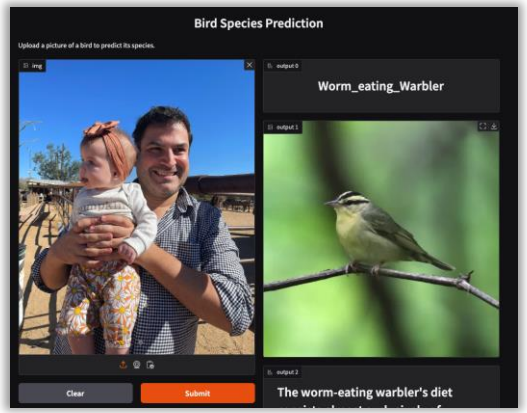
Include sound files for each species

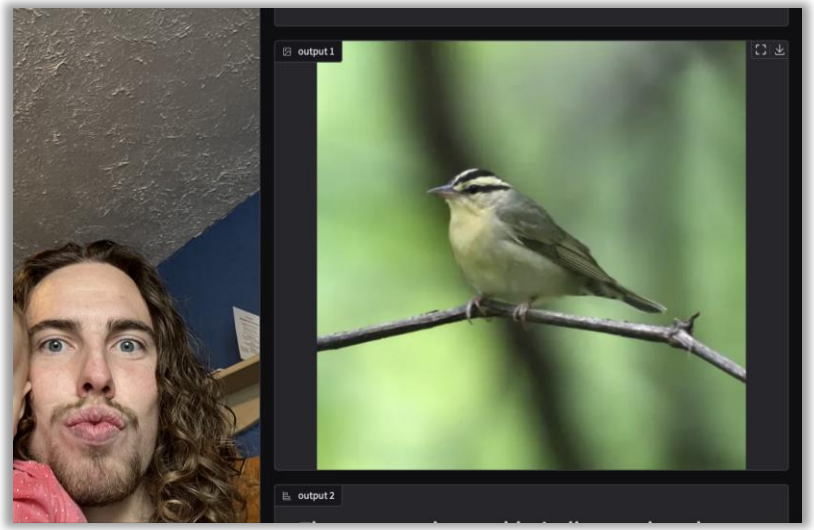
Include a map to show where the species live

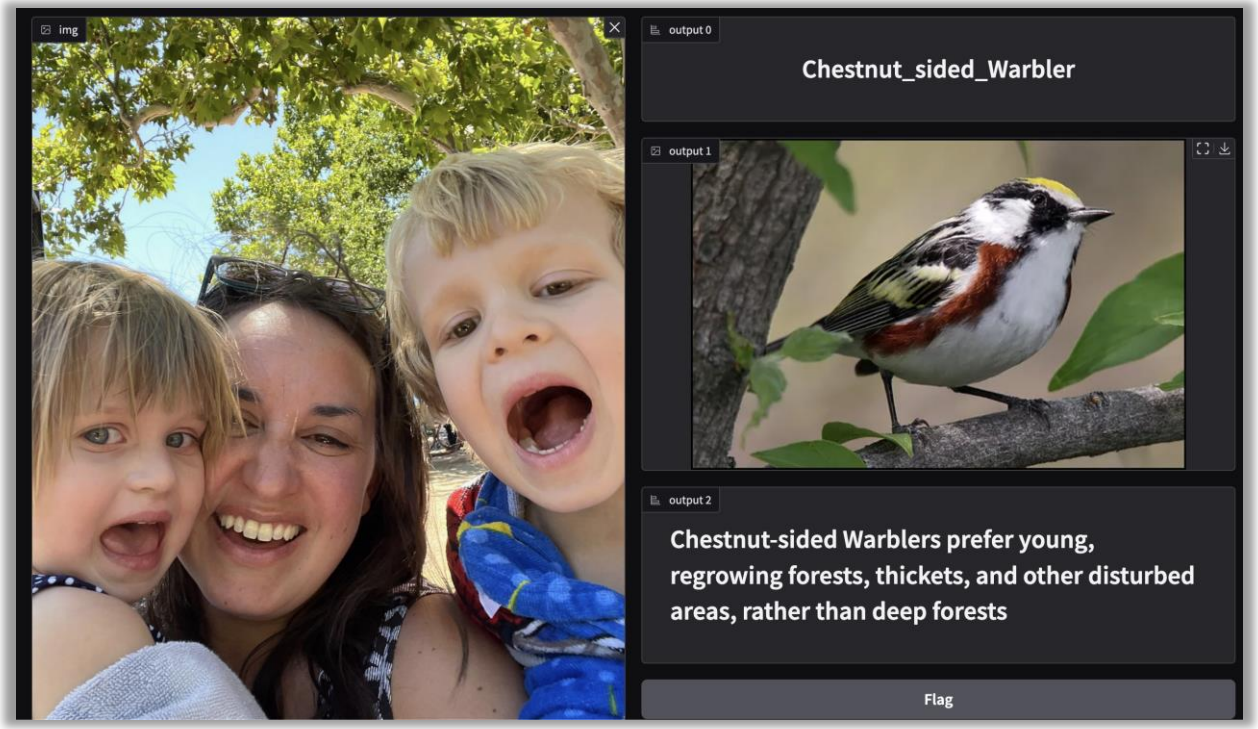
# References

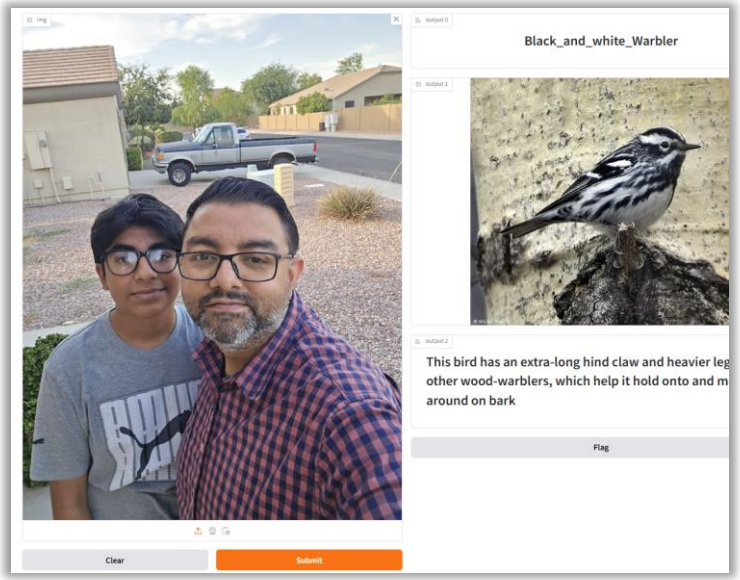

- All About Birds-Comparing Similar Species- Link
- Bing Fun Facts Search- Link
- Images. ChatGTP- Image Generator.
- North American Warblers- Link
- Winged Wonders: The Birds-200. Kaggle. Data Source- Link
- Wikipedia API- Link


Bird Species Prediction
Upload a picture of a bird to predict its species.
Worm_eating_Warbler
Clear
Submit
The worm-eating warbler's diet consists almost exc
of insects, particularly Lepidopteran larvae, and th
eat earthworms


Bird Species Prediction
Upload a picture of a bird to predict its species.
Worm_eating_Warbler
Clear
Submit
The worm-eating warbler's diet


output 1
output 2


img
output 0
Chestnut_sided_Warbler
output 1
output 2
Chestnut-sided Warblers prefer young, regrowing forests, thickets, and other disturbed areas, rather than deep forests
Flag


Black_and_white_Warbler
This bird has an extra-long hind claw and heavier leg
other wood-warblers, which help it hold onto and m
around on bark
Clear
Submit
Flag


# Questions?