PRO-C126

Live Image Prediction

What is our GOAL for this MODULE?

The goal of this module is to create a React Native app and make a POST request on the API.

What did we ACHIEVE in the class TODAY?

- We created a React Native app and made a POST request on the API.
- We tested the prediction model to identify the digit in the image.

Which CONCEPTS/CODING BLOCKS did we cover today?

- Classifier
- React Native app
- Ngrok to host the local servers online

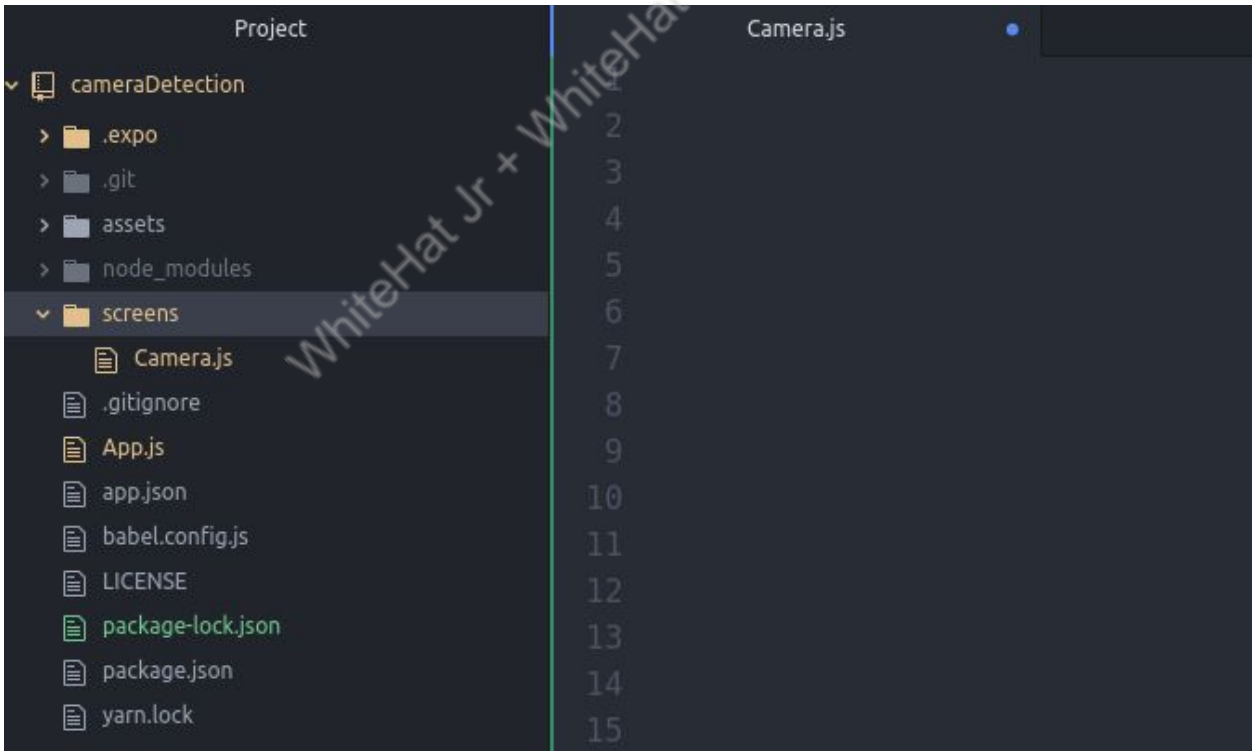
How did we DO the activities?

1. We started by creating the React Native app which used camera roll to pick up the image from the gallery.

```
ashura@zeros:~$ expo init imageDetection

   There is a new version of expo-cli available (3.27.4).
   You are currently using expo-cli 3.21.5
   Install expo-cli globally using the package manager of your choice;
   for example: `npm install -g expo-cli` to get the latest version

? Choose a template: (Use arrow keys)
  ----- Managed workflow -----
❯ blank                  a minimal app as clean as an empty canvas
  blank (TypeScript)     same as blank but with TypeScript configuration
  tabs                   several example screens and tabs using react-navigation
  ----- Bare workflow -----
  minimal                bare and minimal, just the essentials to get you started
  minimal (TypeScript)   same as minimal but with TypeScript configuration
```



- Installed the packages

```
$ expo install expo-permissions
```

```
$ expo install expo-image-picker
```

Camera.js App.js

```
import * as React from "react";
import { Button, Image, View, Platform } from "react-native";
import * as ImagePicker from "expo-image-picker";
import * as Permissions from "expo-permissions";

export default class PickImage extends React.Component {
  state = {
    image: null,
  };
```

Camera.js App.js

```
import React from "react";

import PickImage from "./screens/Camera.js";

export default class App extends React.Component {
  render() {
    return <PickImage/>;
  }
}
```

```
render() {
  let { image } = this.state;

  return (
    <View style={{ flex: 1, alignItems: "center", justifyContent: "center" }}>
      <Button
        title="Pick an image from camera roll"
        onPress={this._pickImage}
      />
    </View>
  );
}
```

```
getPermissionAsync = async () => {
  if (Platform.OS !== "web") {
    const { status } = await Permissions.askAsync(Permissions.CAMERA_ROLL);
    if (status !== "granted") {
      alert("Sorry, we need camera roll permissions to make this work!");
    }
  }
};
```

```
componentDidMount() {
  this.getPermissionAsync();
}
```

```
_pickImage = async () => {
  try {
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.All,
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1,
    });
    if (!result.cancelled) {
      this.setState({ image: result.data });
      console.log(result.uri)
      this.uploadImage(result.uri);
    }
  } catch (E) {
    console.log(E);
  }
};
```

```javascript
uploadImage = async (uri) => {
  const data = new FormData();
  let filename = uri.split("/")[uri.split("/").length - 1]
  let type = `image/${uri.split('.')[uri.split('.').length - 1]}`
  const fileToUpload = {
    uri: uri,
    name: filename,
    type: type,
  };
  data.append("digit", fileToUpload);
  fetch("https://07afd951a187.ngrok.io/predict-digit", {
    method: "POST",
    body: data,
    headers: {
      "content-type": "multipart/form-data",
    },
  })
    .then((response) => response.json())
    .then((result) => {
      console.log("Success:", result);
    })
    .catch((error) => {
      console.error("Error:", error);
    });
};
```

2. Host the local host server on ngrok.

```
ngrok http 5000
```

```
ashura@zeros:~/Desktop/API$ python3.8 app.py
/home/ashura/.local/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was rea
ched which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
/home/ashura/.local/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was rea
ched which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
 * Debugger is active!
 * Debugger PIN: 328-239-193
```

```
data.append("digit", fileToUpload);
fetch("https://07afd951a187.ngrok.io/predict-digit", {
  method: "POST",
  body: data,
  headers: {
    "content-type": "multipart/form-data",
  },
```

3. Start the server and check for output.

```
$ expo start -c
```

```
Running application on vivo Y51L.
file:///data/data/host.exp.exponent/cache/ExperienceData/%2540anonymous%252FcameraDetection-57504155-ddae-49fb-8b3a-3acf11
c629ec/ImagePicker/87d79f86-f61e-4634-89a8-7af7b719862d.jpg
Success: Object {
  "prediction": "2",
}
```

What's NEXT?

In the next class, we will explore more usage of API and it's methods.

EXTEND YOUR KNOWLEDGE:

Learn about the ngrok service through the following doc: https://ngrok.com/docs