Ghost In the Machine

Kexin_Mei_20013506

My Github Link :

https://github.com/orionmel/Coding3_Final_Artwork_Exploring_To_Machine_Intelligence

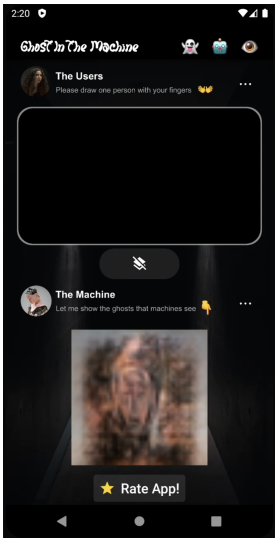
Video Link :

https://vimeo.com/721718129

Project Summary

She drew a rough outline with stick figures, She tried to draw what she remembered as a lover，and I invoked my learning ability to try to draw what humans wanted to look like, but the result was deformed, ugly, chaotic, unclear, and one day people said, "There must be ghosts in that machine, and they produce such a terrible look..."

This project mainly satirizes the human worship of machines, but the fear of the power of the machine in the heart, when the machine outputs blurred and deformed images, humans think that ghosts and gods are in it, and the machine always tries to show the information provided by humans to the best and real look, but people have more unease about him.



What I used in this project

This project is mainly through the pix2pix cGAN in machine learning to train images, by entering a rough sketch to convert into photos, because I want to create an app that let user painting and the machine guesses the image , so I trained this model with the existing pix2pix cGAN code (more details can be viewed through the description below)

I use the dart language and flutter framework structure to create the UI and functionality of the app I want, and I'm running with the local API

Link：

https://colab.research.google.com/drive/1GUEl-2DGPhc4Zsl5HLI3s0AQwRhkhwtD?usp=sharing

https://generated.photos/faces

https://www.tensorflow.org/tutorials/generative/pix2pix

https://docs.flutter.dev/development/ui/layout

https://www.youtube.com/watch?v=mK9N0obmiZQ

Process

The following mainly shows the important code, to learn the full code please go through the links below 👇

https://github.com/orionmel/Coding3_Final_Artwork_Exploring_To_Machine_Intelligence/tree/main/Ghost_In_the_Machine/lib

1. 1 Create home screen

- In the [main.dart] page

```
class MyApp extends StatelessWidget {
  // 此小部件是应用程序的根。
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Ghost In the Machine',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,

    ),
    );
  }
}
```

- In the [homeScreen.dart] page

```
@override
  Widget build(BuildContext context)
  {
    //添加背景图
    return Scaffold(
      //标题颜色和主题
      appBar: AppBar(
```

```
        //标题❤❤❤❤❤❤❤❤
        flexibleSpace: Container(
          decoration: BoxDecoration(
            image: DecorationImage(
              image: AssetImage('assets/title.png'),
              fit: BoxFit.cover
            )
          ),
        ),
      ),
      body: Container(
        //背景图
        decoration: BoxDecoration(
          image: DecorationImage(
              image: AssetImage("assets/background_img.png"),
              fit:BoxFit.fill
          ),
        ) ,
 );
}
```

1.2 Create drawing place

- In the [homeScreen.dart] page

```
child: Stack(
          children: [
            Center(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.center,
                mainAxisAlignment: MainAxisAlignment.center,
                children: [

                  //用户
                  Padding(
                    padding: EdgeInsets.symmetric(vertical: 0.2),
                    child: Image(
                      image: AssetImage('assets/human.png'),
                    ),
                  ),

                  //黑色画布大小
                  Padding(
                    padding: EdgeInsets.all(7.0),
                    //黑色画布大小
                    child: Container(
                      width: 356,
                      height: 200,
                      decoration: BoxDecoration(
                        borderRadius: BorderRadius.all(Radius.circular(22)),
                        boxShadow:
                        [
                          BoxShadow(
                            color: Colors.white54,
                            //blurRadius: 6.0,
                            spreadRadius: 2,
                          ),
                        ]
                      ),

                      //✍✍✍✍✍✍ 画笔
                      child: GestureDetector(
                        onPanDown: (details){
                          this.setState(() {
                            allPoints.add(
                              SetDrawingArea(
                                points: details.localPosition,
                                paintArea: Paint()
                                  ..strokeCap = StrokeCap.round
                                  ..isAntiAlias = true
                                  ..color = Colors.white
                                  ..strokeWidth = 2.0
                              ),
```

```
                          );
                        });
                      },
                      onPanUpdate: (details){
                        this.setState(() {
                          allPoints.add(
                            SetDrawingArea(
                                points: details.localPosition,
                                paintArea: Paint()
                                  ..strokeCap = StrokeCap.round
                                  ..isAntiAlias = true
                                  ..color = Colors.white
                                  ..strokeWidth = 2.0
                            ),
                          );
                        });
                      },
                      onPanEnd: (details)
                      {
                        saveSketchToImage(allPoints);
                        this.setState(() {
                          allPoints.add(null);
                        });
                      },
                      child: SizedBox.expand(
                        child: ClipRRect(
                          borderRadius:BorderRadius.all(
                            Radius.circular(22.0),
                          ),
                          child: CustomPaint(
                            painter: MyCustomPainter(allPoints: allPoints),
                          ),
                        ),
                      ),
                    ),
                  ),
                ),
              ),
        ),
      ),
    );
```

2. Create splash screen

- In the [homeScreen.dart] page

```
class MyApp extends StatelessWidget {
  // 此小部件是应用程序的根。
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      //设计主页mysplashscreen
      home: MySplashScreen(),
    );
  }
}
```

- In the [splashScreen.dart] page

```
import 'package:flutter/material.dart';
import 'package:Ghost_In_the_Machine/homeScreen.dart';
import 'package:splashscreen/splashscreen.dart';

class MySplashScreen extends StatefulWidget {
  //const MySplashScreen({Key? key}) : super(key: key);

  @override
```

```
  State<MySplashScreen> createState() => _MySplashScreenState();
}

class _MySplashScreenState extends State<MySplashScreen> {
  @override
  Widget build(BuildContext context)
  {
    return SplashScreen(
      seconds: 10,
      navigateAfterSeconds: HomeScreen(),
      imageBackground: Image.asset("assets/background_img.png").image,
      image: new Image.asset('assets/main.png'),
      photoSize: 220.0,
      useLoader: true,
      loaderColor: Colors.red,
      loadingText: Text(
        "From Kexin Mei ",
        style: TextStyle(
          color: Colors.yellowAccent,
          fontSize: 16.0,
        ),
      ),
    );
  }
}
```

3.1 Draw and display

- In the [setDrawingArea.dart] page

```
import 'dart:ui';
import 'package:flutter/material.dart';

class SetDrawingArea{
  Offset points;
  Paint paintArea;

  SetDrawingArea({
    this.points,
    this.paintArea,
  });
}

class MyCustomPainter extends CustomPainter
{
  List<SetDrawingArea> allPoints =[];

  MyCustomPainter({List<SetDrawingArea> allPoints}) : this.allPoints = allPoints.toList();

  @override
  void paint(Canvas canvas, Size size) {
    // TODO: implement paint
    Paint background = Paint()..color = Colors.black;
    Rect rect = Rect.fromLTWH(0, 0, size.width, size.height);
    canvas.drawRect(rect, background);
    canvas.clipRect(rect);

    for(int i = 0; i<allPoints.length -1;i++){
      if(allPoints[i] !=null && allPoints[i+1] !=null)
      {
        canvas.drawLine(allPoints[i].points,allPoints[i+1].points,allPoints[i].paintArea);
      }
      else if(allPoints[i] !=null && allPoints[i+1] ==null)
      {
        canvas.drawPoints(PointMode.points, [allPoints[i].points], allPoints[i].paintArea);
      }
    }
  }

  @override
```

```
  bool shouldRepaint(MyCustomPainter oldDelegate) {
    // TODO: implement shouldRepaint
    return oldDelegate.allPoints != allPoints;
  }

}
```

3.2 Clear drawing image

- In the [homeScreen.dart] page

```
    Container(
                    width: MediaQuery.of(context).size.width*0.30,
                    height: 45.0,
                    decoration: BoxDecoration(
                      color: Colors.white10,
                      borderRadius: BorderRadius.circular(22.0),
                    ),
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: [
                        IconButton(
                          icon: Icon(Icons.layers_clear,color: Colors.white,),
                          onPressed: (){
                            this.setState(() {
                              allPoints.clear();
                            });
                          },
                        ),
                      ],
                    ),
                  ),
```

4.1 Using neural network

My source code :

https://colab.research.google.com/drive/1GUEl-2DGPhc4Zsl5HLI3s0AQwRhkhwtD?usp=sharing

https://generated.photos/faces

4.2 Building server and API

My Flask Server Files Link :

https://github.com/orionmel/Coding3_Final_Artwork_Exploring_To_Machine_Intelligence/tree/main/FaceAppFlaskServer

```
from keras.models import load_model
from  flask import Flask, jsonify, request
import base64
import io
from keras.preprocessing.image import img_to_array
import numpy as np

import  time
from keras.preprocessing.image import save_img
import tensorflow as tf
from PIL import Image
from flask_restful import Resource, Api, reqparse
```

```
app = Flask(__name__)
api = Api(app)



model = load_model("test.h5")
print(" * Model Loaded ")




def make_image(image, target):
    if image.mode != "RGB":
        image = image.convert("RGB")

    image = image.resize(target)
    image = img_to_array(image)

    image = (image - 127.5) / 127.5
    image = np.expand_dims(image, axis=0)
    return image




class PredictFace(Resource):
    def post(self):
        json_data = request.get_json()
        img_data = json_data["Image"]

        image = base64.b64decode(str(img_data))

        img = Image.open(io.BytesIO(image))

        preparedImage = make_image(img, target=(256,256))

        pred = model.predict(preparedImage)

        outputFile = "output.png"
        savePath = "./output/"

        output = tf.reshape(pred, [256,256,3])
        output = (output + 1) / 2
        save_img(savePath+outputFile, img_to_array(output))

        imageNew = Image.open(savePath+outputFile)
        imageNew = imageNew.resize((50,50))
        imageNew.save(savePath+"new_"+outputFile)

        with  open(savePath+"new_"+outputFile, "rb") as image_file:
            encoded_string = base64.b64encode(image_file.read())

        outputData = {
            "Image" : str(encoded_string),
        }
        return outputData



api.add_resource(PredictFace, "/predict")

if __name__ == "__main__":
    app.run(debug=True)
```

4.3 Output image

- In the [homeScreen.dart] page

```
class _HomeScreenState extends State<HomeScreen>
{
  List<SetDrawingArea> allPoints = [];

  Widget outputImageWidget;


  //图像呈现
  void saveSketchToImage(List<SetDrawingArea>points)async
  {
    final sketchRecorder = ui.PictureRecorder();

    final canvas = Canvas(sketchRecorder,Rect.fromPoints(Offset(0.0,0.0),Offset(200,200)));

    Paint paint = Paint()
      ..color = Colors.white
      ..strokeCap = StrokeCap.round
      ..strokeWidth = 2.0;

    final paintBackground = Paint()
      ..style = PaintingStyle.fill
      ..color = Colors.black;

    canvas.drawRect(Rect.fromLTWH(0, 0, 356, 256),paintBackground);

    for(int i = 0; i<points.length -1;i++){
      if(points[i] !=null && points[i+1] !=null)
      {
        canvas.drawLine(points[i].points,points[i+1].points,paint);
      }
    }

    final picture = sketchRecorder.endRecording();
    final imgFile = await picture.toImage(356,200);

    final pngBytes = await imgFile.toByteData(format: ui.ImageByteFormat.png);
    final listbites = Uint8List.view(pngBytes.buffer);

    String base64 = base64Encode(listbites);
    retrieveResponse(base64);
  }


  void retrieveResponse(var base64Image)async{
    var data = {"Image":base64Image};

    //:5000/predict
    var urlFlaskServer = "http://192.168.0.133:5000/predict";

    Map<String,String>headers = {
      "Content-type":"application/json",
      "Accept":"application/json",
      "Connection":"Keep-Alive",
    };

    var body = json.encode(data);

    try
    {
      var response = await http.post(Uri.parse(urlFlaskServer),body:body,headers:headers);

      final Map<String,dynamic> responseData = json.decode(response.body);

      String outputBytes = responseData["Image"];

      print("This is OUTPUT Bytes");

      print(outputBytes);

      displayOutputImage(outputBytes.substring(2,outputBytes.length - 1));

    }
    catch(e)
    {
      print("Error Occured.");
```

```
      print(e);
      return null;
    }

  }

  //输出图像
  void displayOutputImage(String bytes)async
  {
    Uint8List convertedBytes = base64Decode(bytes);

    setState((){
      outputImageWidget = Container(
        width: 256,
        height: 256,
        child: Image.memory(
          convertedBytes,
          fit: BoxFit.cover,
        ),
      );
    });

  }

}
```

```
        Padding(
                      padding: EdgeInsets.symmetric(vertical: 10.0),
                      child: Container(
                        height: 200,
                        width: 200,
                        child: outputImageWidget,
                      ),
                  ),
```

5.1 Create rating button

- In the [homeScreen.dart] page

```
    TextButton.icon(
                      onPressed: (){
                        openRatingDialog(context);
                      },
                      style: ButtonStyle(
                        backgroundColor: MaterialStateProperty.all(
                          Colors.white10.withOpacity(0.1),
                        ),
                      ),
                      icon: Icon(
                          Icons.star,
                          color: Colors.amberAccent,
                      ),
                      label: Text(
                        'Rate App!',
                        style: Theme.of(context).textTheme.headline6.copyWith(color:Colors.white),

                      ),
                  ),
```

```
openRatingDialog(BuildContext context){
    showDialog(
        context: context,
        builder:(context){
          return Dialog(
            child:RatingView(),
```

```
          );
        },
    );
  }
```

5.2 Create rating page

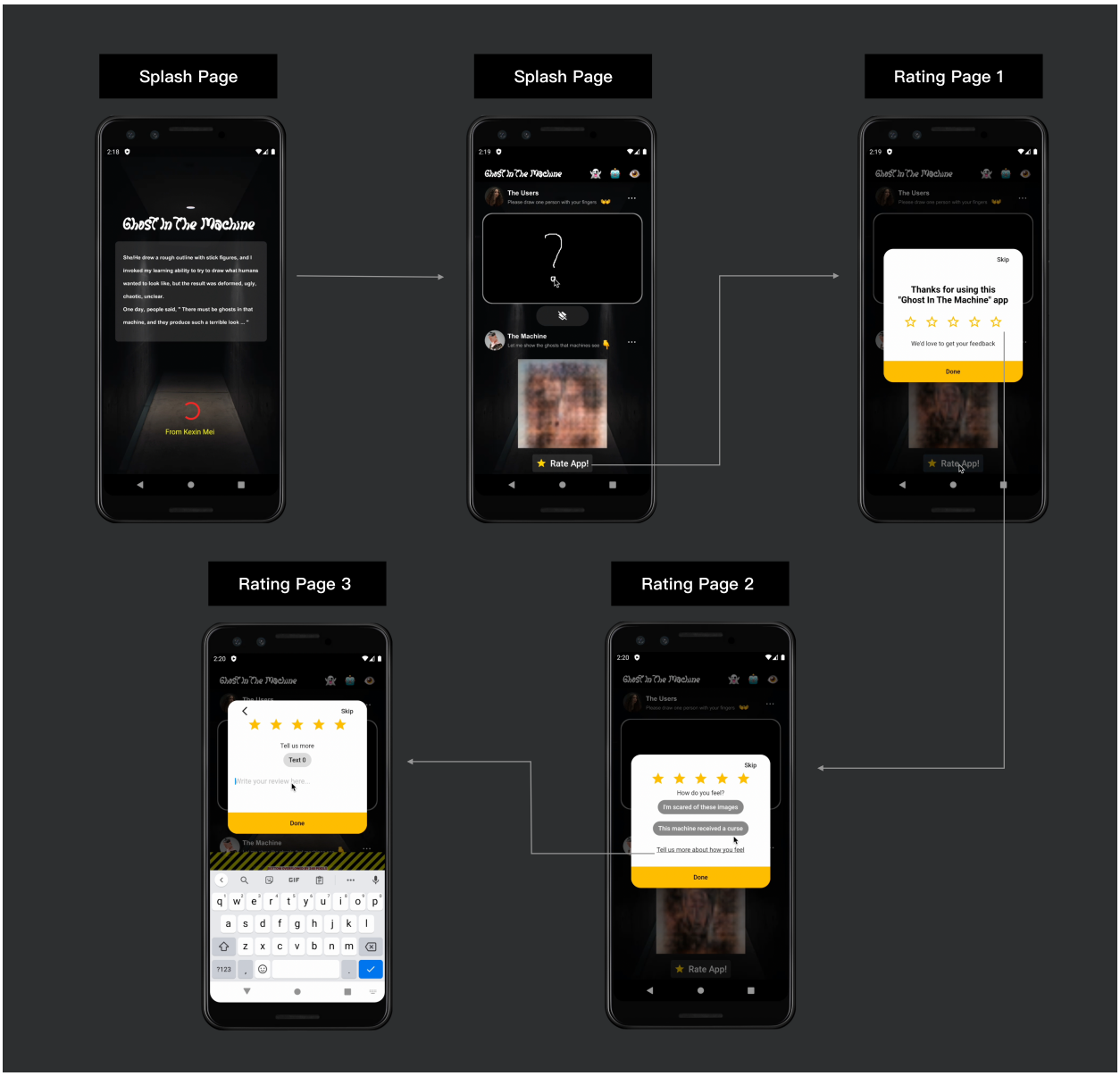
- In the [homeScreen.dart] page

```
import 'dart:math';

import 'package:flutter/material.dart';

class RatingView extends StatefulWidget {
  //const RatingView({Key? key}) : super(key: key);

  @override
  State<RatingView> createState() => _RatingViewState();
}

class _RatingViewState extends State<RatingView> {
  var _ratingPageController = PageController();
  var _starPosition = 140.0;
  var _rating =0;
  var _selectedChipIndex = -1;
  var _isMoreDetailActive = false;
  var _moreDetailFocusNode = FocusNode();

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(22),
      ),
      clipBehavior: Clip.antiAlias,
      child:Stack(
        children: [
          //thanks note
          Container(
            height: max(300, MediaQuery.of(context).size.height*0.3),
            child: PageView(
              controller:_ratingPageController ,
              physics: NeverScrollableScrollPhysics(),
              children: [
                _buildThanksNote(),
                _causeOfRating(),
              ],
            ),
          ),
          //done button
          Positioned(
              bottom: 0,
              left: 0,
              right: 0,
              child: Container(
                color: Colors.amber,
                child: MaterialButton(
                    onPressed: _hideDialog,
                    child: Text('Done'),
                    textColor: Colors.black,
                ),
              ),
          ),
          //skip button
          Positioned(
            right: 0,
            child: MaterialButton(
                onPressed: _hideDialog,
                child: Text('Skip'),
            ),
```

```
          ),
          //star rating
          AnimatedPositioned(
            top: _starPosition,
            left: 0,
            right: 0,
            child: Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: List.generate(
                  5,
                  (index) => IconButton(
                    icon: index < _rating ? Icon(Icons.star,size: 32):Icon(Icons.star_border,size: 32),
                    color: Colors.amber,
                    onPressed: (){
                      _ratingPageController.nextPage(duration: Duration(milliseconds:300), curve: Curves.easeIn);
                      setState((){
                        _starPosition = 30.0;
                        _rating = index +1;
                      });
                    },
                  ),
              ),
            ), duration: Duration(milliseconds: 300),
          ),
          //back button
          if(_isMoreDetailActive)
            Positioned(
              left: 0,
              top: 0,
              child: MaterialButton(
                  onPressed: (){
                    setState((){
                      _isMoreDetailActive = false;
                    });
                  },
                child: Icon(Icons.arrow_back_ios),
              ),
            ),
        ],
      ),
    );
  }

  _buildThanksNote(){
    return Column(
      //文字位置
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      //mainAxisSize: MainAxisSize.min,
      children: [
        Text(
          'Thanks for using this \n"Ghost In The Machine" app',
          style: TextStyle(
            fontSize: 20,
            color: Colors.black,
            fontWeight: FontWeight.bold,
          ),
          textAlign: TextAlign.center,
        ),
        Text('We\'d love to get your feedback'),
        //Text('How was your ride today'),
      ],
    );
  }

  _causeOfRating(){
    return Stack(
      alignment: Alignment.center,
      children: [
        Visibility(
          visible: ! _isMoreDetailActive,
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                Text('How do you feel?'),
                //选择框
                Wrap(
```

```
                spacing: 8.0,
                alignment: WrapAlignment.center,

                children: const [
                  Chip(
                    label: Text('I\'m scared of these images'),
                    labelStyle: TextStyle(color: Colors.white),
                    //avatar: Icon(Icons.backup,color: Colors.white),
                    backgroundColor: Colors.black45,
                  ),

                  Chip(
                    label: Text('This machine received a curse'),
                    labelStyle: TextStyle(color: Colors.white),
                    //avatar: Icon(Icons.all_inbox,color: Colors.white),
                    backgroundColor: Colors.black45,

                  ),

                ]

              ),
              SizedBox(height: 16),
              InkWell(
                onTap: (){
                  _moreDetailFocusNode.requestFocus();
                  setState((){
                    _isMoreDetailActive = true;
                  });
                },
                child:Text(
                  'Tell us more about how you feel',
                  style: TextStyle(decoration: TextDecoration.underline),
                ),
              ),
            ],
          ),
        replacement: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            Text('Tell us more'),
            Chip(label: Text('Text ${_selectedChipIndex + 1}')),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 16.0),
              child: TextField(
                focusNode: _moreDetailFocusNode,
                decoration: InputDecoration(
                  hintText: 'Write your review here...',
                  hintStyle: TextStyle(
                    color: Colors.grey[400],
                  ),
                  border: InputBorder.none,
                ),
              ),
            ),

          ],
        ),
      ),
    ],
  );
 }
 _hideDialog(){
   if(Navigator.canPop(context)) Navigator.pop(context);
 }
}
```

Final Outcomes



Reference

https://github.com/codingcafe1/human_face_generator_app

https://generated.photos/faces

https://www.youtube.com/watch?v=PipsYvmUT7c

https://docs.flutter.dev/development/ui/layout

https://www.tutorialkart.com/flutter/flutter-canvas-draw-rectangle/

https://www.youtube.com/watch?v=mK9N0obmiZQ

https://www.youtube.com/watch?v=E8T4JQZFZdg

https://www.kindacode.com/article/using-chip-widget-in-flutter/

https://api.flutter.dev/flutter/material/Icons-class.html