AN INTELLIGENT MOBILE FLOATING APPLICATION TO AID SENIORS IN USING THE SMARTPHONE USING MACHINE LEARNING

Yilin Chen[1] and Yu Sun[2]

[1]Troy High School, 2200 Dorothy Ln, Fullerton, CA 92831, USA
[2]California State Polytechnic University, Pomona, CA, 91768, USA

ABSTRACT

The twenty-first century is a century of rapid technological growth, one significant area being the smartphone [4]. By 2021, more than eight percent of US adults own a smartphone. Smartphones are capable of making phone calls, messaging texts, making purchases, taking pictures, playing games, finding roads, and more. However, not everyone is a beneficiary of this technology. Seniors often fall behind in this technology advancement. They often struggle with finding the right button to press or get confused with the variety of functions. This paper develops a floating application that when launched, checks the opening application and displays a list of its functions. Then, the user can select what they want to do, and the application will begin a tutorial to guide the senior in using their phone. We applied our application to Google Play and conducted a qualitative evaluation of the approach. The results show that this application will be effective in facilitating seniors in using the smartphone.

KEYWORDS

Machine Learning，Big Data，Mobile Application.

1. INTRODUCTION

The twenty-first century saw a rapid growth of technology in transportation, aerospace, scientific research, recreation, home utilities and communication [5]. Since tech giant IBM's development of the world's first smartphone---Simon---smartphones and mobile devices have undergone a series of revolutions and changes. From sending messages to searching online forums, from taking pictures to editing videos, the function of the smartphone has extended to all parts of our lives [6]. However, the reliance on technologies has resulted in a series of problems. Whilst appreciating technology's great power, the term "technical difficulties" is common without professionals. Even the national television makes errors with technology sometimes. This phenomenon is especially common among the older generation, or rather, any generation other than Generation Z. In 2021, the digital divide is not only between the rich and the poor, but has also emerged between the young and the old [7].

In the world of COVID-19, the reliance on technology has created a series of issues for those who are unfamiliar with it. In the beginning of the pandemic, many stores and restaurants were shut down, most of which shifted their service online [8]. Many schools and workplaces require online qualtrics or surveys to enter. Even vaccine centers rely on online appointments. However, these interfaces are usually filled with information too complicated for seniors， because it is hard for older people. While some elder-lies may ask young people for help, this may not be the case all the time. Also, this is merely shifting the reliance on technology to the reliance on youngsters and does not solve the problem. Should technology wait for the people? But there are people benefiting from it. The implementation of technology during COVID-19 has encouraged social distance, reduced face-to-face contacts, and facilitated contact tracing [15]. Thus, we came to the conclusion that assistance is needed to seniors and those who struggle with technology.

There have been some proposals to facilitate seniors in using their smartphones. Some phones and applications have incorporated an Easy Mode into their program. For example, the Easy Mode on Samsung Galaxy Smartphones alters the UI to a version with a simpler layout, larger icons, and larger texts. The Bald Phone is another interface designed for senior citizens. It replaces the phone's original interface into a bigger and simpler one. However, this approach has many limitations. First, this approach makes some settings and functions unavailable, which may be more counterproductive in some circumstances. Because the algorithm does not apply on browsers and websites, users will still struggle on the internet. If the easy mode or interface is not up to date, they will still be struggling with technology when they want to use a new function that just came out. Additionally, this approach does not teach the user how to perform a certain function. While it helps with some functions on the smartphone, the user will still be left in confusion when they need to use technology on other devices. The fundamental problem, seniors' struggle to understand the necessary workflow in using an interface, is neglected [9].

Our proposed method is a floating application that provides tutorials to functions on a variety of APPs. When the user needs help, they can simply click on the floating icon and the app will display a list of available tutorials in the form of questions. The user can then select a question they have, and a corresponding tutorial will display by drawing indicators and notes on the screen. Our goal is not only to facilitate senior usage in the mobile phone with our application, but also to teach seniors how to perform a function in the normal interface. This approach will teach seniors the processes of normal APPs with guidance and help them understand the fundamental logic behind APP layouts [10]. Ideally, the seniors will eventually be able use APPs without any additional help.

In two application scenarios, we demonstrate how the above combination of techniques increases. First, we show the usefulness of our approach by a comprehensive case study on the evolution of the Easy Mode application. Second, we analyze the evolution of our floating app. To show our solution actually solves the problem, we design two different experiments. Experiments 1 test AI prediction results with test cases, Experiments 2 shows user surveys which can help us know if the app is useful or not.

The rest of the paper is organized as follows: Section 2 outlines the details on the challenges we encountered while designing the sample; Section 3 describes our methodology and algorithm in detail; Section 4 presents the experiments and evaluations process of designing our APP; Section 5 discusses works relating to our research; finally, Section 6 ends the paper with a conclusion and remarks on future works.

2. CHALLENGES

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

2.1. Identify the type of the APP the user is on

The first challenge we encountered was how to identify the type of the APP the user is on in order to provide the correct list of tutorials. We don't want to have the user tell us and want our algorithm to be able to detect it. Our solution was to take a screenshot of the screen, and identify it with machine learning using our database, which consists of screenshots from a variety of APPs [12]. Another challenge embedded in taking the screenshots is the timing. When the user clicks on the icon, the APP takes a screenshot of the screen and displays a set of functions in the form of questions. However, if the timing is incorrect, the list might display after the screenshot, and the algorithm will not be able to evaluate the correct APP.

2.2. Overlaying the instructions on screen real-time

Another challenge we faced was how to overlay the instructions on screen real-time. We want to put our tutorials directly above the user's screen instead of displaying a marked up screenshot. This is because we need to work with other APPs to draw tutorials over it. It is a struggle to have third-party APPs cooperate with us. Also, this function requires the APP to be constantly aware of what the current screen is displaying. If the display changed, we can't have the drawn tutorials still be in their original places.

2.3. Choose the right tip and rendering strategy

Another challenge we faced was how to accurately choose the right tip and rendering strategy [11]. To display our tutorial, we will draw boxes and add notes to let the user know where to press. However, we can't manually draw out all tutorials for all screen sizes, so we calculate it through keywords. It is very difficult because different phones have buttons of different shapes and sizes. Although we can find the general area at which the rectangles should be drawn, it is very difficult to get the box right around the button. We also faced the risk of the screen changing after we determined where to draw the box and write the notes.

3. SOLUTION

Tutorial4Senior is an intelligent mobile floating application to aid seniors in using the smartphone using machine learning. When first launched, the APP will ask for permission to float over other APPs and permission to take screenshots. The APP will then appear on the screen as a floating icon. When a user needs help, the APP will take a screenshot of the current screen and identify what APP the user is using, using machine-learned data from our screenshot database. Then, the APP will display a list of questions that asks how to perform a function on the APP such as: How to write a message? The user can then select a question that matches their needs. The APP will then draw boxes and display notes on the screen to indicate what the user should do to achieve their purpose.

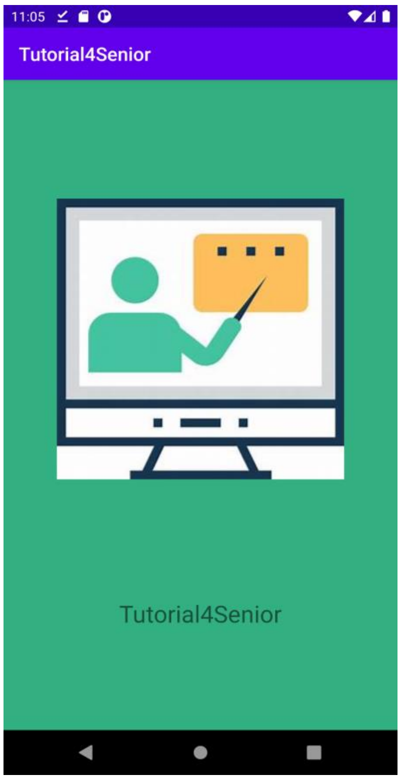
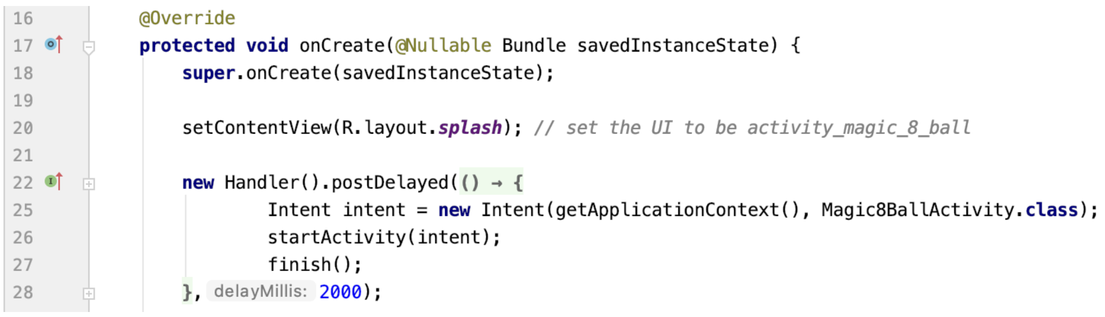


Figure 1. Screenshot of Tutorial4Senior



```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.splash); // set the UI to be activity_magic_8_ball

    new Handler().postDelayed(() → {
            Intent intent = new Intent(getApplicationContext(), Magic8BallActivity.class);
            startActivity(intent);
            finish();
    }, delayMillis: 2000);
```

Figure 2. SplashActvitiy.java

When the APP is first launched, the user will first see a splash screen, giving the APP a few seconds to load completely. The implementation of the splash screen is done using a very standard Android activity. It is based on a standard layout with a logo in the center. The Android intent is used to delay and trigger the start of the next screen. The default delay is 2 second but we can customize the delay time with any given number.

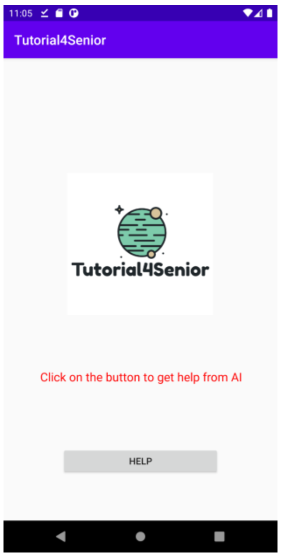
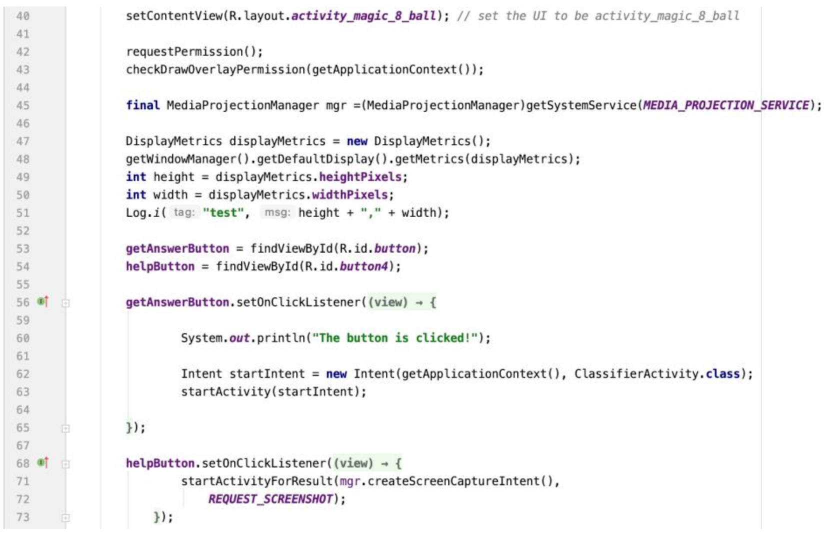


Figure 3. Screenshot of getting help



```java
setContentView(R.layout.activity_magic_8_ball); // set the UI to be activity_magic_8_ball

requestPermission();
checkDrawOverlayPermission(getApplicationContext());

final MediaProjectionManager mgr =(MediaProjectionManager)getSystemService(MEDIA_PROJECTION_SERVICE);

DisplayMetrics displayMetrics = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
int height = displayMetrics.heightPixels;
int width = displayMetrics.widthPixels;
Log.i( tag: "test",  msg: height + "," + width);

getAnswerButton = findViewById(R.id.button);
helpButton = findViewById(R.id.button4);

getAnswerButton.setOnClickListener((view) → {

        System.out.println("The button is clicked!");

        Intent startIntent = new Intent(getApplicationContext(), ClassifierActivity.class);
        startActivity(startIntent);

});

helpButton.setOnClickListener((view) → {
        startActivityForResult(mgr.createScreenCaptureIntent(),
            REQUEST_SCREENSHOT);
    });
```

Figure 4. Magic8BallActivity.java

This is the home screen of Tutorial4Senior. The code sets up a button that listens for the user to press. This is a very critical screen for the whole application [13]. Once the user presses the button the app will first check the permissions required to run this app. This app runs at the system level because it needs to check the status of the app as well as putting overlays and getting screenshots at the system level [14]. These types of functionality require system permissions such as recording the screen, rendering content as overlays on the other apps, and setting the floating icon as a launcher app on the top of the screen. If any of these permissions were not given by the user when we run this app, the app will pop up the dialogue to ask the user to enable these permissions before we can move on to the next screen. A specific 3rd party library is used to simplify the permission checking and permission requesting process, as a standard Android way of getting these permissions requires more code. Once all the permissions are given and set, this screen will be stopped and the app will move on to the launcher app mode.

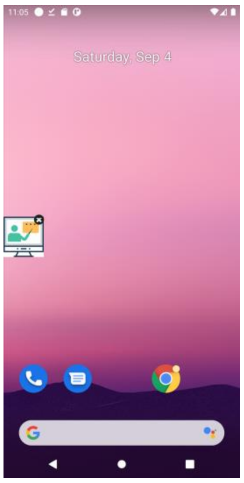
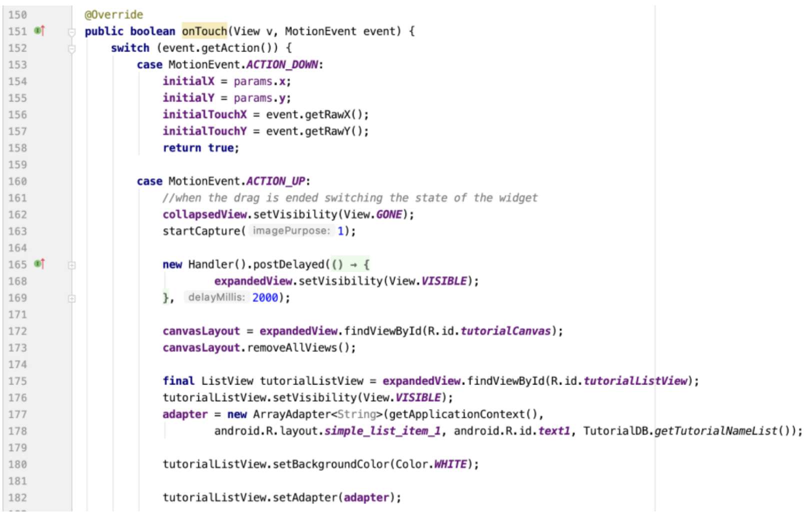


Figure 5. Screenshot of the button

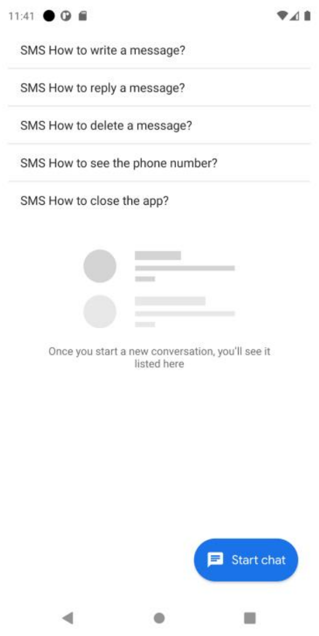


```java
@Override
public boolean onTouch(View v, MotionEvent event) {
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            initialX = params.x;
            initialY = params.y;
            initialTouchX = event.getRawX();
            initialTouchY = event.getRawY();
            return true;

        case MotionEvent.ACTION_UP:
            //when the drag is ended switching the state of the widget
            collapsedView.setVisibility(View.GONE);
            startCapture( imagePurpose: 1);

            new Handler().postDelayed(() -> {
                    expandedView.setVisibility(View.VISIBLE);
            },  delayMillis: 2000);

            canvasLayout = expandedView.findViewById(R.id.tutorialCanvas);
            canvasLayout.removeAllViews();

            final ListView tutorialListView = expandedView.findViewById(R.id.tutorialListView);
            tutorialListView.setVisibility(View.VISIBLE);
            adapter = new ArrayAdapter<String>(getApplicationContext(),
                    android.R.layout.simple_list_item_1, android.R.id.text1, TutorialDB.getTutorialNameList());

            tutorialListView.setBackgroundColor(Color.WHITE);

            tutorialListView.setAdapter(adapter);
```

Figure 6. FloatingViewService.java (1)

An OnTouch method is set up for the float. The widget can be moved around the screen at the system level. When it's clicked on, it captures a screenshot of the screen. This is the core part of the code of the whole application. It mainly does the following tasks: first of all, it allows the app to run as a launcher in mode and display an icon on the top of the screen. Secondly the app can trigger getting and capturing a screenshot of the whole app whenever that user creates a contact button, and the screenshot will be sent to the machine learning engine for further analysis. Next the machine learning engine which is built-in into the app will read the image and run a deep learning classification to try to figure out what type of app is. Lastly, once we figure out the type of the app, the relevant tips and tutorials about this app will be shown up on the top of the screen. The user can select one of the tips from the screen and figure out the tutorial mode to see all the help information. More details about the tips and tutorial rendering will be discussed in the next section.

In order to differentiate the different types of app based on a given screenshot, we have applied deep learning and neural networks to do the prediction and classification. Tensorflow Lite has been applied in this case, because of its advantage of a small-sized and well-trained image classification model that is applicable on mobile devices. Running deep learning analysis on mobile devices is really important as it provides an almost real time prediction without the costs of sending images to the cloud and waiting for the result to be sent back. In our experiments which will be discussed in section 4, we have trained over 500 screenshot images for 10 different Android apps in the Android operating system. It turns out that the result of image classification in this case is very accurate. Even though some of the apps share very similar UI, the TensorFlow based machine learning model can precisely differentiate the different types of app if sufficient images are given in the training data-set. This is a key differentiator of our app compared with most other tutorial apps available in the store, since most other apps require users to identify manually what the type of app is. However, a lot of users, particularly the senior users do not even know the name of the app, and that is when these kinds of apps cannot be used as effectively as designed. Thus, we believe that the AI-based image classification can close this gap and greatly improve the user experience.



11:41

SMS How to write a message?

SMS How to reply a message?

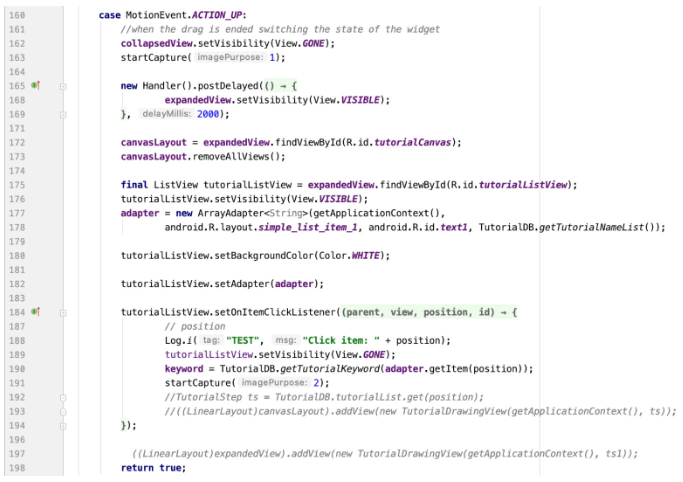
SMS How to delete a message?

SMS How to see the phone number?

SMS How to close the app?

Once you start a new conversation, you'll see it listed here

Figure 7. Screenshot of general questions



```
case MotionEvent.ACTION_UP:
    //when the drag is ended switching the state of the widget
    collapsedView.setVisibility(View.GONE);
    startCapture( imagePurpose: 1);

    new Handler().postDelayed(() → {
            expandedView.setVisibility(View.VISIBLE);
    }, delayMillis: 2000);

    canvasLayout = expandedView.findViewById(R.id.tutorialCanvas);
    canvasLayout.removeAllViews();

    final ListView tutorialListView = expandedView.findViewById(R.id.tutorialListView);
    tutorialListView.setVisibility(View.VISIBLE);
    adapter = new ArrayAdapter<String>(getApplicationContext(),
            android.R.layout.simple_list_item_1, android.R.id.text1, TutorialDB.getTutorialNameList());

    tutorialListView.setBackgroundColor(Color.WHITE);

    tutorialListView.setAdapter(adapter);

    tutorialListView.setOnItemClickListener((parent, view, position, id) → {
            // position
            Log.i( tag: "TEST", msg: "Click item: " + position);
            tutorialListView.setVisibility(View.GONE);
            keyword = TutorialDB.getTutorialKeyword(adapter.getItem(position));
            startCapture( imagePurpose: 2);
            //TutorialStep ts = TutorialDB.tutorialList.get(position);
            //((LinearLayout)canvasLayout).addView(new TutorialDrawingView(getApplicationContext(), ts));
    });

      ((LinearLayout)expandedView).addView(new TutorialDrawingView(getApplicationContext(), ts1));
    return true;
```

Figure 8. FloatingViewService.java (2)

Based on the result obtained from the deep learning model, we can decide the type of the app. A list of tips and tutorials have been stored in our local database for different kinds of apps. The relevant app tips will be filtered out in this case so that the users will only see the useful tips in tutorials based on the current context.

The database of storing all the tips in tutorials has been structured in such a way that it is very easy to extend it with more types of apps or tips. We have implemented a very efficient search engine to quickly match the tips in tutorial space on the name, tags, keywords, so that as long as we know the name of the app, all the relevant tutorials will be organized and displayed on top of the screen. One special UI implementation we have done here is to make sure the background color of the ListView is transparent. This is very important because it provides and allows users to see the actual app running which we believe will make the users easier to choose the right tips and tutorials. Clicking on a tutorial will trigger the next step of the tutorial help.

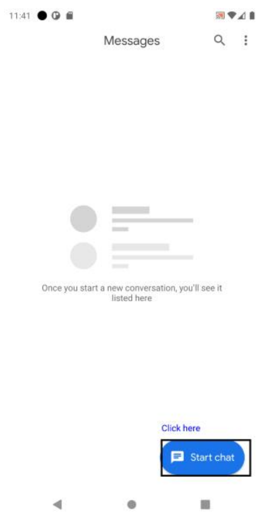
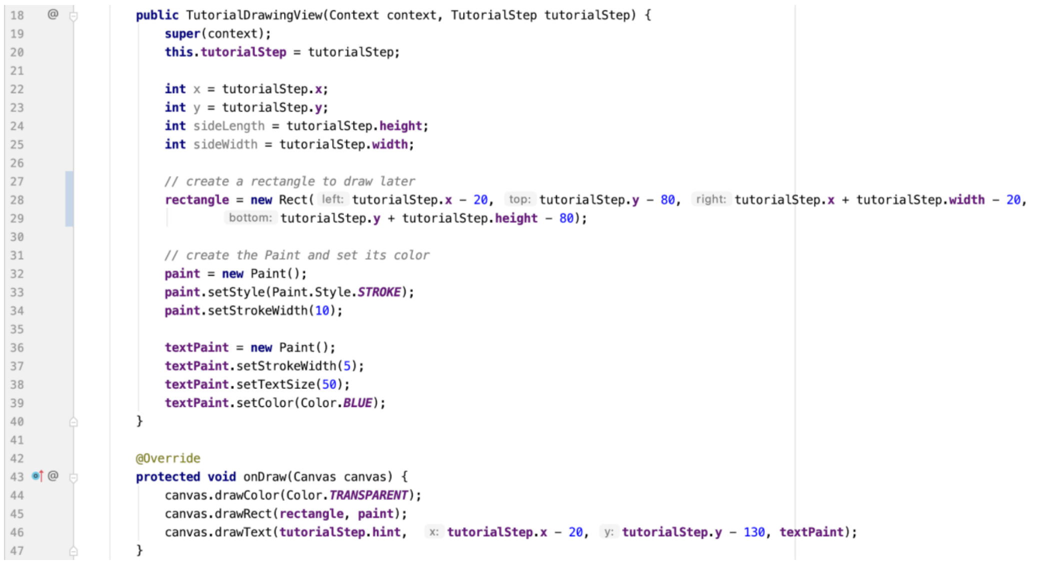

11:41
Messages
Once you start a new conversation, you'll see it listed here
Click here
Start chat

Figure 9. Screenshot of messages



```java
public TutorialDrawingView(Context context, TutorialStep tutorialStep) {
    super(context);
    this.tutorialStep = tutorialStep;

    int x = tutorialStep.x;
    int y = tutorialStep.y;
    int sideLength = tutorialStep.height;
    int sideWidth = tutorialStep.width;

    // create a rectangle to draw later
    rectangle = new Rect( left: tutorialStep.x - 20,  top: tutorialStep.y - 80,  right: tutorialStep.x + tutorialStep.width - 20,
             bottom: tutorialStep.y + tutorialStep.height - 80);

    // create the Paint and set its color
    paint = new Paint();
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeWidth(10);

    textPaint = new Paint();
    textPaint.setStrokeWidth(5);
    textPaint.setTextSize(50);
    textPaint.setColor(Color.BLUE);
}

@Override
protected void onDraw(Canvas canvas) {
    canvas.drawColor(Color.TRANSPARENT);
    canvas.drawRect(rectangle, paint);
    canvas.drawText(tutorialStep.hint,  x: tutorialStep.x - 20,  y: tutorialStep.y - 130, textPaint);
}
```

Figure 10. TutorialDrawingView.java

As you can see in the figure shown above, the displayed tips and tutorials are purely based on the current setting of the screen. It is not a hard-coded content or image like most other apps have been providing. Instead, the app tries to identify and locate the important UI widgets such as buttons and text boxes, and draw the overlay and highlight the position for users to interact directly on top of the screen. Even if the screen size changes or the app updates the UI, as long as the label on this UI widget remains the same, the algorithm will be capable of identifying the UI widgets and displaying overlay the tutorial content in the precise position.

The main technique we use to identify the UI widget is based on text recognition. Every tip or tutorial has a bunch of keywords that we used in searching. We will send the screenshot of the app to the machine learning engine which will trigger the tax recognition algorithm and extract all the tax on that image. We will run a quick search algorithm to match all the text with the relevant keywords in each of the tips. If we find that any of the tapes is matched based on these keywords, we will compare all the match tips with their confidence about us. The highest confidence battle will determine which is the final tip and the rendering content to use. Every tip and the tutorial has specific overlay content information to display. We have developed a simple content rendering engine in the app so that it can briefly and clearly draw most of the commonly used highlights such as rectangles, boxes, text and labels, using different colors.

4. Experiment

4.1. Experiment 1

In experiment 1, We want to test the accuracy of the Deep Learning Image Classification Model. To evaluate the Precision, recall rate, accuracy and F1score of our deep learning model, we have collected 200 real dataset from 5 different users. Every user tries 40 different times and records if the deep learning model predicts the operation correctly. In order to compare the approaches, we conducted experiments to verify three different aspects: the accuracy of using different deep learning networks, and the result table of the network with the best accuracy performance.

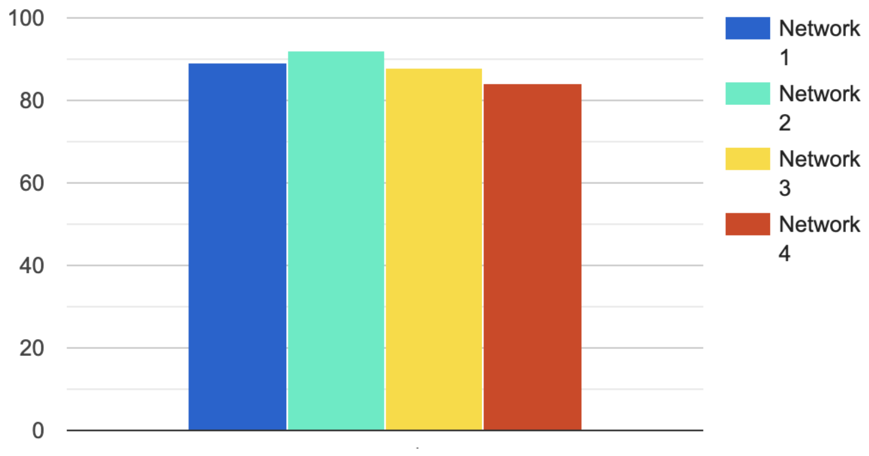
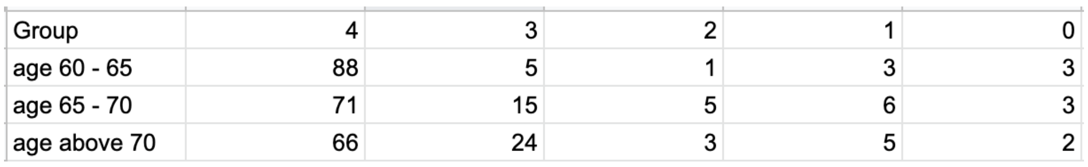


Figure 11. Experiment result

The result shows the second network has the best performance with the accuracy at 94%. It shows it can successfully predict the senior user’s next operation on the mobile phones most of the time. We test the network to find the Confusion Matrix and the result shows below: Accuracy = 94%, precision = 80%

4.2. Experiment 2

In experiment 2, we designed a user review survey to test if the app actually helps senior people and check if they have any reviews and suggestions. We found 300 seniors from LA county, the test group numbers are big enough for a user survey test. We divide them into 3 different groups. Senior ages from 60 - 65 as group one, Senior ages from 65- 70 as group two, Senior ages above 70 as group three, the score table from the survey shows below:



Group	4	3	2	1	0
age 60 - 65	88	5	1	3	3
age 65 - 70	71	15	5	6	3
age above 70	66	24	3	5	2

Figure 12. Table of result

The graph below shows the rate of score 4 from different group:

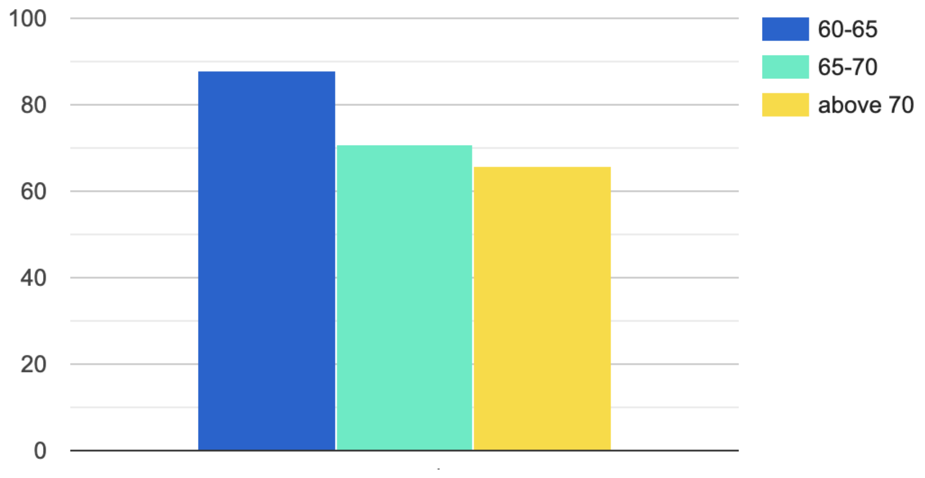


Figure 13. The rate of score 4 from different group

As elaborated above, we designed two Experiments to prove our effective solution has a high accuracy of prediction results. Experiments 1 test AI prediction results with test cases, Experiments 2 shows user surveys which can help us know if the app is useful or not. Both of the two Experiments have enough test cases with reasonable testing group and stable diversity, which can be used to prove our floating app can actually help senor people learn how to use smart iPhone.

5. Related Work

Peacock and Kunemund theorized that senior citizens’ lack of participation in internet technology is primarily due to “prive access possibilities, motivational indifference, and deficient knowledge.” They state that efforts to close the digital age gap are crucial [1]. This paper was written in 2007 when smartphones and the internet did not have as much impact as now. By 2021, however, more than 85 percent of adult Americans will use a smartphone. While the financial-caused digital divide can be lessened with a growing economy and advancing technology, the age-caused digital divide is much harder to resolve. The older generation tends to have less experience with technology and lower education, making it harder for them to comprehend the logistics behind

APPs on mobile phones. This makes our proposal significant because not only does our APP provide tutorials, it can also teach seniors where and how to perform functions on a display like everyone else's.

Hsiao, S. et al proposed a user interface that uses the Kinect sensor [2]. The interface can be operated by hand gestures. This method will make it very easy for seniors to perform basic functions like close, swipe, return, etc, while our proposed method will make it easier to navigate more complicated functions. The method will also make it easier for seniors with neural disorders, because they will be able to perform a function without looking for that one tiny button on the small screen. On the other hand, our proposed method has an advantage to help seniors understand technology on a cognitive level.

Werner et al. suggests an iPad model designed for seniors [3]. While our proposed method provides tutorials to regular APPs, this model suggests developments of tablet based applications. The advantage of this method is that seniors will be able to use their mobile devices without any external aid. Using a tablet will also make it easier for seniors with presbyopia or other eye problems. In contrast, our proposed method helps seniors learn and adapt to APPs, which could potentially help them use their normal APPs without any aid.

6. CONCLUSIONS

Tutorial4Seniors is an intelligent mobile floating application that aids seniors in using the smartphone.

This application is primarily limited by its database. The functions users wish to perform would have to be included in our database for us to provide tutorials. As a result, we might not be able to provide for all the functions a user needs, or stay up to date with all APPs. We also do not have perfect accuracy in detecting which APP the user is on, for some APPs look alike and are hard to distinguish from one another. We were only able to design our APP for android devices because of iOS restrictions.

For future works, we hope to develop tutorials with artificial intelligence. We also wish to come up with better ways to identify the APP the user is on, either through adding more screenshots to our APP database, or initiating communications with the phone. In addition, we would like to figure out ways to implement our program on iOS.

REFERENCES

[1] Peacock, S. E., & Künemund, H. (2007). Senior citizens and Internet technology. European journal of ageing, 4(4), 191-200.

[2] Hsiao, S. W., Lee, C. H., Yang, M. H., & Chen, R. Q. (2017). User interface based on natural interaction design for seniors. Computers in Human Behavior, 75, 147-159.

[3] Werner, F., Werner, K., & Oberzaucher, J. (2012). Tablets for seniors – an evaluation of a current model (iPad). In Ambient assisted living (pp. 177-184). Springer, Berlin, Heidelberg.

[4] Young, Peg. "Technological growth curves: a competition of forecasting models." Technological forecasting and social change 44.4 (1993): 375-389.

[5] Mouritz, Adrian P. Introduction to aerospace materials. Elsevier, 2012.

[6] Holtz, Peter, Nicole Kronberger, and Wolfgang Wagner. "Analyzing internet forums." Journal of Media Psychology (2012).

[7] Van Dijk, Jan AGM. "Digital divide research, achievements and shortcomings." Poetics 34.4-5 (2006): 221-235.

[8] Morens, David M., Gregory K. Folkers, and Anthony S. Fauci. "What is a pandemic?." (2009): 1018-1021.

[9] van Der Aalst, Wil MP, et al. "Workflow patterns." Distributed and parallel databases 14.1 (2003): 5-51.
[10] Copi, Irving M., Carl Cohen, and Victor Rodych. Introduction to logic. Routledge, 2018.
[11] kenine-Moller, Tomas, Eric Haines, and Naty Hoffman. Real-time rendering. AK Peters/crc Press, 2019.
[12] Castano, Silvana, et al. Database Securi. Addison—Wesley, 1995.
[13] Fricker, Elizabeth. "Critical notice." Mind 104.414 (1995): 393-411.
[14] Carr, Brian, and Ira P. Goldstein. Overlays: A theory of modelling for computer aided instruction. MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1977.
[15] Model, A. Dag Based Decentralized Oracle. "Implementation and Evaluation." Blockchain and Applications: 3rd International Congress. Springer Nature, 2004.