**Project Final Report**

Xurui Zhong

zhong87@purdue.edu

Shuai Shao

shao85@purdue.edu

Andrea Nguyen

nguye174@purdue.edu

**ABSTRACT**

In this paper, we will describe the project we made for CNIT 355: Waifu2x, which is a image super-resolution for anime-style art using deep convolutional neural networks, while also supporting photos [1].

**Keywords**Android Studio; waifu2x; High resolution image; Deep learning

**1.** **INTRODUCTION  
1.1 Topic Introduction**Waifu2x is a sophisticated tool that uses deep convolutional neural networks, or machine learning for short [2]. It is a program used for upscaling images and removing JPG noise artifacts using artificial neural networks. This program is intended for use with anime art, though photos are supported.

For our project, we incorporated Waifu2x into a server that would connect to an app of our making so users can upscale and remove JPG noise artifacts on their mobile devices.

**1.2 Motivation**Are you tired of low quality anime images and wallpapers, wanting to glorify them? I believe many people have met images they love, but the low resolutions kept them from saving the images. Essentially, higher resolutions come with images which are clear and detailed, while lower ones are blurry and undefined [3]. We want to replace the low resolution photos with better ones, so we came up with the idea to make this application.

**1.3 Objectives**Our objective of this project is to help people find high resolution photos they want by just simply turning on our application and selecting their images as well as converting styles. They don’t need to struggle asking people online or trying to find the high resolution images from nowhere. Below is a comparison of the original image and the image after converting [4].



**Figure 1. Comparison of images**

**2.** **LITERATURE REVIEWS**

Waifu2x Photo zoom Pro was inspired by waifu2x [1]. Unlike Photoshop or any other photo editor, Waifu Photo zoom Pro uses convolutional neural network to process image and provides higher resolution image than normal photo editor. Although there are many existing websites that provide image enlargement services, few of them are made into phone apps. According to the app download and usage statistics in 2018 [5], the usage of apps is keep increasing so it will be profitable for us to develop an app that can be used to photo zoom 2D images. Different from other photo enhance mobile apps, our app can provide more upscale settings such as 4x and 8x. Each user can upload unlimited images every day and 20 images can be up scaled to 8x. To get the unlimited 8x upscaling, users need to upgrade their account. We will also decorate our app to attract more people to use it.

# 

# 

# **3.** **APPROACHES AND METHODS**

## **3.1** **Your Ideas**

First, we create Linux GPU server from Google cloud. Then we install waifu2x and dependencies from GitHub. After we deploy waifu2x, we will design an Android app that can show and upload the images in our android phones. The images will be uploaded to the server, where they would be converted into high resolution images and then transferred back. The users will receive the high resolution images and see the difference between the original images and the converted ones.

## **3.2** **Technical Specifications**

Waifu Photo zoom Pro would be able to convert blurred comic pictures to high resolution pictures. It requires Android 7.0 mobile phone or above. It requires Storage and Network permissions. The resolution should not exceed 720x1280 pixels otherwise the app will look distorted. Users are expected to have strong internet connectivity since the converted image usually have very large size. The processed images would be deleted from the server within 2 days.

Our potential users are people interested in illustration art and willing to get high quality images. The expected age of our users range from 16-28.

This app will provide many benefits to users compared to those who use other apps. First, the app is 100% free and it will provide x8 amplify while other apps only provides x2 amplify. What's more, this app will use NVIDIA Tesla k80, so users won’t be expected to wait too long while converting the images.

However, this app contains several defects that could be improved. First, since the GPU server has limited computing power, only one user can use this app at a time. In addition, this app can only be used to process 2D images such as manga or illustration images.

## **3.3** **Building the Server**

In order to implement Waifu2x server, first we needed to create a GPU server in Google cloud. We chose 1 CPU and 1 gtx k80 GPU with 20 Gigabytes of disk. After that we configured the server to have sftp function so the app could upload and download images by using the server. Next, we installed CUDA and Torch7 to setup the Waifu2x environment. Then we used GitHub to download and implement Waifu2x. Finally, we wrote a bash script to automatically convert the input images into higher resolution images. In that way, the app could request to download the converted images into the local storage.

**3.4 Creating the App**

To begin creating our app, we first needed to know what it would do. We decided that we would need a layout where the user can choose an image to convert, a layout where the user could see the converted image and download it, a layout to view past converted images, and a layout to display basic information about us and our app. Once that was established, we started designing our UI.

After the design of the UI was finished and created on Android Studio, we began working on code that did not rely on our server. Those codes were selecting an image, downloading and sharing the converted image, displaying a library, and coding buttons that allowed navigation between the activities.

When coding the layout that would display the converted image, we decided to also allow the user to view the original image for comparison. The user could switch between viewing the original image and the converted image by either clicking a button or swiping. This was accomplished by using a Frame Layout with two image views overlapping. The gestures for swiping were added after CNIT 355 Lab 13.

For sharing, we first thought it would be simple. After further research, we discovered that that was only the case with APIs lower than 24. After API level 24, sharing images required a file provider. Thankfully this was not too hard to figure out.

Coding the Download button created a slight issue. Anywhere we looked, we were suggested that the best way to download an image was to create a bitmap and compress the image into that bitmap. This is an unwanted method, as compressing the image to a bitmap would lower image quality. With our goal being to help users create high-quality images, we needed to search for a new method. One such method was found on StackOverflow, using an inputStream to read the image by kilobytes and writing to the destination file. Downloading would also require using a MediaScanner to add the downloaded image to the media library.

For the in-app library, we found it was best to use a SQLite Database to get the past images. The method to create a SQLite database was easily found in Android Studio Development Essentials. The plan was to store only an id number and the file path that held the image.

After creating the SQLite database, a database handler was created, adding methods to add, delete, update, and find rows. Adding rows would be called when the layout to display the user’s converted image is called. Updating would happen alongside adding, incrementing the ids of existing rows by one. Storing would use a push method, meaning the last row entered would be the first row in the list. As only sixteen image views existed in the library, only sixteen rows were needed in the table. With a push method, we could easily check if the number of rows exceeded sixteen and delete the oldest rows along with the cache files that were associated with the deleted rows. The method to find rows would be called at the onCreate method of the library, where the file paths in the rows would be taken and used to generate drawables to put into the image views.

With the library done, we wanted to add one more feature to the library: displaying an enlarged image of the image view we tapped. This was a simple process, creating a new activity that would hold an image view that would take up the entire phone screen and a back button that would kill the activity.

Once the code for this was completed and tested to be working fine, we moved on to connecting our app to our server, allowing it to upload images to convert to the server and receive the converted images back.

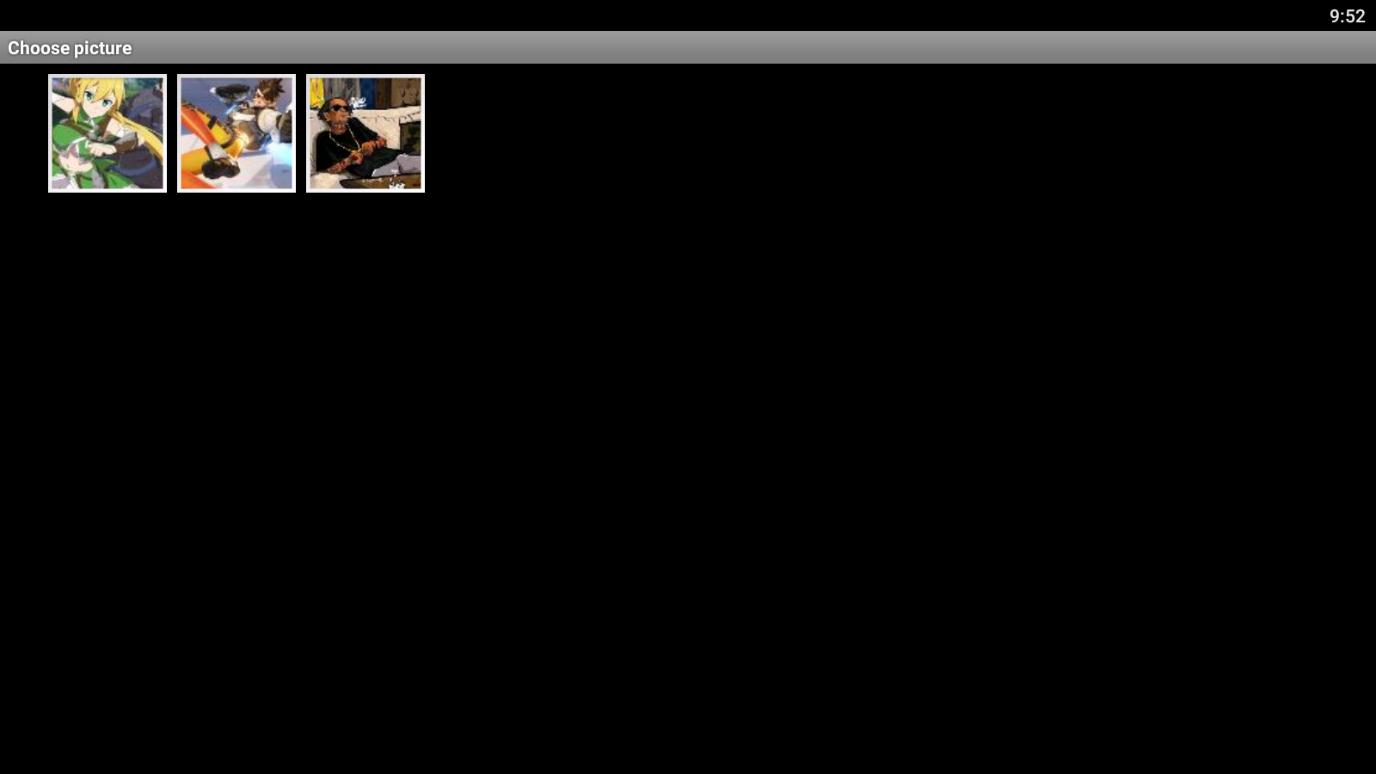
**3.5 Connecting the App and the Server**

To connect the app and the server, we decided to use sftp transport protocol to transfer images. The reason we didn’t use https protocol is due to the fact that it needs HTTPS certificate. Also, we didn’t use tfp due to the security concern. We added jsch-0.1.52.jar package to the library since it contains the necessary sftp connection tools. After that, we wrote the code to open the image library and selected the image that users wanted to upload. Then we put the image into imageview for users to see. At the same time we wanted to get the local path and name of the image so we could upload the image into server. Also, we need the names to download the images we want. In order to do that, first we got the uri of the image when user selected the image. Then we used cursor method to convert the format of uri to string. Finally, when we clicked the upload button, the image that was selected would upload to server. When we clicked download button, the converted image would be download based on the name of the original image.

While in the process of connecting the app and the server, we learned that we could download the image from the server directly to the created image directory on the external storage. This removed the need of a download button. Instead of a download button, we have replaced it with a “Download and Share” button. The new button will download the image from the server and display it on the image view. There is a delay of a few seconds, so the user may have to click more than once for their converted image to appear.

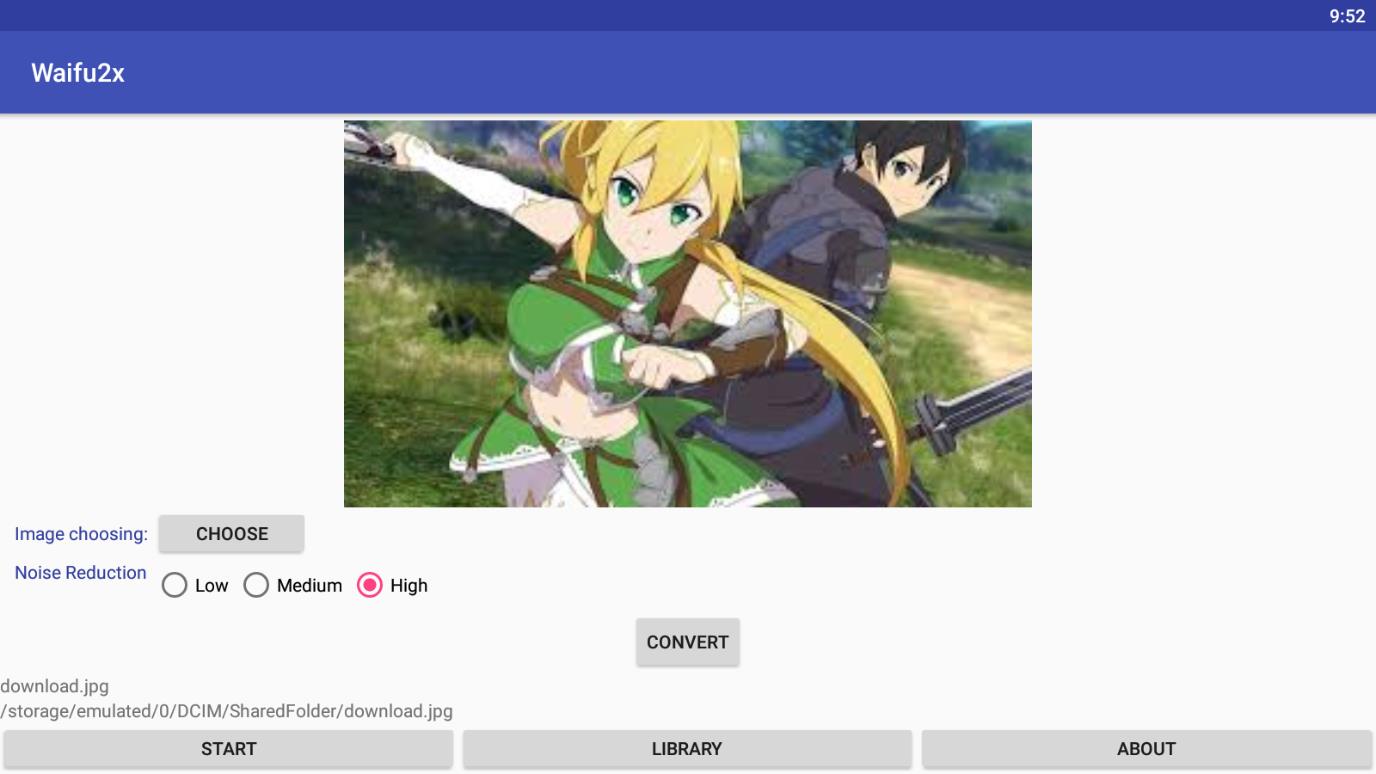
**4. ANALYSIS AND RESULTS  
4.1 Analysis**This app can be successful run in either physical or virtual devices with SDK 24 or upper. It can fit different sizes of phone screen. On the first launch of the app, the app will ask for external storage access permission.

If the user accepts the request, the app will open the image library and select the image they want.

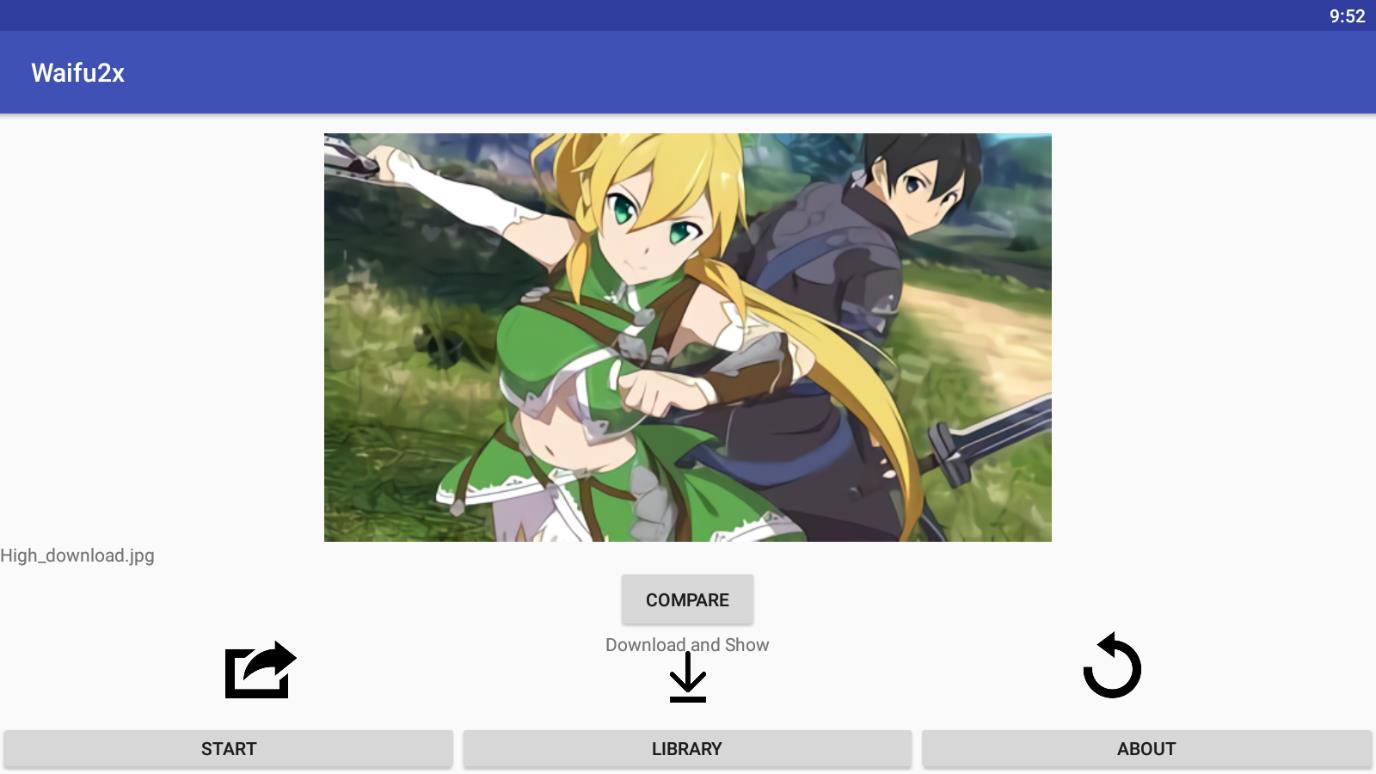


After the user select the image, the user can see the uploaded image in the imageview.

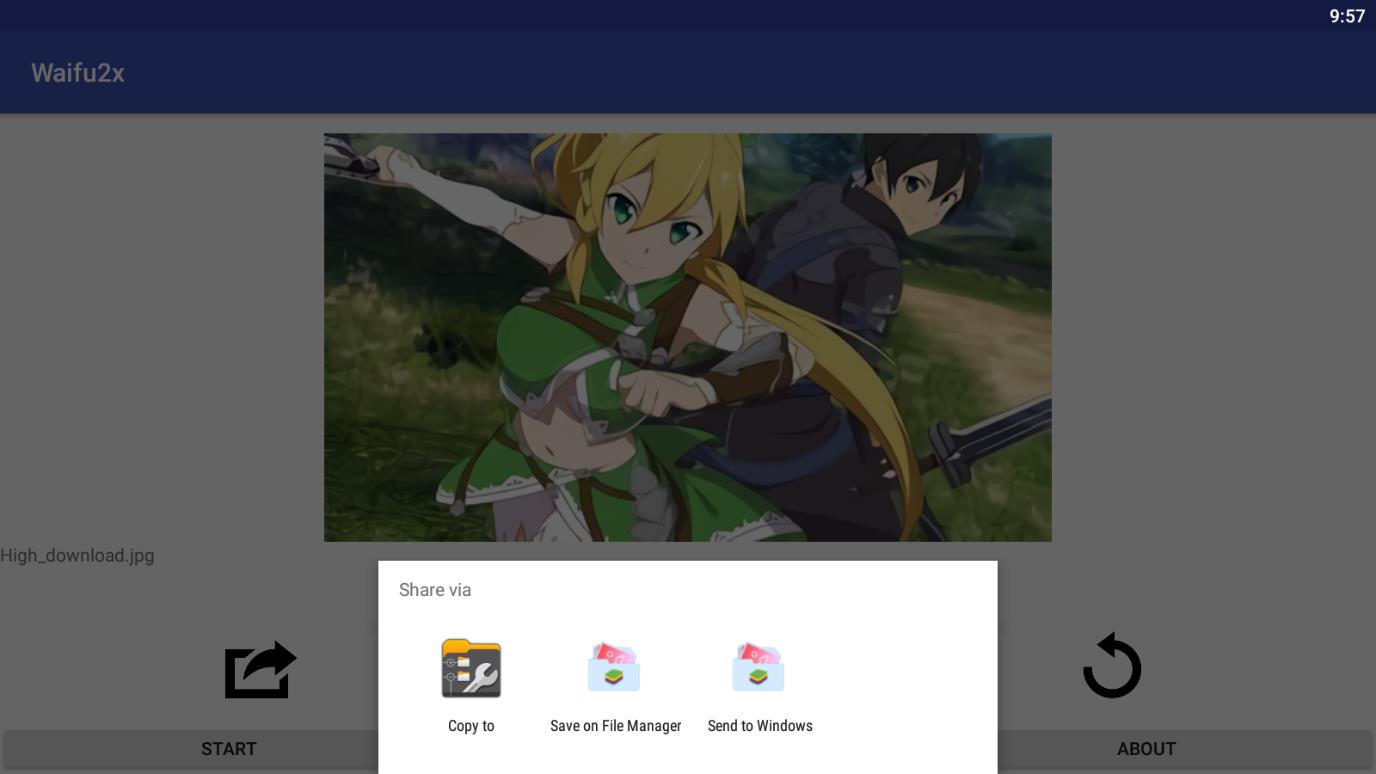
Then the user needs to select the noise reduction level. Once it is done, the user can click convert button to upload the image to server. The convert button will also open an new activity that can show the converted image.



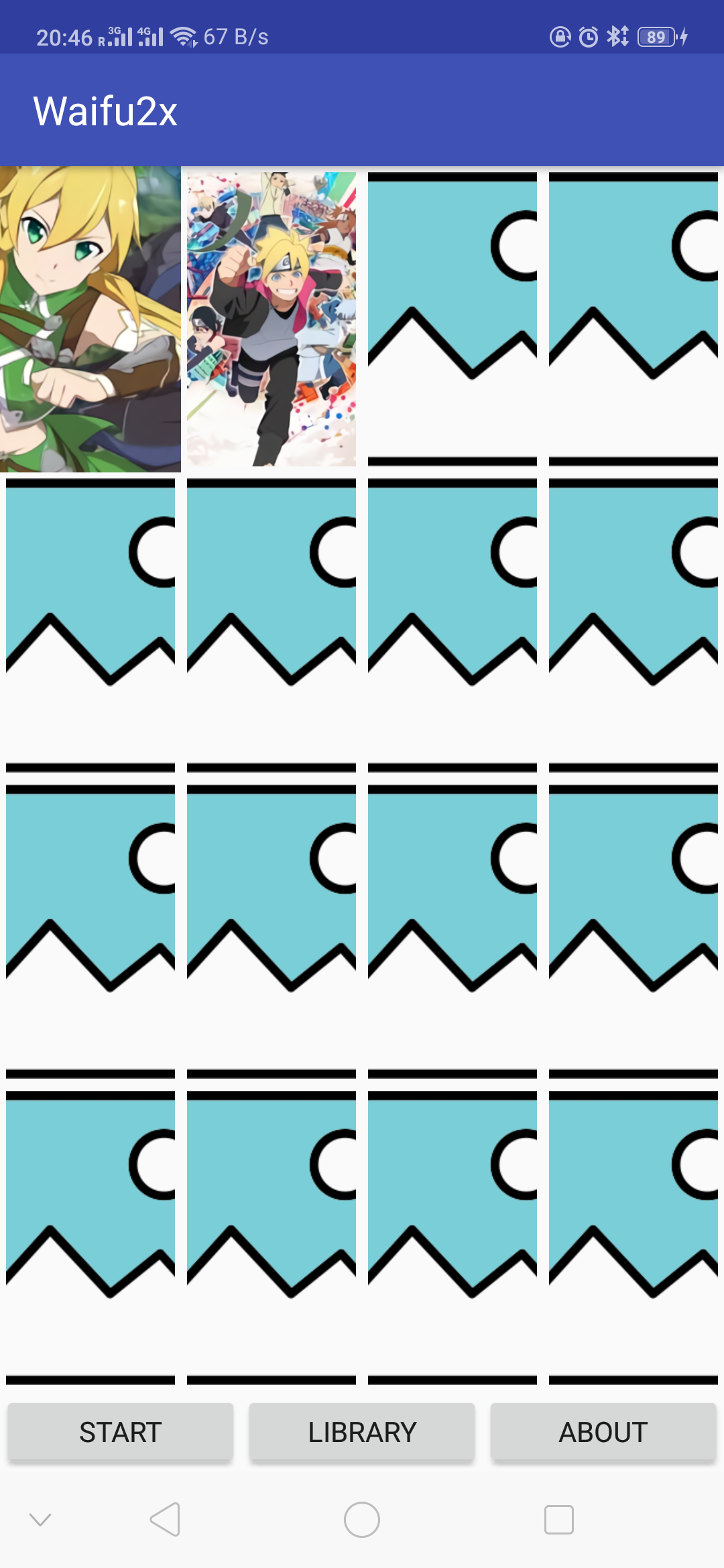
When the user clicks the download icon, if the image is still converting, it will display ‘still converting, please wait’. Otherwise the new image would be shown on the screen and saved in /Download/Waifu2x\_Images/ folder.



When clicking the share icon, the user would be able to share the converted image through text, email, etc. When clicking the undo icon, the user would go back to the first activity to select other images to convert.



In the “library” tab, all the recent converted images are shown here. The user can see the original image by clicking each thumbnail.



**4.2 Results**The user first chooses the image willing to convert by clicking on the choose button. Then he/she can choose low, medium, or high for the noise reduction. The image name and path would be displayed, then clicking on the convert button would upload the image to our server.

When the user successfully uploads the image and jumps to the new activity involving the converted image, he/she has to wait a while for the image to be converted. When clicking the download icon, the image would be displayed and downloaded. Clicking the share icon would allow the user to share the image.

Clicking on the library button would jump to a library storing the past converted images. An enlarged version of the image would be displayed when tapping on the small icons.

**5. CONCLUSION AND FUTURE WORKS  
5.1 Conclusion**After implementing the methods we previously mentioned, we successfully implemented Waifu2x to a mobile app, allowing users to upload, convert, download, and sharing images from their mobile devices.

**5.2 Future Works**Though our app is done, there are still many places where we could make improvements. Something we thought to add was the ability to crop a selected image and convert the cropped image. This one is not yet implemented due to the fact that the crop method we chose did not work on phones lacking a specific app, and, as this was proposed late in the project, we did not have enough time to find and test a new cropping method.

There were some features that we had intended to include but couldn’t due to lack of time. These are allowing the user to choose the style of the image they had (whether it was an artwork or a photo), and to choose the level of upscaling they wanted (whether they wanted the image to be two times, four times, or eight times bigger).

Again, if we had time, we could have also edited our layout to look nicer and create custom layouts. This would not achieve much, but it would be pleasing to the users’ eyes.

# **6. INDIVIDUAL CONTRIBUTIONS**

**6.1 Xurui Zhong**Set up sftp and deep learning server. Wrote android sftp codes. Wrote bash script to automatically execute Waifu2x code. Recorded the app demo. Contributed to the report.

**6.2 Shuai Shao**Designed the application user interface. Wrote code for choosing, uploading, and downloading images.

**6.3 Andrea Nguyen**Set up SQLite database. Wrote code for sharing, library, and displaying from library.

# **7.** **REFERENCES**

[1] Nagadomi. (n.d.). nagadomi/waifu2x. Retrieved October 21, 2018, from https://github.com/nagadomi/waifu2x

[2] Atolstoy, A. (2018a, October 2). Even better image upscaling with Waifu2x - Fedora Magazine. Retrieved October 21, 2018, from https://fedoramagazine.org/better-image-upscaling-waifu2x/

[3] Cherepanya, M.(2009, March 6). High vs. low resolution. Retrieved October 21, 2018, from https://www.inverseparadox.com/2009/03/high-vs-low-resolution/

[4] Candyland (2016, May 24). Overwatch – PC Low vs. Medium vs. Max Graphics Comparison. Retrieved October 21, 2018, from https://www.youtube.com/watch?v=uojZp-kHE4k

[5] Dogtiev, A. (2018, October 8). App Download and Usage Statistics (2018). Retrieved October 21, 2018, from http://www.businessofapps.com/data/app-statistics/