Bilkent University

Department of Computer Engineering

Senior Design Project

Project name: Upfix

Final Report

Meryem Efe, Hamza Pehlivan, Özge Yaşayan, Hazal Aksu, Rabia Nur Önal

Supervisor: Uğur Güdükbay

Table of Contents

Final Report

Project Short-Name: Upfix

1. Introduction

Gaming is undoubtedly a significant source of entertainment for many people. The gaming industry has been growing rapidly, notably with the ever-rising popularity of video games as well as online versions of classic games such as chess.

The number of people who play games is increasing thanks to the popular games and online platforms [1, 2]. Consequently, the number of people who consume video contents related to gaming is increasing simultaneously as well. As a result, a new sub-industry named Gaming Video Content (GVC) has emerged. According to data collected in 2017, the number of GVC viewers has reached 666 million globally [2]. In the GVC industry, Twitch is the leading platform, accounting for 54% of the gaming video content platform revenue in 2017 with Youtube following right behind [3]. Even though these platforms are widely popular, users can still face issues in watching game videos in high quality which is mostly due to limited internet connection. Since watching videos in high resolution consumes more data, users opt to watch videos in low resolution even though it may not be desired. Furthermore, the same problem occurs when people want to upload a game video on the internet. Therefore, all these issues compel us to pay attention to the need of improved video upscaling techniques.

It should be noted that big game companies are trying to provide high quality game videos for their viewers. For example, Valve provides GOTV to stream Counter Strike tournaments and DotaTV for Dota2 tournaments [4, 5]. Their approach requires geographically distributed proxy servers, which small game companies may not prefer. Furthermore, in general, tournaments and important events are streamed on the network with this approach. This means viewers who want to watch gameplays other than predetermined contents may not be able to obtain high quality videos.

The purpose of our Senior Design Project is to design and implement an application that provides higher quality game videos by upscaling. In this project, we provided a platform for chess, go, Age of Empires [6] and Among Us [7] game videos. Using our application, games of relatively small companies, which do not prefer investing money in proxy servers, can be watched with high quality. Moreover, viewers who want to watch their favorite streamer - some chess or go player for example- with high quality can use our application.

2. Requirement Details

2.1. Functional Requirements

- The users can upload a video from the local file system to be upscaled.
- The application upscales the uploaded video by using GPU or CPU offline. The usage of GPU or CPU is determined by the program according to the system properties.
- The user can watch the upscaled video.
- The user can save the upscaled video to the local file system.

2.2. Non-functional Requirements

2.2.1. Performance

- The user is able to start watching the upscaled video in at most 3 minutes.
- The response time for the desktop application does not exceed 2 seconds.
- Even though the total upscaling time depends on the length of the video, the upscaling time of one frame is less than 0.01 second.

2.2.2. Usability

- The system is easy enough to be used by anyone of all ages with basic computer skills.
- The system comes with an explanatory user's manual, that can be viewed at the end of this report.

2.2.3. Scalability

- The desktop application is able to upscale one video at a time.

2.2.4. Extensibility

- The system is designed in a way that makes it easy to integrate new upscaling algorithms, improved models, and support for more games in the future.
- The system can be extended in a way that it upscales not only the game videos but also low-quality photographs. In this way, the system can use high-qualified but slow models as well. The slow models are not preferred in video upscaling but they can be preferred in image upscaling.

2.2.5. Security

- The system does not disclose the data of its users to third parties.

3. Final Architecture and Design Details

3.1. Overview

In this part, we are going to explain the subsystem decomposition which describes the subsystem structure of our system in detail with diagrams. After, we will provide a hardware / software mapping of the system. Then, we will outline persistent data management, access control and security, global service control, and boundary conditions.

3.2. Subsystem Decomposition

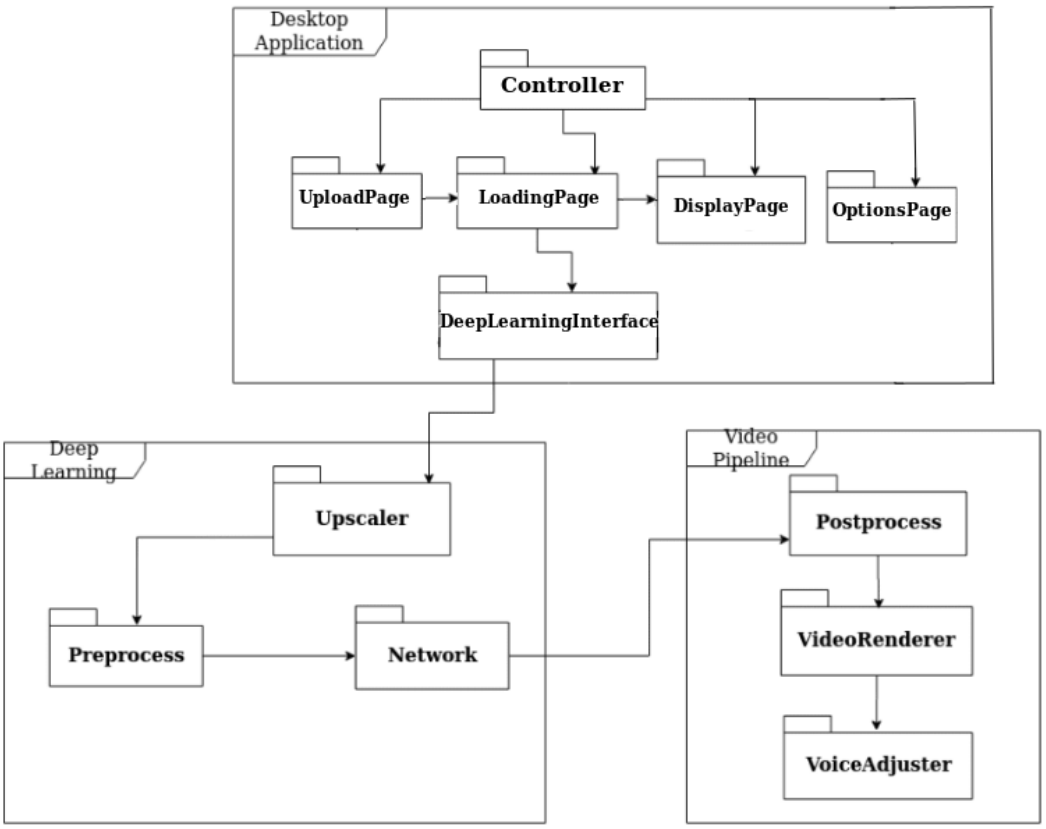


Figure 1. Subsystem Decomposition Diagram

3.3. Hardware / Software Mapping

The application behaves differently if the user has a CUDA enabled GPU. If the user doesn't have any GPU, the upscaling process is executed in the CPU. However, if the user has a CUDA enabled GPU, the upscaling model works on the GPU. The GPU is used to speed up inference and video rendering time.

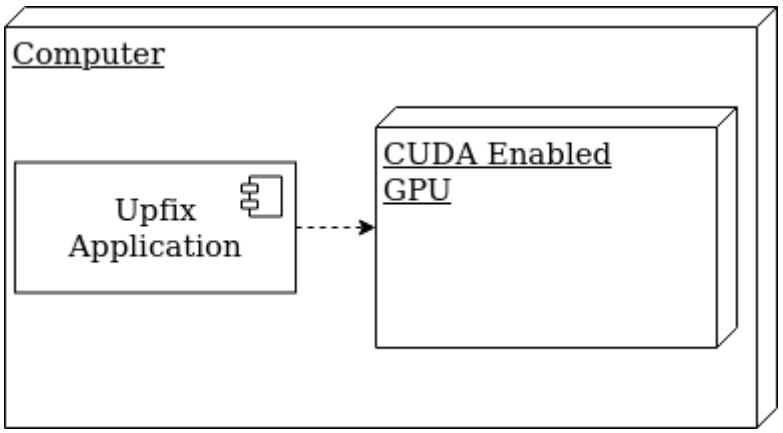


Figure 2. Hardware/Software Mapping Diagram

3.4. Persistent Data Management

In the Upfix desktop application, models need to be persistent. We used a filesystem to persist models for the desktop application. Since our aim was to upscale game videos by using local resources, we didn't use any servers. Instead, we stored the models in the user's localhost. Additionally, in the desktop application, we provided user settings options. In the settings, users can choose the default save location and they can activate the auto-save option.

4. Development / Implementation Details

In this section, the development and implementation details of the model and application part will be explained separately.

4.1. Model

At the beginning of the development of the model, the model team determined the strategy as follows:

- Plan A: Start with a simple CNN model on chess video. According to the results, continue with RNN and GAN models.

- Plan B: If the response time is too long, apply transfer learning on OpenCV pre-trained models.

In this section, we are going to explain the model development process in detail.

4.1.1. Data Collection and Preprocessing

Regardless of the plan we applied, we needed a dataset because there was no well-prepared dataset which consisted of any game we chose. Therefore, we have collected game videos from different streamers, channels, and platforms. Since we started training on chess videos, we first collect the chess dataset. In data collecting and preprocessing, we followed these steps:

1. Chess videos were collected.
    1.1. One frame was extracted in each 10 seconds.
    1.2. Unnecessary parts were cropped.
    1.3. Some frames were eliminated to reduce dataset size.
    1.4. All the frames were shuffled and renamed accordingly.
2. Among Us videos were collected.
    2.1. One frame was extracted in each 10 seconds.
    2.2. Unnecessary parts were cropped.
    2.3. Some frames were eliminated to reduce dataset size.
    2.4. All the frames were shuffled and renamed accordingly.
3. Age of Empires videos were collected.
    3.1. One frame was extracted in each 10 seconds.
    3.2. Unnecessary parts were cropped.
    3.3. Some frames were eliminated to reduce dataset size.
    3.4. All the frames were shuffled and renamed accordingly.
4. To create a single dataset, all these three datasets were merged, shuffled and renamed accordingly.

4.1.2. First Model Trial

Since we dealt with both low and high quality images in training, we predicted that we could have resource problems. Therefore, we started with a simpler

model, CNN model instead of GAN model. Based on the research [8], we created a structure as in Figure 1. Even though we created the structure based on the research and started to train, we encountered some resource problems and we switched to plan B.

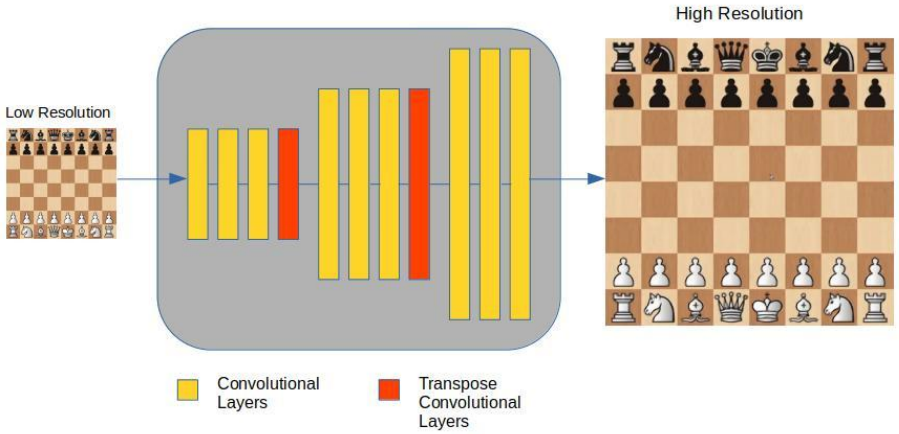


Figure 3. Initial CNN Structure

4.1.3. Existing OpenCV Models

When we gave up training our models from scratch, we did research about the current works on super resolution:

OpenCV is one of the open source libraries which provides deep learning based pre-trained models for super resolution [9]. There are currently 4 different super resolution models provided by OpenCV. These are EDSR, ESPCN, FSRCNN, LapSRN [10]. All these models differ in accuracy, size, and speed. The best performing, biggest, slowest model is EDSR. ESPCN and DSRCNN are smaller and faster models as well as they can do real-time video upscaling. Lastly, LapSRN is a medium sized model and can upscale by a factor of 8 while the other models can upscale by a scale of 4 at most.

We chose ESPCN to apply transfer learning and train on our dataset.

4.1.4. Training ESPCN [11]

We trained ESPCN network with our dataset. It is one of the fastest models that can do real time upscaling. It is a CNN based model meaning that it can

be used with any image size as long as the image and the model fit in the GPU.

Model:

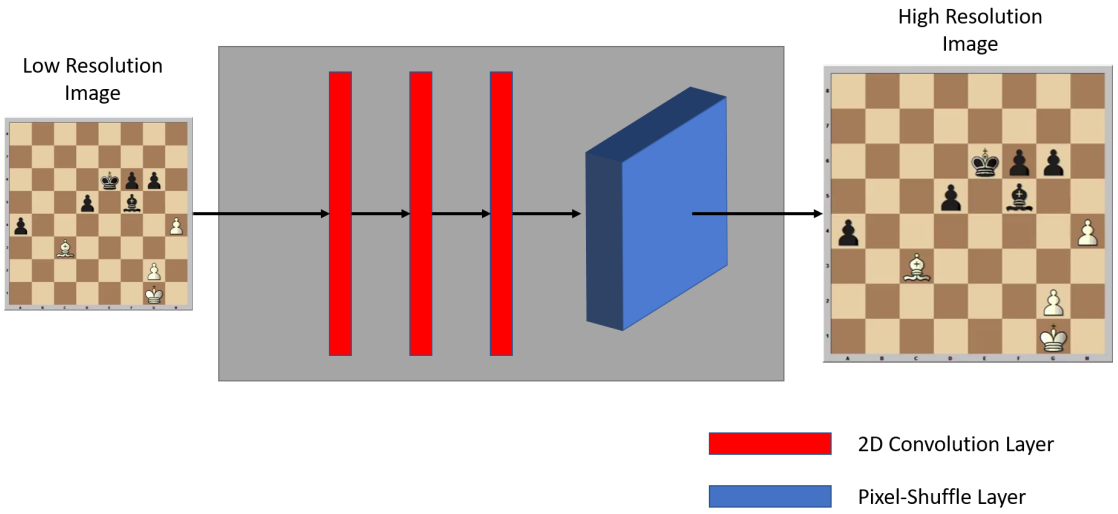


Figure 4. ESPCN Network

The network consists of 3 convolutional layers followed by one pixel-shuffling layer. Similar to transpose convolutional layers, pixel-shuffling layer is used to increase width and height. For training, a single GeForce GTX 1650 Nvidia GPU is used. Training time is usually between 3-4 hours depending on the number of epochs. Adam optimizer with default parameters is used.

Loss Function: In addition to the mean square error loss function, we implemented another loss function, which makes the network focus on edge preservation. We refer to this loss function as the “Sobel Loss Function” in this report. So the total loss is:

$$L = \frac{1}{xy}\sum_{x}\sum_{y}[HR_{orig}(x,y) - HR_{out}(x,y)]^2 + W\frac{1}{xy}\sum_{x}\sum_{y}[SOBEL_{orig}(x,y) - SOBEL_{out}(x,y)]^2$$

where,

x and y: Pixel locations,

HR_{orig}: Original high resolution image,

HR_{out}: Output of the network,

W: Weight for the second loss function,

$SOBEL_{orig}$: Gradient values of HR_{orig} after Sobel Edge Detection,

$SOBEL_{out}$: Gradient values of HR_{out} after Sobel Edge Detection.

Experiments:

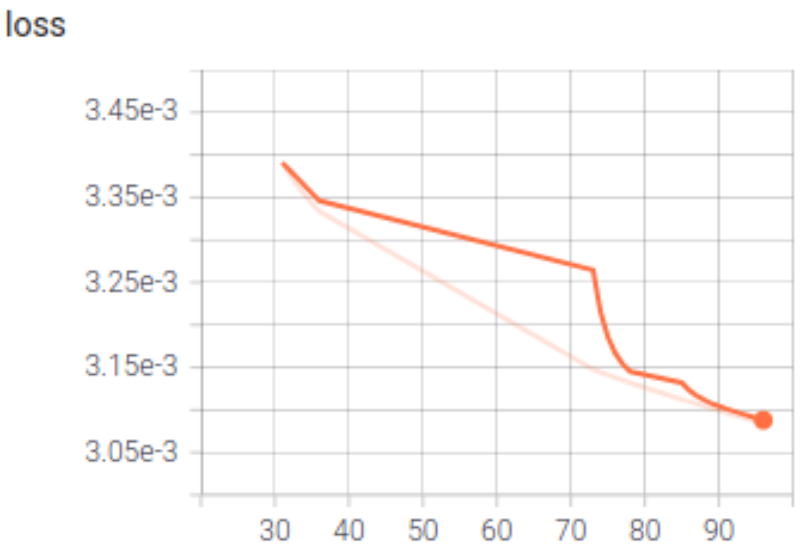
For the following table low resolution image size is 17 and batch size is 256. Created high resolution image size is 51, which means upscaling factor is 3.

Shuffle Dataset at Each Epoch	Learning Rate	Sobel Loss Function Weight	Final Loss	Final PSNR Score
True	0.01	0	0.003454613	29.35945436
True	0.01	0.1	0.003964783	28.00200416
False	0.001	1	0.002773533	31.84912184
False	0.0001	1	0.003011706	32.43387493
False	**0.001**	**2**	**0.002589095**	**33.03260336**

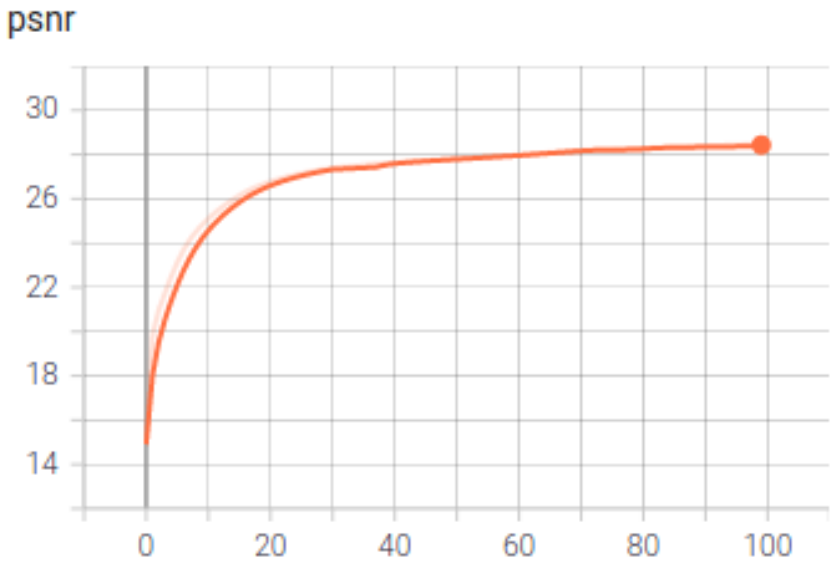
Table 1. Experiment Results

We also conducted experiments with different initial image size, batch size and learning rates - like 0.003- ;however, none of them were good enough.

Graphs:



Graph 1. Loss vs Epoch Number



Graph 2. PSNR Score vs Epoch Number

The results are better than the traditional upsampling:

Figure 5. Left image is ESPCN network, right image is bicubic interpolation. We observed that the edges are smoother.

4.1.5. SRGAN [12]

Even though we got better results from traditional upscaling methods, we wanted to see whether we can obtain more good-looking results. Therefore, we decided to use SRGAN, which is a Generative Adversarial Neural Network. When we deployed this network we realized that upscaling of an image takes 3 seconds. This number is not suitable for real time upscaling;

therefore we modified the network. We decreased the number of residual blocks from 12 to 4, filter sizes from 64 to 32. Lastly, we converted regular residual blocks to inverse residual blocks, which increases time efficiency. We managed to decrease elapsed time to 1 second; however, this also was not good enough. Finally, we found a fast SRGAN implementation on the internet, which uses inverse residual blocks, and decreased the number of filters together with some additional modifications. These modifications include reducing the size of the Discriminator and some other parts of the Generator. Because this code was more optimized than our modified network, we decided to use it.

Models:

Generator:

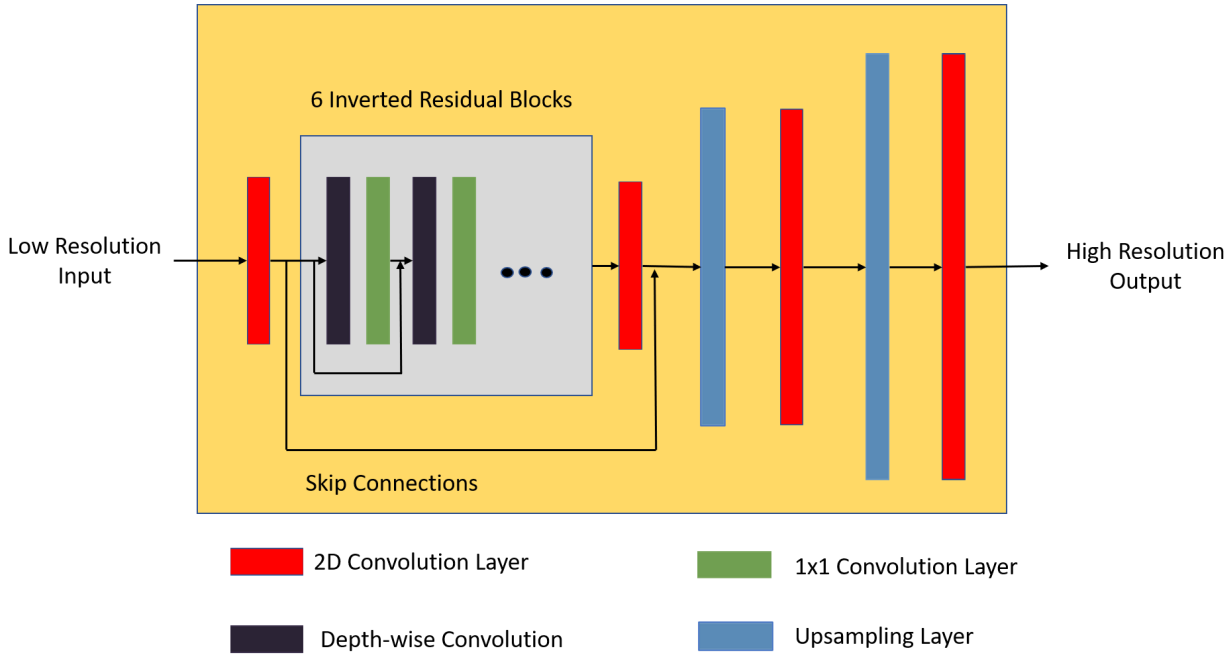


Figure 6. Modified SRGAN Generator

Discriminator:

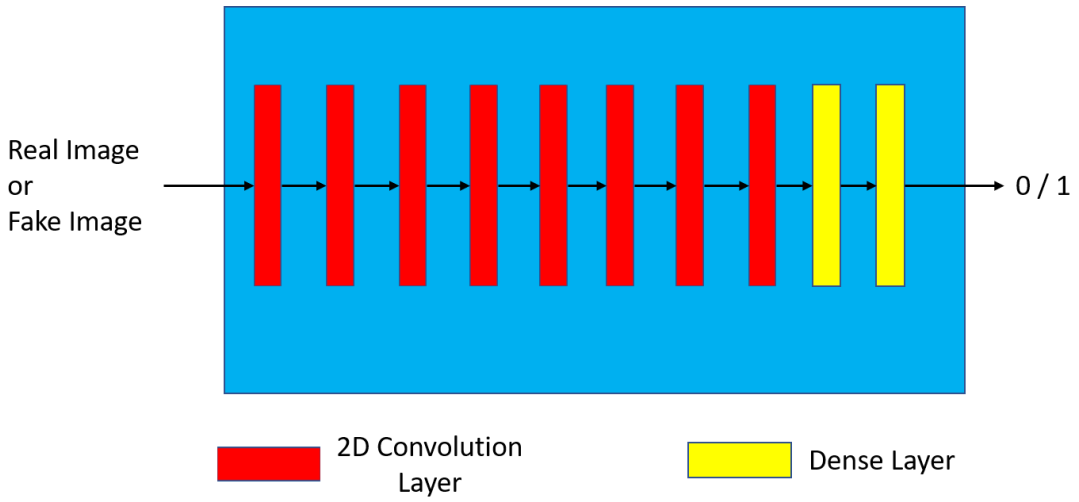


Figure 7. Modified SRGAN Discriminator

The network is trained on a single GeForce GTX 1650 Nvidia GPU. The learning rates are 0.0001 and 0.0005 for generator and discriminator, respectively. Learning rate scheduling with 0.1 exponential decay is used. Low resolution image size is taken as 32x32 and the generator produces 128x128 high resolution images, meaning upscale factor is 4. Because the generator has a CNN architecture, this setup can be applied to any size of images. Adam optimizer with default parameters is used. The model is trained for 1000 epochs.

Loss Functions:

Generator Loss: Generator loss is the summation of adversarial, content and mean squared error losses.

$G = 10^{-3}ADVERSARIAL + CONTENT + MSE$

Adversarial loss: Adversarial loss is the standard method for generator adversarial neural networks. The generator tries to fool the discriminator by producing realistic images. It tries to decrease this function.

$$\frac{1}{K}\sum_{i=1}^{K}\log(1-D(\hat{I}_i))+\log(D(I_i))$$

D(Î) : The probability that the discriminator thinks the fake image (output of generator) is real.

D(I) : The probability that the discriminator thinks the real image is real.

Content Loss: Content loss is calculated using a pretrained VGG network. Both fake and real images pass through VGG. The tensor values from a specific layer are extracted for both images. The difference between these values gives the content loss.

$$\frac{1}{K}\sum_{i=1}^{K}\sum_{p=0}^{P-1}\frac{1}{h_p w_p c_p}\|\Phi_p(I_i)-\Phi_p(\hat{I}_i)\|_1$$

I_i : Real Image
$\hat{I}$: Fake Image
h,w,c : Dimensions of the Current Layer
P : Layer Number

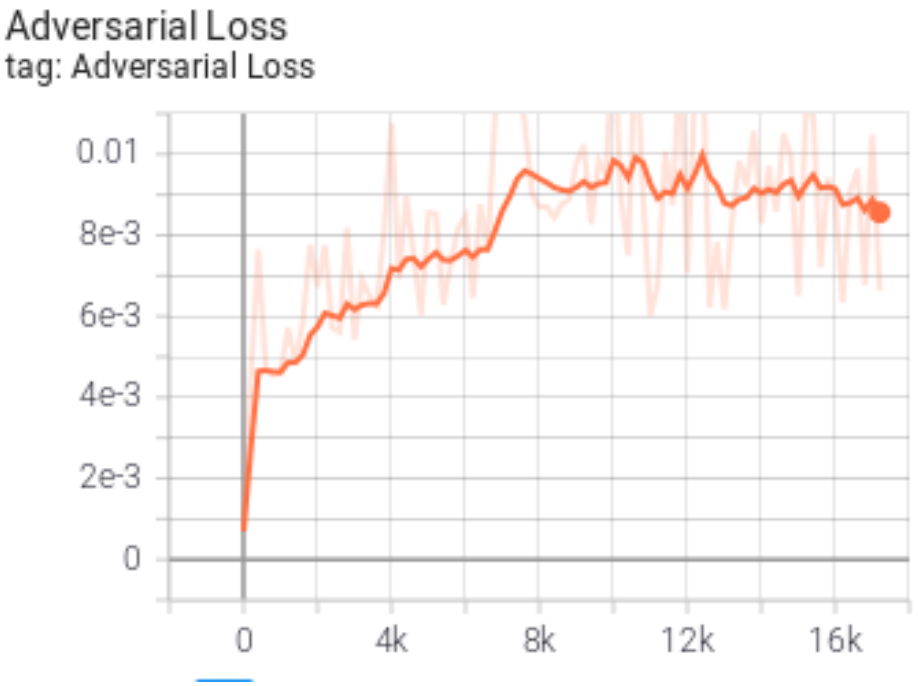
Mean Square Error: This function is the same function that we used in CNN network.

$$L=\frac{1}{xy}\sum_x\sum_y[HR_{orig}(x,y)-HR_{out}(x,y)]^2$$

Discriminator Loss: The discriminator tries to find out if a given image is real. It tries to increase standard adversarial loss function.

$$\frac{1}{K}\sum_{i=1}^{K}\log(1-D(\hat{I}_i))+\log(D(I_i))$$

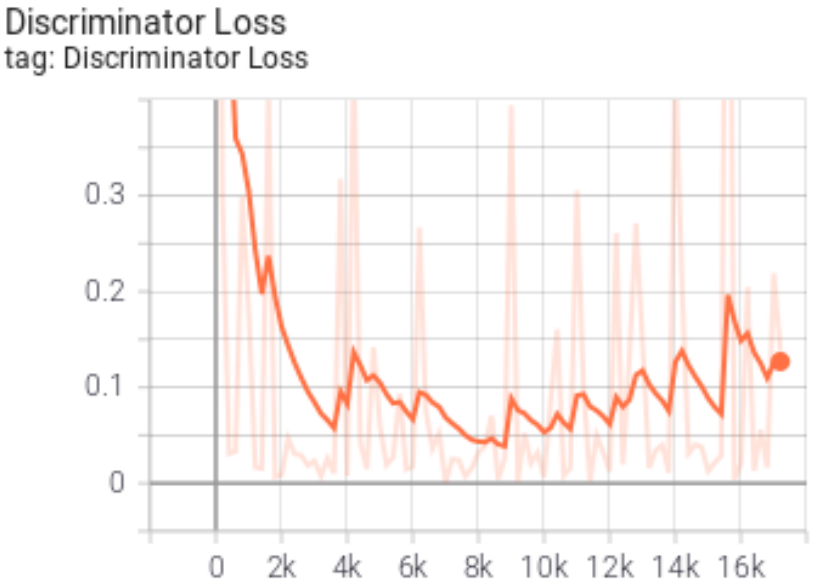
Graphs:



Graph 3. Adversarial Loss vs Number of Iterations

Graph 4. Content Loss vs Number of Iterations

Graph 5. Mean Square Error Loss vs Number of Iterations



Graph 6. Discriminator Loss vs Number of Iterations

Even though the network is small, we observed that even 100x100 images together with the generator does not fit into a 4 GB GPU (Batch size is 1, smallest value possible). Therefore, we decided to divide the image into 32x32 size parts, and upscale these smaller parts. After upscaling, we merged them, obtaining a high resolution image. Because of this, our network can upscale any image as long as their width and height are a multiple of 32.

Figure 8. Original low resolution image

Figure 9. Upscaled version of the original image. Note that the difference is quite noticeable around the buildings.

4.2. Application

4.2.1 Application Implementation Overview

The implementation of the desktop application for Upfix was divided into three parts: the graphic user interface, the back-end of the user interface and finally the integration of the desktop application with the model. There are four pages that the users can interact with: Upload Page, Loading Page, Display Page and Options Page.

The flow of the application is mainly as follows: the user is prompted to upload a video in the Upload Page. After fetching the path of the desired video, this path is forwarded to the next page, which is called the Loading Page.

Loading Page is responsible for using the Model Interface to make necessary calls for the Python scripts to be executed for the upscaling process. In the meantime, the progress is displayed to the user by a text that lets them know what part of the upscaling is being done, e.g. extracting frames, merging frames, concatenating audio etc. This page forwards the upscaled video to the next page, which is called the Display Page.

Display Page has the video thumbnail, extracted from the upscaled video itself, by choosing a random frame from it. The user can watch the video by clicking a button which starts up their default media player application. They can also choose to save the video to their computers. If they have the auto-save option off, they will choose the save location themselves. Otherwise, the video will be saved automatically to the location they chose in the Options Page. After this, the user may choose to upscale another video, or terminate the program.

4.2.2 Application UI

The user interface was prepared using Java, JavaFX and XML with the open source visual layout tool Scene Builder.

While the FXML files displayPage.fxml, loadingPage.fxml, optionsPage.fxml and uploadPage.fxml dictates what the pages in the user interface should look like, their respective Java controller classes DisplayPageController, LoadingPageController, OptionsPageController and UploadPageController determine how the objects on the page interact with each other and/or the user. A more detailed synopsis of the controller classes is given below.

UploadPageController

Details the functions: In the event that the upload button is clicked, it prompts to open the file explorer of the user's computer and requires that they choose a video to be upscaled. Once the user chooses their video, a new Video object is created with the chosen video's information. It loads the Loading Page if the next button is clicked, or loads the Options Page if the options button is clicked. Loading Page has the access to the Video object created in this class.

OptionsPageController

Details the functions: Upon the select button being clicked, it prompts the folder chooser and requires the user to choose a folder destination. If a directory is chosen, then it is set as the default save location for the upscaled video. If the auto-save checkbox is checked by the user, then the auto-save functionality is enabled, which means that the user will not see the file chooser window when they want to save the video. Instead, the video will be automatically saved to the desired save location, and the user will see a confirmation message.

The option preferences are saved in an XML file, which is updated when the user terminates the program and they are read from the XML file whenever the user starts the program. This is done in order to not lose the data when the program terminates.

The back button loads the previous page the user was on.

LoadingPageController

Details the functions: Communicating that the upscaling process of the model can start. This means that the Python scripts will start executing in this order: extracting the frames, upscaling the video, merging the frames, concatenating the sound, and lastly, extracting a thumbnail for the video.

This screen also updates the text and images on the screen accordingly as the upscaling process continues. If the back button is clicked, it deletes the video information and loads the Upload Page. If the next button is clicked, it creates and passes the upscaled video object and loads the Display Page.

DisplayControllerPage

Details the functions: Displaying the thumbnail extracted from the upscaled video. If the save button is clicked and auto save was not selected beforehand, it prompts open the file chooser of the computer and requires the user to choose a destination for the upscaled video. The initial location that shows up in the file chooser is the one that the user set on the Options Page.

If auto-save is enabled, the video is saved to the desired location automatically and the user is prompted with a confirmation message.

If the home button is clicked, temporary files made while upscaling the video are removed, and the Upload Page is displayed once again to the user so that they can upload a new video.

4.2.3 Integration with the Model

We wrote utility Python scripts almost like a service layer that does the actual frame extraction, upscaling etc. and a Java class to act as an interface to the Python scripts to integrate the model to our application. Here are the details:

ModelInterface

This Java class is our interface to run Python scripts from the Java application. It contains the information about the folder names used during the

upscaling process and public methods to initiate each of the scripts that do the following steps of the upscaling process:

extract_frames

This is the first script to be called, extracts the frames of the input video file at the specified rate, resizes them and saves them into the specified folder.

upscale

Improves the resolution of the extracted frames by running the model on each of them and saves them back into the folder.

merge_frames

Merges the upscaled frames into a video file and saves it into the specified folder.

concat_sound

Concatenates the sound of the source video onto newly created upscaled video.

create_thumbnail_image

From the upscaled video, extracts a random frame and saves it into the specified folder, to be used as the thumbnail in the later stages.

remove_folders

Clears the unnecessary folders that were produced during the process.

5. Testing Details

Since the testing strategy of the model and the application are slightly different, we are going to explain them separately.

5.1. Testing Details on Model Development and Integration

Docker

First of all, we should emphasize that testing in neural network development is not like testing any other software. We didn't write any test code such as unit test, module test etc. to test our model classes. However, this does not mean that the model code works everywhere and at all times. While the testing strategy is different in model development as we said, it has similar aspects to other software testing problems such as environment dependency. The model code requires some specific Python, OpenCV, TensorFlow versions. In order to get rid of the problems caused by version differences and OS differences, we containerized the model code by using Docker. In this way, we did no longer need to consider installations. Also, both two teammates in the model team could easily share the codes with each other and use them in both Windows and Ubuntu computers.

Metrics for Testing Model

As we said, testing in the neural network is not like testing any other software. The reason is that neural network development already consists of training and testing parts. However, the test in neural network development is to check how much the model's results have improved over time. In order to examine this progress, we used some metrics such as SSIM, PSNR values.

Integration Testing

In the beginning, we worked independently as the model team and the application team. However, we had to integrate the upscaling code into the application after the model training was completed. As the model team, we developed an interface so that the application team could use proper Python codes without knowing their inside. We tested this interface both by using unit testing and testing manually before integration. Also, it is important to say that we tried to minimize defects by applying integration testing.

5.2. Testing Details on Application Development

Before the integration of the model into the desktop application, there was only an interface that allowed switching between different pages in the application, and some other buttons that had the functionality of selecting a file from the local file system. These buttons (select file, go to the next page, go back to the previous page) were tested manually as there weren't many of them and it was faster to try them out than writing unit tests. A lot of the tests were related to the visuals of the application so we checked if the items were in the right place and they looked as intended.

After the integration of the model was made by using the interface provided by the machine learning team, several types of gameplay videos were tested in order to confirm the thumbnail extracting functionality, auto-save functionality. We also tested whether the Python scripts were being executed, and if they were being executed in the correct order. Further information about the integration tested was provided in section 5.1.

6. Maintenance Plan and Details

The main focus of our maintenance plan is the feedback from our users. Since we created an offline application, we can not monitor its health and performance directly so we will be depending on user feedback to continuously improve our application. The communication will be established by the contact section on our website. We will also be monitoring the updates for the tools and libraries that we used and create B plans for each of them in case they will not be supported in the future.

Improvements in both model and application context will be delivered to the user by version updates. The intermediate versions are planned to be released in case of bugfix requirements and smaller improvements in three months periods at the latest whereas the main versions are planned to be released for major updates like support for new games at least once every

year. We will also be working on the variants of Upfix which are described in section 8 of this document in more detail.

7. Other Project Elements

This section covers various project elements such as Consideration of Various Factors in Engineering Design, Ethics and Professional Responsibilities, Judgements and Impacts to Various Contexts, Teamwork Details and New Knowledge Acquired and Learning Strategies Used, which are written about in detail below.

7.1. Consideration of Various Factors in Engineering Design

Various factors affecting the design and implementation of the application have been considered and explained as follows:

7.1.1. Public Health

Upfix deep learning based upgrade models were trained on three games. We chose these games considering the age groups they appeal to. There are many games in the industry and most of them can negatively affect especially children negatively. Therefore, if we increase the variety of games in the future, we will consider public health and we will introduce some restrictions for some age groups to use.

7.1.2. Public Safety

Upfix does not hold any personal information about its users, nor does it sell to or share any information with third party applications. Since no information about users is kept or displayed, it is out of reach from malicious parties as well.

7.1.3. Public Welfare

Upfix does not affect public welfare, therefore it can not pose a threat to it.

7.1.4. Global, Cultural & Social Factors

Because Upfix's user experience consists only of interaction with the application itself rather than interaction with other users who use the application, it is not modified heavily by global, cultural, and social factors. The text that appears on the application's user interface does not contain any cultural insensitivity or hate-speech, and it is inclusive to all races and genders. The application has its user interface in English as it is the current lingua franca.

7.1.5. Environmental Factors

Upfix is a software application, so it is not affected by environmental factors.

7.1.6. Economic Factors

Upfix is completely free of charge to download and use. It also does not require a network connection.

7.2. Ethics and Professional Responsibilities

In order to not violate the privacy of our users, the application does not hold any personal information about its users nor does it sell such information to and/or share it with any third-party applications.

All data used by and in this project, including the intended neural network, complies with KVKK and GDPR regulations and does not violate any data privacy laws enforced by both Turkey and the European Union.

All sources, source libraries and third-party software used in this project are open-source and nonproprietary.

7.3. Judgements and Impacts to Various Contexts

	Impact Level (out of 10)	Judgement
Impact in Safety Context	6	Does not inquire any private user data to operate, does not share data to any third parties. However, requires access and/or ability to modify files as well as saving new ones to the system.
Impact in Global Context	4	While it is free, Upfix is yet an application that only exists in its own niche space. It could be made more accessible to the public via promotion/ ad campaigns or by adding to the system more video types other than just gaming video content.
Impact in Economic Context	0	Free to use and does not demand an internet connection to operate.
Impact in Environmental Context	0	None.

Table 2. Impact Level / Judgement Table

7.4. Teamwork Details

The members of the group worked on the following aspects of the project:

Rabia Nur Önal: She worked on the desktop application, designed and implemented mostly the backend structure of the application, tested it before the model integration and collaborated with other team members on verifying the system as a whole.

Hamza Pehlivan: He was the head of the model team since he has great interest and experience in deep learning. He took a role in preparing the dataset and creating the first model. He applied transfer learning on the ESPCN model. He also trained the GAN model and made it work in GPU. He found and read necessary research papers. He helped to integrate the neural network model into the app. Also, he tested the application after integration and helped to solve bugs.

Meryem Efe: She prepared the dataset and created the first CNN structure. She also helped Hamza in training the ESPCN model. She implemented Model Interface codes which includes all the video processing steps and took a role in integrating them into the application. Also, she tested the application after integration and helped to solve bugs.

Hazal Aksu: She worked in the desktop application part of the project. She helped in creating the graphical user interface and participated in the debugging and testing processes as well as contributing to the overall front-end design with her ideas.

Özge Yaşayan: She worked on the implementation of the desktop application. She constructed the general structure of the UI classes. She mainly designed and coded the layouts of the different pages and interactions among them. She contributed to coding the controllers and the FXML files for various pages. She also attended group debugging sessions and tested the application.

7.4.1. Contributing and Functioning Effectively on the Team

Our group consisted of different teams that had different responsibilities. Teams communicated among the members of the same team and we all tried to communicate with everyone in the group to summarize our team's progress regularly. Having subgroups, or teams, as opposed to one big group, helped us keep track of the tasks more effectively, and contributing was easier than having just one group because the tasks could be assigned more easily and we could spend more time focusing on our tasks rather than constantly reporting to a big group of people. It was faster to communicate with 2-3

members. We could also spend more time on defining the tasks more clearly and discussing the specifications among our teams, and since there were less members, it was easier to reach a consensus.

Moreover, in order to ensure proper teamwork, we divided the project into work packages and we chose each team member as a leader of one work package. The leaders determined the tasks in the work page which they are responsible for. Additionally, we divided the tasks into equal workload. In this way, we could make a proper and equal work-share. We met regularly in order to follow our progress and distribute new tasks.

Furthermore, even though every team member had a separate task, if one finished their individual part, they would help the other team members to do their parts in order to ensure the equality of teamwork and a collaborative work environment.

The project had a private Github repository to keep the progress of the project. Additionally, there were two separate folders in Google Drive for this purpose. One of them was used for the reports and tracking the weekly distribution of tasks. The other one included Google Colab files to train neural network models. Through these repositories and files, we could track our progress with ease and they could be shared with the supervisors and the jury members.

7.4.2. Helping Creating a Collaborative and Inclusive Environment

Before starting the project, every group member talked about their past experience with relevant technologies and mentioned what part of the project they would prefer to work on. Having this discussion allowed us to focus on our strengths as individuals and divide the work among ourselves in a way that would emphasize our skills. Some members took machine learning classes, and they had more experience with generating models, therefore they offered to work on the machine learning model. The rest of the members were more experienced in web and desktop application development, so they chose to focus on those aspects of the project.

Allowing members to choose what they wanted to work with, increased the collaboration and made sure that everyone was included in the project. Everyone had a chance to put their past experiences and existing skills into practice with the project and build upon those skills.

To encourage collaboration further, we had coding sessions as a group, especially at the times where we had issues with certain bugs. We would meet online as a group and one of us would share their screen, and we would debug the code together. This increased the collaborativeness in our group and allowed the two teams to interact more.

7.4.3. Taking Lead Role and Sharing Leadership on the Team

Our group did not have a pre-determined leader, however, some members were more involved in organizing meetings and arranging the report templates as well as reminding others about the deadlines. Overall, subgroups divided the work among themselves however they wanted, and then reported to each other. Everyone contributed to each of the reports as well. That being said, there were members who took more initiative in order to regulate teamwork among our group while the rest of the group members complied.

7.4.4. Meeting Objectives

Initially, we aimed to develop both a web application and a desktop application. We could not meet this objective completely, as we only developed a desktop application. This was because of the timing constraints and our lack of knowledge and experience about making use of web servers for upscaling purposes. We decided that it would make more sense to prioritize the desktop application since we were more familiar in that area and the desktop application utilized local resources of the users' computers. Developing the web application is mentioned as a part of the Future Work section as we believe that it would be very useful for users who do not own powerful computers.

As for the GUI of the project, our objectives were met except for a few things such as not being able to embed the video player into the application. We

faced some problems that we could not solve and instead, we chose to show the thumbnail of the upscaled video and redirect users to their default media players with the click of a button.

For the model part of the project, our objectives were largely met. Even though we didn't try to train the RNN model, we trained both CNN and GAN models. Also, we met our objectives for upscaling performance. In the beginning, we were aware that there is a trade-off between inference time and upscaled image quality. Therefore, we chose small networks in order to reduce inference time. However, in the end, we realized that even though we used a small network which works in real time, it still took a long time for video upscaling. The main reason for time inefficiency is relatively small GPU memory sizes. That is why it is not possible to copy the whole image or a batch of images into GPU. As a result, even though we got a good result, this super resolution technology is not ready to be used by end-users for real time video upscaling.

7.5. New Knowledge Acquired and Learning Strategies Used

The knowledge that was acquired by group members and the strategies used to learn can be mentioned separately for the application development team and the model team.

First and foremost, the application development team learned how to work with JavaFX and Scene Builder. This was done by following the official documentation of JavaFX and looking through online guides and examples. Since JavaFX is popular, there was an abundance of content on the internet. Scene Builder was integrated into the IDE that we used, which was IntelliJ Idea, and it was mainly figured out by the members with trial and error. Since it had a straightforward interface, we did not need external help except for a few instances, where we had to look up our problems on the internet, mainly on Stack Overflow [13]. Apart from that, we had to learn FXML which is the markup language that can be used with JavaFX. For this, we followed the official tutorial provided by Oracle [14]. Lastly, since the application was

written mainly in Java, we used online forums such as Stack Overflow [13] to find the answers to many questions we had about exceptions, bugs etc.

For the model development, even though the model team had little experience about machine learning and deep learning techniques before, they didn't know the neural networks used for super resolution. Therefore, first of all, the model team used online learning tools and followed the GAN course offered by deeplearning.ai in Coursera [15]. Since improving the results is based on doing more research, the model team have examined several articles about super resolution. Additionally, they learned OpenCV to perform image/video manipulation tasks, and Tensorflow to deploy CUDA-enabled processing. For this purpose, OpenCV and TensorFlow documentations are highly utilized [16], [17]. Another important new knowledge was Docker. The team used Docker to containerize and replicate the training and testing environments. DockerHub is used to find necessary Docker images [18]. Lastly, similar to the application team, the model team faced some errors and used online forums such as Stack Overflow [13] to solve them.

8. Conclusion and Future Work

In fact, the main aim of this project was to investigate upscaling techniques and to try to improve existing models. In order to improve results, we restricted the content to only three game types and we provided users with a desktop application that allows users to upscale these game videos without using internet connection.

We used super resolution in a specific content and with insufficient resources. Therefore, we had to limit our requirements as well. However, we are also aware that the super resolution techniques will be more improved, easy to use, and preferred in the future.

As a future work of Upfix, we have several aims as listed below:

Game Variety: For now, Upfix models were trained on only three game types which are Chess, Among Us, Age of Empires. In the future, we are planning to train our models on more kinds of games.

Upfix Web App: Even though we initially planned to develop a web app as well as a desktop app, our innovative expert strongly suggested that we give up a web app and focus on only model training. However, in the future, if Upfix would be developed as a web application, we could provide both B2C (Business to Client) and B2B (Business to Business) services. This means that we are planning to develop an upscaling service. We will both use this service for our web application (B2B) and allow the contracted businesses to use our services (B2C).

Browser Extension: There are some platforms leading the GVC industry such as YouTube, Twitch, Dailymotion. In the future work, we believe if users upscale the videos on these platforms by using just an Upfix browser extension, it would be more user friendly and preferable.

Upfix FotoScaler: Even though Upfix is a game video upscaler application, we actually upscale the frames of the video and then, we merge them again. So, the same models we used can also be trained for any other images.

The main difference between upscaling a whole video and a single photograph is the concern of inference time. Therefore, we could not use some models such as SRGAN for video upscaling because it was very slow even though it gave very impressive results. If we develop Upfix FotoScaler as well as the current desktop app, we could also use such models. Furthermore, we would give users the chance to see their old photos in high quality.

9. User Manual

9.1. Homepage, Uploading Video

On the homepage, users can click the *UPLOAD* button and choose the video by using the file chooser.

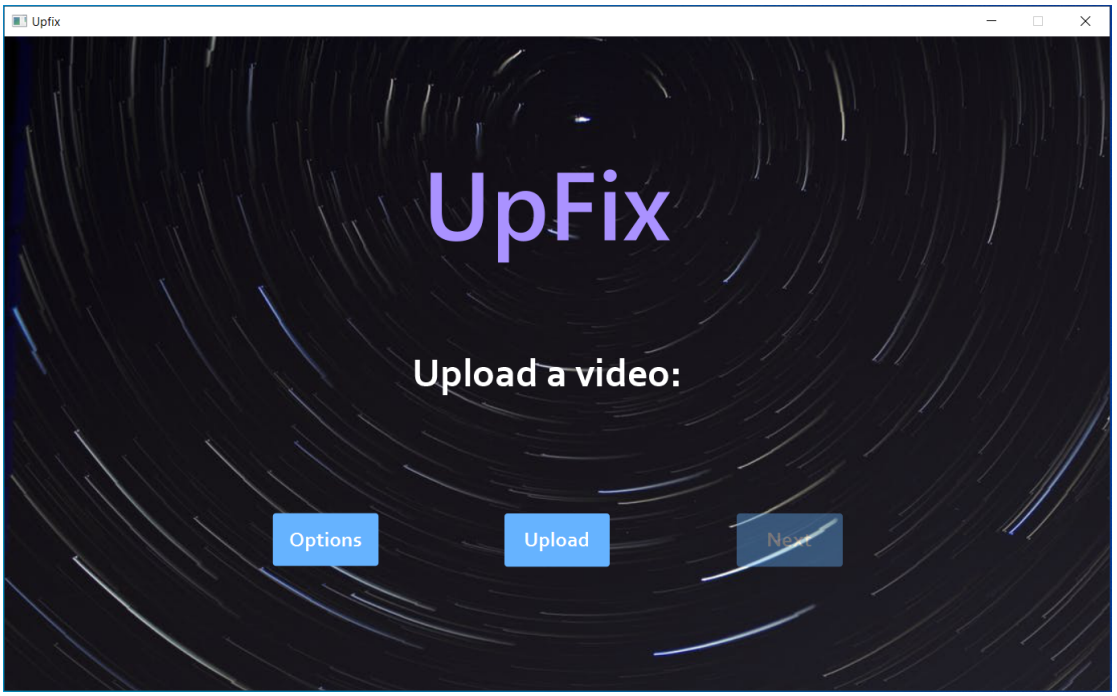


Figure 10. Initial Upload Page

When users upload their videos, they can start the upscaling process by clicking the *NEXT* button.

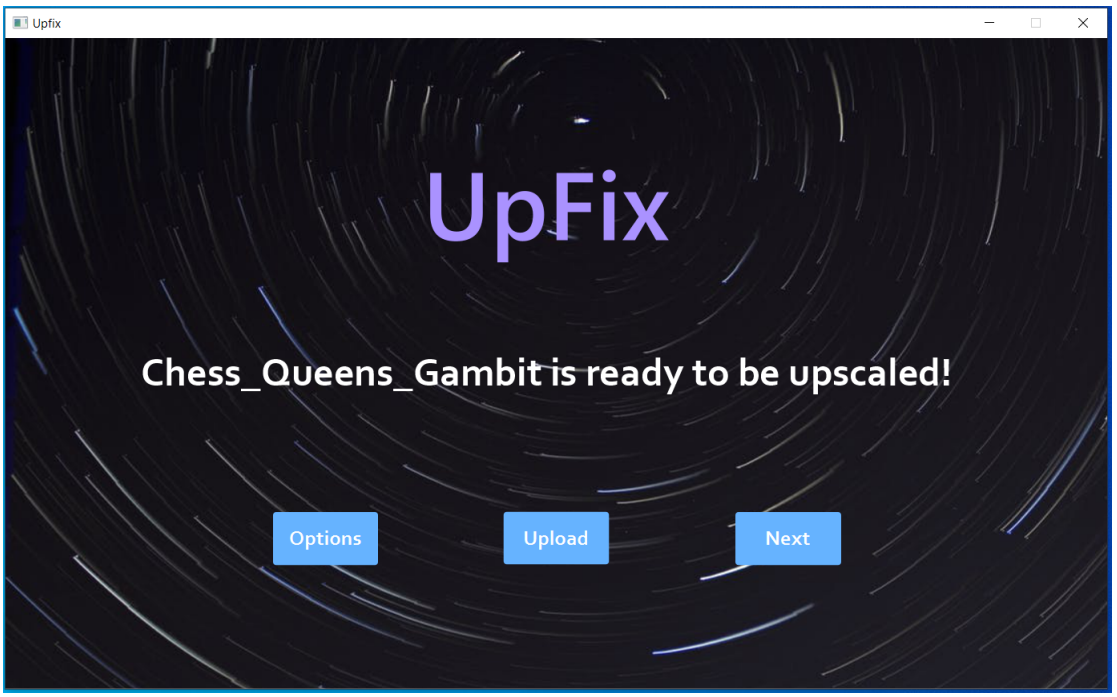


Figure 11. Upload Page after a video is selected

9.2. Video Processing

When the users start the upscaling process, Upfix firstly extracts the frames and upscales them. After, it merges all the upscaled frames and concatenates this upscaled video and the sound of the original video. Upfix shows the current status of the video processing so that the users can follow the process.

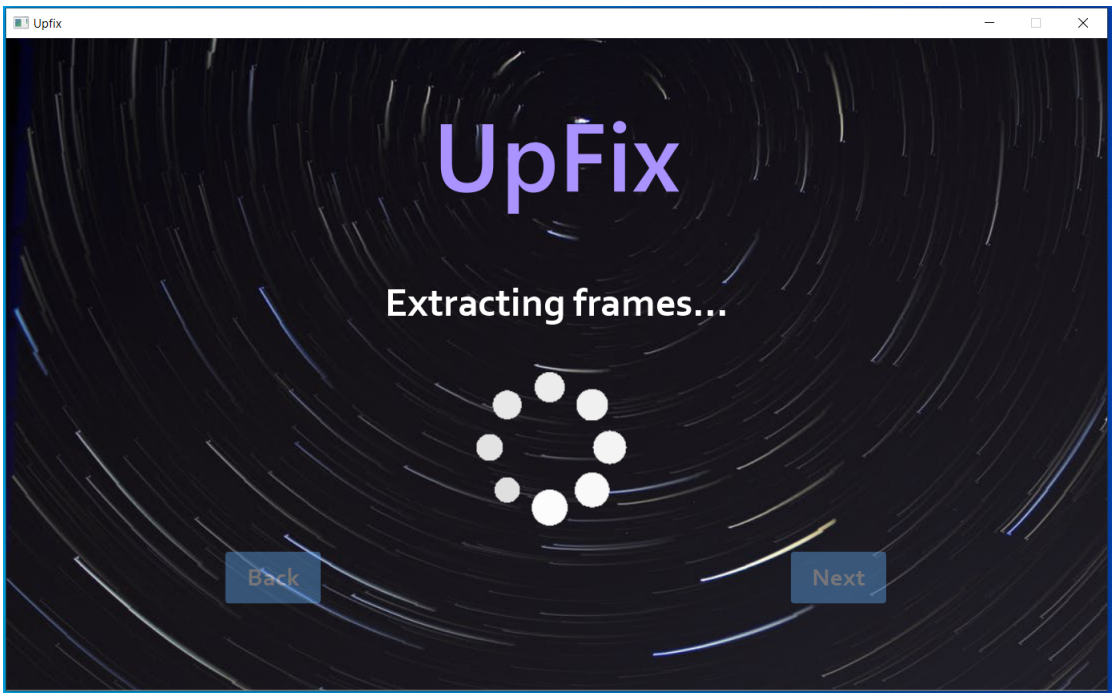


Figure 12. Loading Page 1

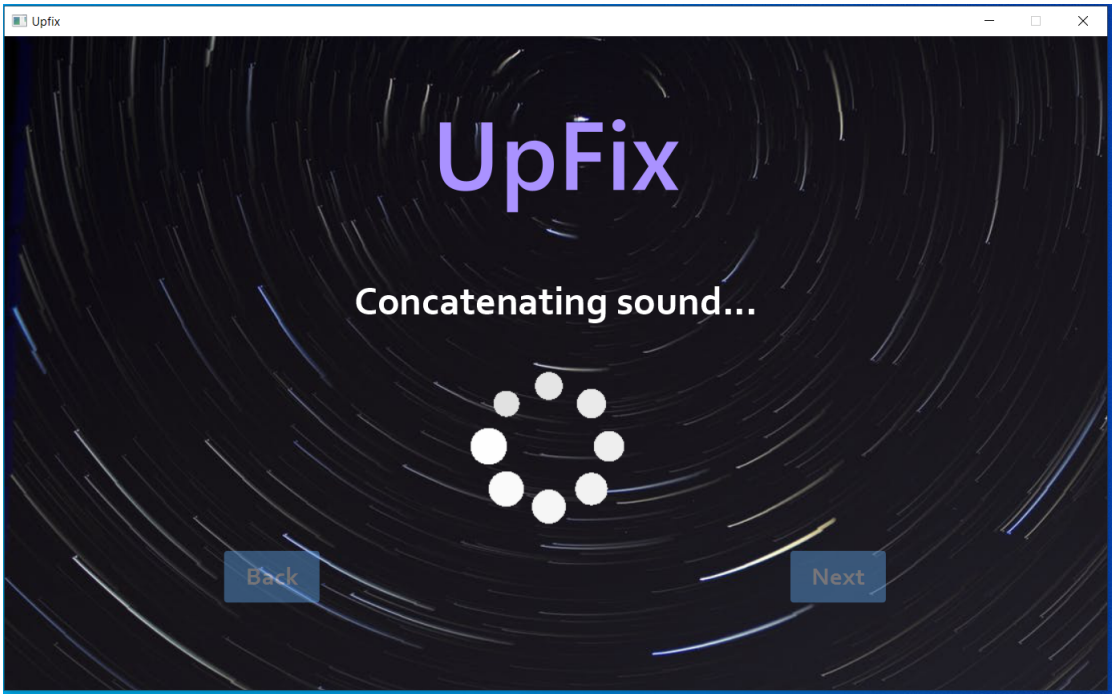


Figure 13. Loading Page 2

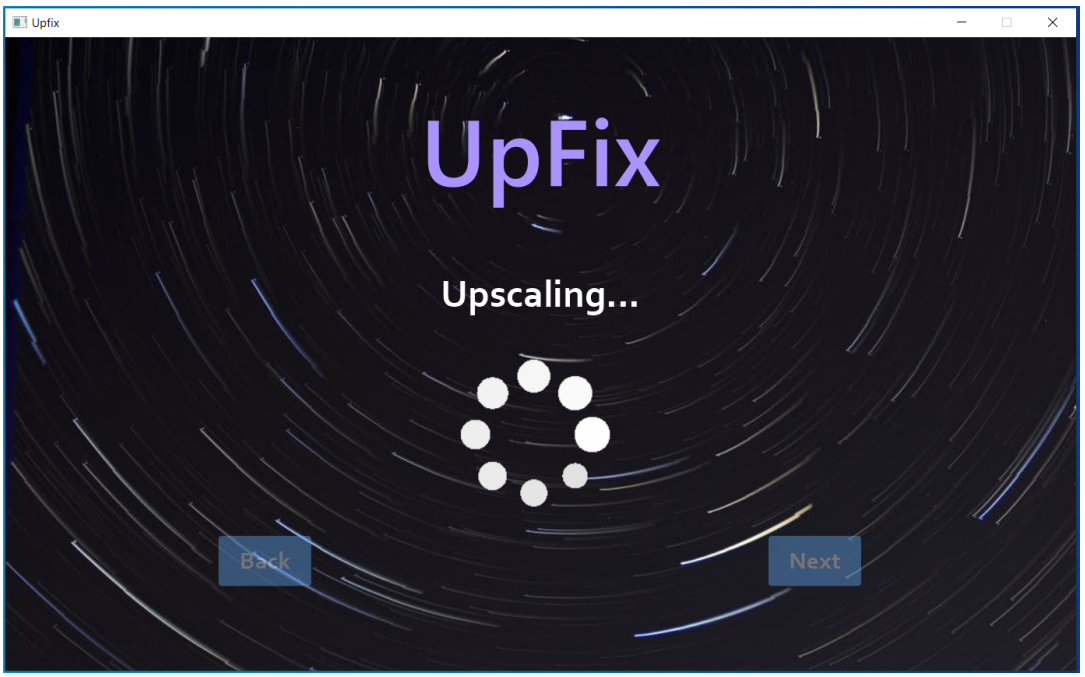


Figure 14. Loading Page 3

9.3. Watching, Saving Upscaled Video

When the video processing is completed, the users can go back and terminate this upscaling process or they can click the *NEXT* button to watch and save the video.

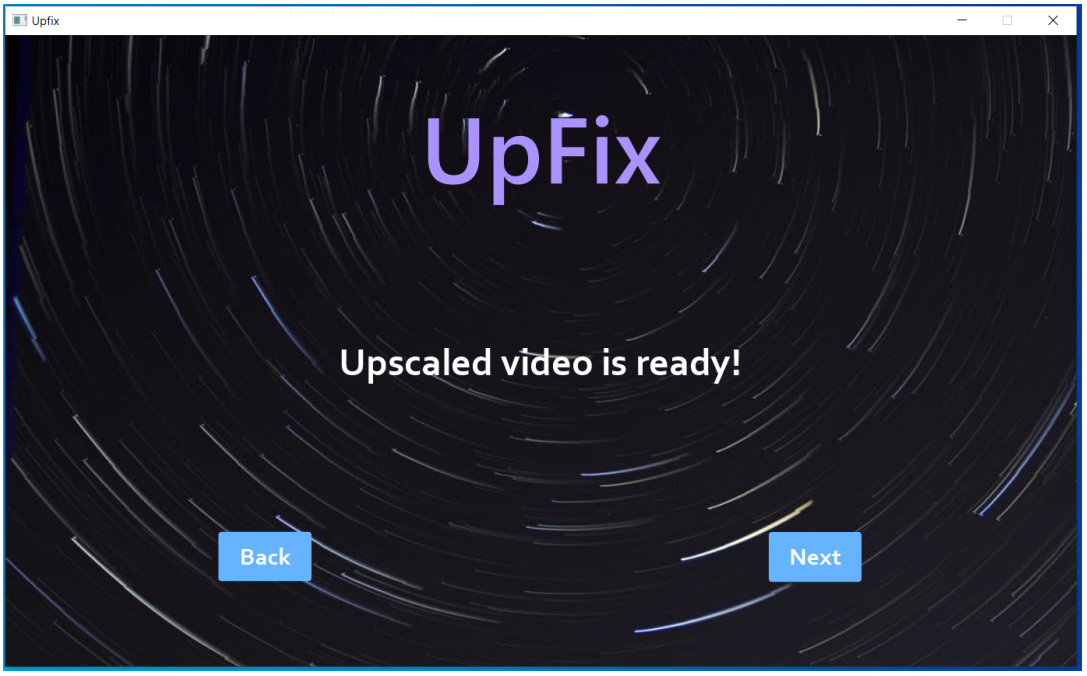


Figure 15. Loading Page 4

The users see the thumbnail of the upscaled video and they can watch the video on the default media player. If they don't click the *SAVE* button and they go back to the homepage, the upscaled video is deleted. If they click the *SAVE* button, they can save the video by using the file chooser.

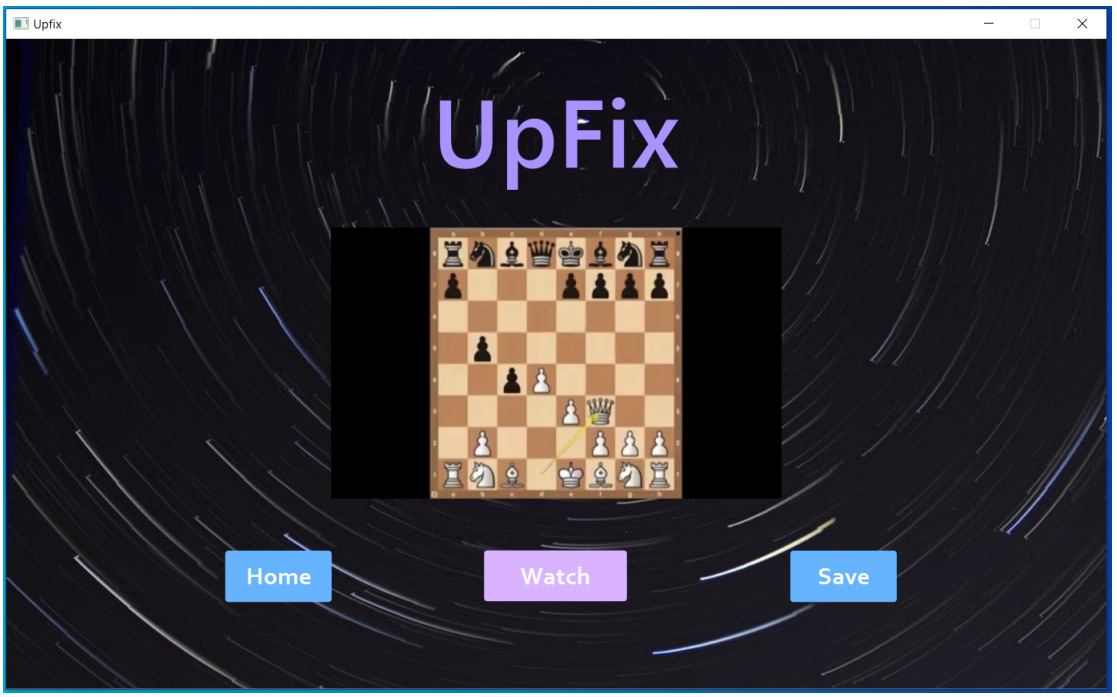


Figure 16. Display Page

9.4. Options

In the homepage, if the users can change the app settings by clicking the *OPTIONS* button. In the options, they can set the default save location. Also, they can activate the auto-save location if they click the *Use Default Location on Save* option. If this option is activated, Upfix saves the upscaled video to the default location automatically when the users click on the *SAVE* button.

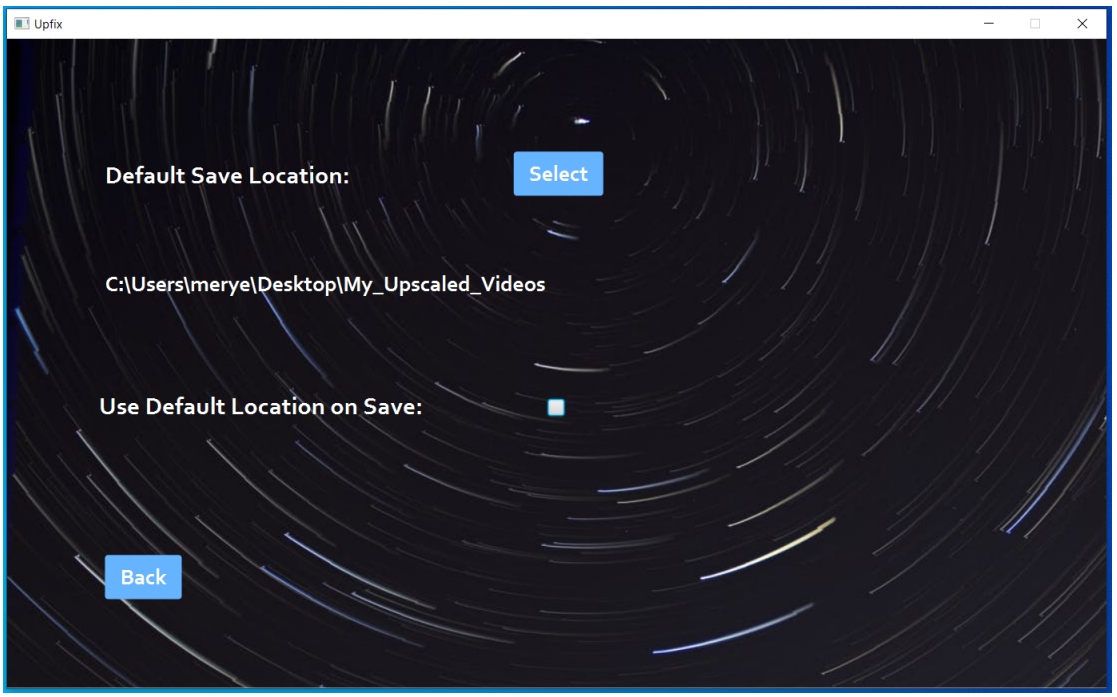


Figure 17. Options Page

References

[1] “Hours of Gaming Unboxing Videos on YouTube Watched on Mobile,” *Think with Google*. [Online]. Available: http://www.thinkwithgoogle.com/marketing-strategies/video/gaming-video-watch-time-statistics-on-youtube/. [Accessed: 27-Dec-2020].

[2] A. Guttmann, “Topic: Gaming video content market,” *Statista*. [Online]. Available: http://www.statista.com/topics/3147/gaming-video-content-market. [Accessed: 27-Dec-2020].

[3] “2020 Video Game Industry Statistics, Trends & Data,” *WePC*, 09-Nov-2020. [Online]. Available: http://www.wepc.com/news/video-game-statistics. [Accessed: 27-Dec-2020].

[4] “Counter-Strike: Global Offensive Broadcast,” *Valve Developer Community*. [Online]. Available: https://developer.valvesoftware.com/wiki/Counter-Strike:_Global_Offensive_Broadc_ast/. [Accessed: 27-Dec-2020].

[5] Dota Watch,” *Dota*. [Online]. Available: http://www.dota2.com/watch/. [Accessed: 27-Dec-2020].

[6] “Age of Empires Franchise - Official Website,” *Age of Empires*, 27-Aug-2020. [Online]. Available: https://www.ageofempires.com/. [Accessed: 27-Dec-2020].

[7] “Among Us on Steam,” *Steam*. [Online]. Available: https://store.steampowered.com/app/945360/Among_Us/. [Accessed: 27-Dec-2020].

[8] C. Dong, C. C. Loy, K. He, and X. Tang, “Image Super-Resolution Using Deep Convolutional Networks,” *arXiv.org*, 31-Jul-2015. [Online]. Available: https://arxiv.org/abs/1501.00092. [Accessed: 07-Jan-2021].

[9] X. Weber, “Deep Learning based Super Resolution with OpenCV,” *Medium*, 18-Jul-2020. [Online]. Available:

https://towardsdatascience.com/deep-learning-based-super-resolution-with-opencv-4fd736678066. [Accessed: 27-Dec-2020].

[10] Opencv, “opencv/opencv_contrib,” *GitHub*. [Online]. Available: https://github.com/opencv/opencv_contrib/tree/master/modules/dnn_superres. [Accessed: 27-Dec-2020].

[11] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] “Where Developers Learn, Share and; Build Careers,” Stack Overflow. [Online]. Available: https://stackoverflow.com/. [Accessed: 30-Apr-2021].

[14] “Release: JavaFX 2.2.40,” *Getting Started with JavaFX: Using FXML to Create a User Interface* | *JavaFX 2 Tutorials and Documentation*, 30-Aug-2013. [Online]. Available: https://docs.oracle.com/javafx/2/get_started/fxml_tutorial.htm. [Accessed: 30-Apr-2021].

[15] “Generative Adversarial Networks (GANs),” Coursera. [Online]. Available: https://www.coursera.org/specializations/generative-adversarial-networks-gans. [Accessed: 21-Nov-2020].

[16] “OpenCV modules,” *OpenCV*. [Online]. Available: https://docs.opencv.org/master/. [Accessed: 30-Apr-2021].

[17] “API Documentation: TensorFlow Core v2.4.1,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/api_docs. [Accessed: 30-Apr-2021].

[18] *Docker Hub*. [Online]. Available: https://hub.docker.com/. [Accessed: 30-Apr-2021].