Bilkent University

Department of Computer Engineering

Senior Design Project

Project name: Upfix

Low-Level Design Report

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

Supervisor: Uğur Güdükbay

Contents

Low-Level Design Report

Project Short-Name: Upfix

1. Introduction

Gaming is undoubtedly a significant source of entertainment for many people. The game 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 such 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.

Note 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 which will provide a platform that provides higher quality game videos. Initially, we are planning to provide 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.

In this report, we are going to narrate the low-level design of the project.

1.1. Object Design Trade-off

1.1.1. Inference Time vs Image Quality

In the neural network side of the project, we are trying to train a model which can give better results in super resolution. There are 4 OpenCV models trained for super resolution. One of them, EDSR, gives impressive results. However, there is a trade-off between time and quality. Even though EDSR has amazing results, it is a huge model and it takes a long time. Therefore, we chose a smaller model which is ESPCN.

1.1.2. Development Time vs Functionality

Initially, we decided to have a dedicated web application and a desktop application that provided similar functionality, even though there were minor differences, such as the web application being able to fetch videos from various websites like YouTube and Twitch. However, because of the development time constraint, we decided to prioritize the desktop application and only build the web application if the time allowed us to do so. This means that we will be losing some of the functionality provided by the web application that does not exist in the desktop application, however, we will at least complete the desktop application in the given time frame. We decided that this would be much more desirable than ending up with two unfinished applications.

1.1.3. Cost vs Performance

Even though we can use a neural network with much higher performance on super-resolution, we decided not to use it, because its cost can be too much for the end user. That is why the neural network's huge architecture requires using powerful GPUs. Therefore, we will use a different neural network which has smaller architecture in order to reduce cost, even though its performance is not as good as the previous one.

1.2. Interface Documentation Guidelines

In this documentation, all class names are named with standard class names in upper camel case. In the hierarchy, after class name, class description, its properties, its methods and explanations are listed.

Class: ClassName
Class description
Properties
property: PropertyType → Property description
Methods
method(param1: ParamType, param2: ParamType): ReturnType → Method description

1.3. Engineering Standards (e.g., UML and IEEE)

In all the reports, we followed UML design principles for all diagrams and the IEEE citation format for referencing the resources [8, 9].

1.4. Definitions, Acronyms, and Abbreviations

Term	Definitions, acronyms, and abbreviations
GVC	Gaming Video Content
SR	Super Resolution
Transfer Learning	Transfer learning is a machine learning technique where a model developed for a task is reused as the starting point for a model on another related task [10, 11].
OpenCV	“OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms [12].”
CNN	Convolution Neural Network [13]
ESPCN	ESPCN (Efficient Sub-pixel Convolutional Neural Network) is a neural network architecture provided by OpenCV. It can do real time video upscaling [14].
EDSR	EDSR (Enhanced Deep Residual Networks) is a neural network architecture provided by OpenCV. It produces impressive results on super resolution [15].
JavaFX	“JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms [16].”

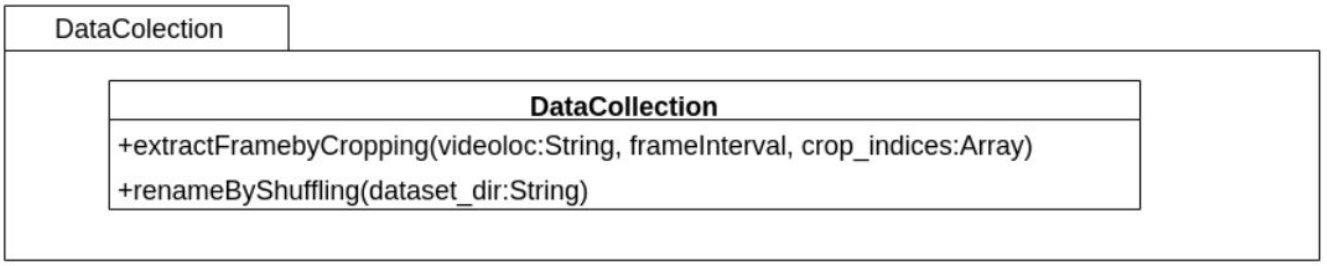
2. Packages

Our development team works on two parts which are training the model and developing the application. Since these two processes are carried out simultaneously, we will explain them separately in the following section.

2.1. Training Model Packages

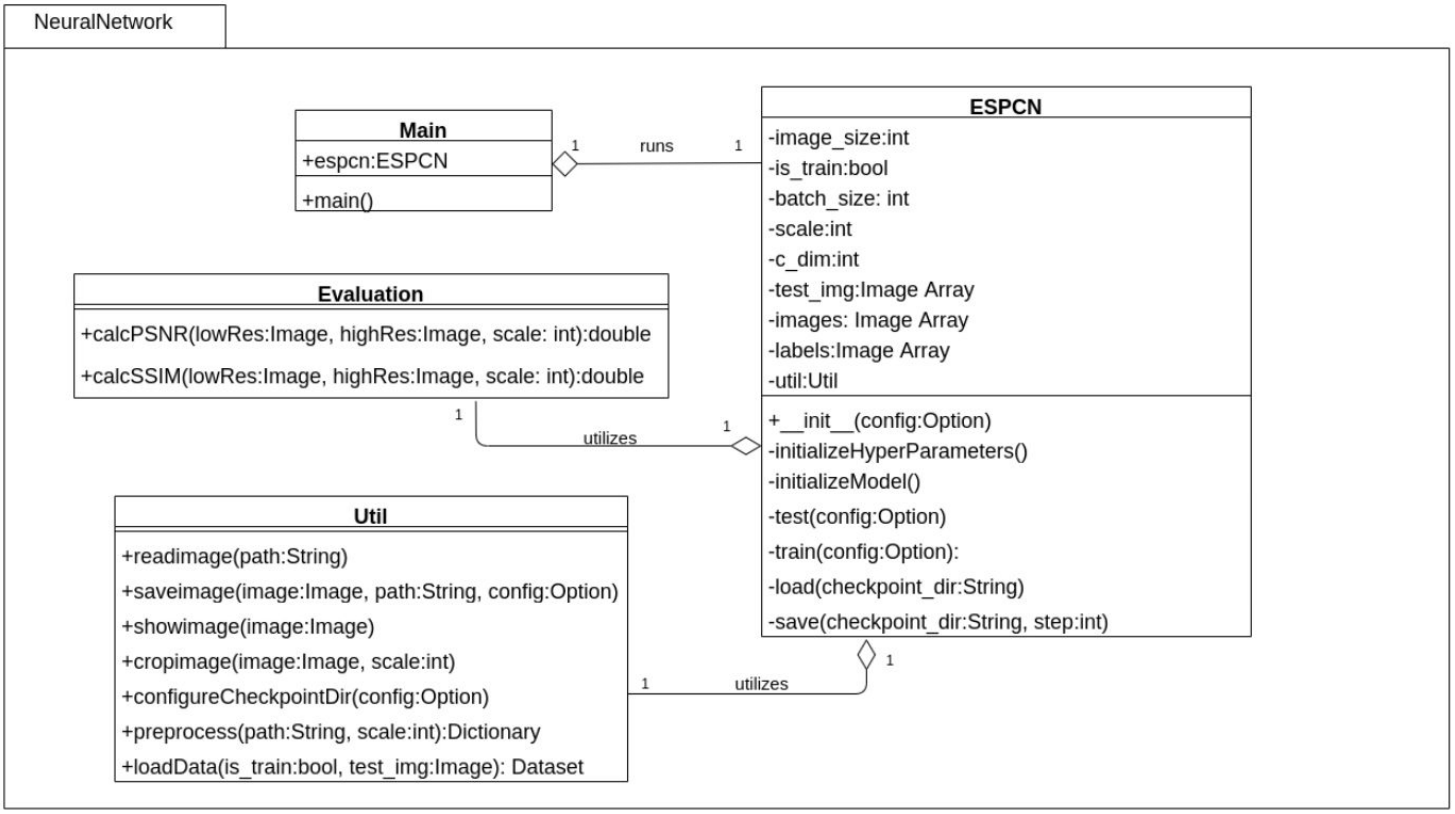
The system used to train models are divided into 3 packages by considering their purposes.

2.1.1. Data Collection Package



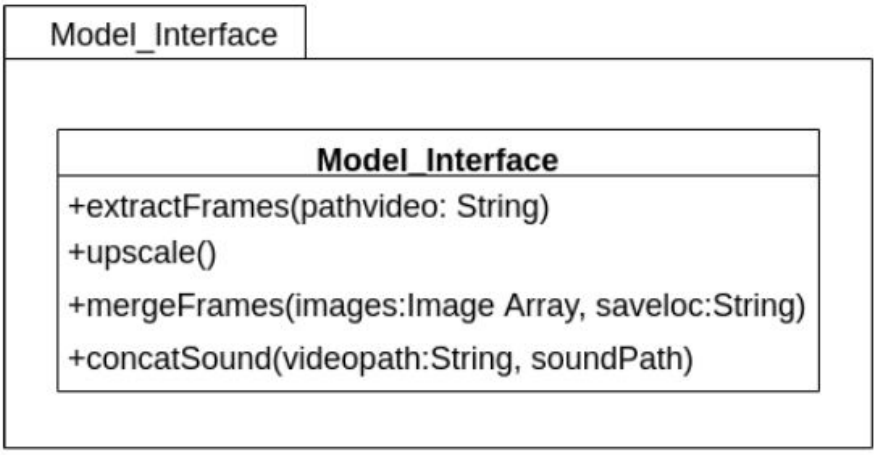
To train models, we did not use any prepared dataset, but we collected data by ourselves. Therefore, we had to preprocess the videos we collected in order to generate a dataset. This package is used for this purpose. We manually downloaded proper game video contents. After, through this package, we created a dataset by extracting and preprocessing the frames.

2.1.2. Neural Network Package



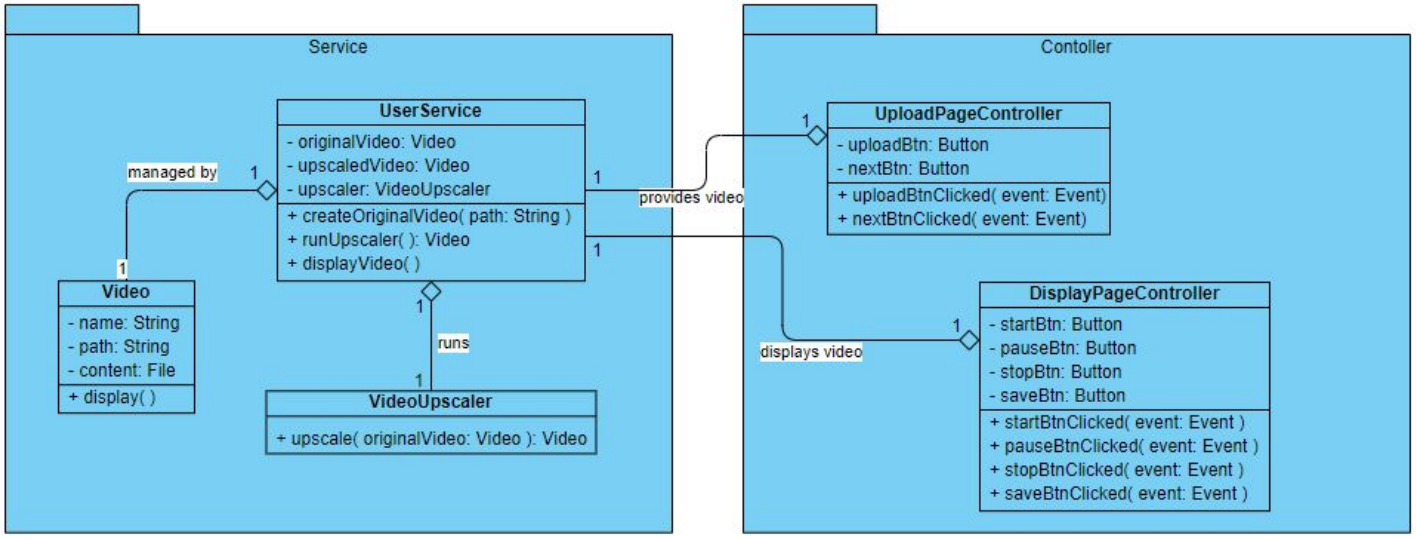
To train models, we planned to apply transfer learning on ESPCN, an OpenCV model that can do real time video upscaling. This package includes the classes which are used for training, evaluating, and testing the model.

2.1.3. Model Interface Package



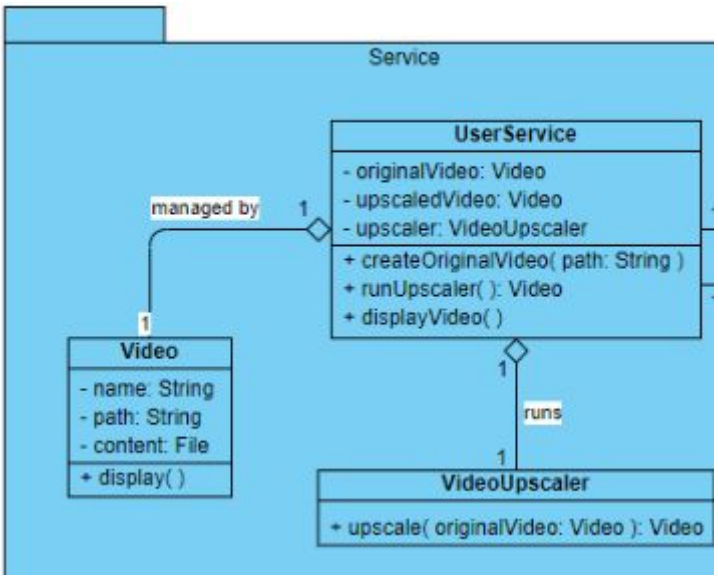
The desktop application is implemented in Java; however, video processing methods are done by using python libraries. Therefore, as a model developer team, we created the package which can have a role as an interface between model and application. In this way, after testing this package is completed by the model developer team, the application will be able to use the model through this interface.

2.2. Application Packages



The classes regarding the desktop application are contained in two packages: service and controller.

2.2.1. Service Package

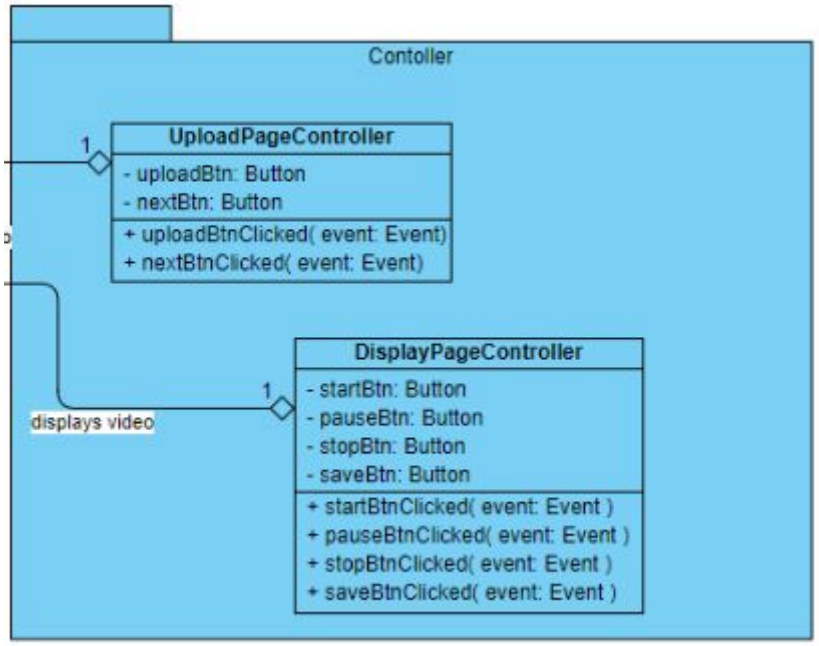


Service package includes a Video class that is necessary in order to store the original and upscaled video objects.

It also has a VideoUpscaler class that is responsible for upscaling the videos with the help of a Python script.

Lastly, the UserService class in this package has the tasks of registering the user's video selection to be upscaled, running the upscaler and displaying the upscaled video.

2.2.2. Controller Package



This package is responsible for providing the user with a way to interact with the application. There are various button objects that provide different functionality for the UI.

UploadPageController controls the page where the user selects a video file via a file selector.

DisplayPageController controls the page where the user can display the upscaled video and save it to a location on their computer that they desire.

3. Class Interfaces

3.1. Training Model Classes

3.1.1. Data Collection Package

Class: DataCollection
This class is used to create and preprocess the dataset.
Methods
extractFrameByCropping(video_loc: String, frame_interval: int, crop_indices: Array) → This method extracts frames from the video in the given path, crops the frames, and saves them to the dataset directory.
renameByShuffling(dataset_dir: String) → This method renames all the images in the given directory by shuffling their orders.

3.1.2. Neural Network Package

Class: ESPCN
This class represents the neural network model, performs training and testing operations.
Properties
image_size: int → Input image size.
is_train: boolean → Indicates whether train or test is run.
batch_size: int → Dataset batch size.
scale: int → Scale factor.
c_dim: int → channel dimensions
test_img: Image Array → Dataset for testing
images: Image Array → Dataset for training
labels: Image Array → Labels (high-resolution images)
util: Util → Util class object.
Methods
__init__(config: Option) → It is a Python constructor for the ESPCN object.
initializeHyperParameters() → It initializes hyperparameters like batch size, epoch and some others like the loss function and optimizer.

initializeModel() → It initializes layers in the model like the convolutional layers and activation functions.
train(config: Option) → This method starts training on the train dataset.
test(config: Option) → This method starts testing on the test dataset.
load(checkpoint_dir: String) → This method loads the model from the given directory.
save(checkpoint_dir: String, step: int) → This method saves the model to the given directory.

Class: Evaluation
To improve the model, it should be evaluated. This class is created for this purpose.
Methods
calcPSNR(lowRes: Image, highRes: Image, scale: int): double → This method calculates the PSNR score of the model.
calcSSIM(lowRes: Image, highRes: Image, scale: int): double → This method calculates the SSIM score of the model.

Class: Util
This class contains utility functions such as preprocessing and saving images.
Methods
readImage(path: String) → This is a utility function that can read images from a given directory.
saveImage(image: Image, path: String, config: Option) → This is a utility function that can save the upscaled image to a specific location.
showImage(image: Image) → This method displays the given image for debugging purposes.
cropImage(image: Image, scale: int) → This method crops the image so that it can be divisible by the scale factor.
configureCheckpointDir(config: Option) → This method configures the checkpoint directory in which models are saved.
preprocess(path: String, scale: int) : Dictionary → This method converts RGB color space into YCrCb color space. Because YCrCb color space requires less information (data), this process will allow us to infer the images faster.
loadData(is_train: bool, test_img: Image) : Dataset → This method loads the train data if is_train parameter is true, otherwise it loads the test data.

Class: Main
This class is the main class which calls ESPCN and starts training / testing.
Properties
espcn: ESPCN → ESPCN object
Methods
main() → This method initializes the ESPCN object and starts the training by calling proper methods.

3.1.3. Model Interface Package

Class: ModelInterface
This class is created to make the connection between the model and application. It includes the necessary methods to perform video processing and upscaling operations.
Methods
extractFrames(video_path: String) → This method extracts frames of the given video in order to upscale these frames.
upscale() → This method upscales the extracted frames by using the ESPCN model whose parameters were fine tuned by transfer learning.
mergeFrames(images: Image Array, save_loc: String) → This method merges the extracted frames and creates a new video.
concat_sound(video_path: String, sound_path: String) → This method concatenates the sound and the created video.

3.2. Application Classes

3.2.1. Service Package

Class: UserService
This class contains the original video that is going to be upscaled and the methods to upscale the video and display it.
Properties
originalVideo: Video → The original video that is going to be upscaled. upscaledVideo: Video → Upscaled video object. upscaler: VideoUpscaler → Instance of the VideoUpscaler class.
Methods
createOriginalVideo(path: String) → Creates the video instance according to the input from the controller package. *runUpscaler( ): Video* → Method that runs the upscaler to upscale the video. *displayVideo( )* → Displays the upscaled video.

Class: VideoUpscaler
This class is where the upscaling process takes place.
Methods
upscale(originalVideo: Video): Video → This method upscales the image quality of the given video and returns the resultant upscaled video.

Class: Video
This entity class contains information about the video objects.
name: String → Name of the video. path: String → Path of the video. content: File → Actual content of the video.
Methods
display( ) → This method displays the video.

3.2.2. Controller Package

Class: UploadPageController
This class controls the page where the user uploads the video to be upscaled.
Properties
uploadBtn: Button → Button that allows the user to upload a video to be upscaled. nextBtn: Button → Button that switches to the next page.
Methods
uploadBtnClicked(event: Event) → Handles the clicking on the uploadBtn event by displaying the file selector for the user to select the video file. *nextBtnClicked(event: Event)* → Handles the clicking on the nextBtn event by moving onto the next page.

Class: DisplayPageController
This class controls the page where the user displays and saves the upscaled video.
Properties
startBtn: Button → Starts the video. pauseBtn: Button → Pauses the video. stopBtn: Button → Stops the video. saveBtn: Button → Saves the upscaled video to the desired location.
Methods
startBtnClicked(event: Event) → Handles the clicking on the startBtn event by starting the video. *pauseBtnClicked(event: Event)* → Handles the clicking on the pauseBtn event by pausing the video. *stopBtnClicked(event: Event)* → Handles the clicking on the stopBtn event by stopping the video. *saveBtnClicked(event: Event)* → Handles the clicking on the saveBtn event by saving the video to the location picked by the user.

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] “Unified Modeling Language”. [Online]. Available: https://www.uml.org/. [Accessed: 08-Feb-2021].

[9] “IEEE Reference Guide.” [Online]. Available: https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf. [Accessed: 08-Feb-2021].

[10] J. Brownlee, “A Gentle Introduction to Transfer Learning for Deep Learning,” *Machine Learning Mastery*, 16-Sep-2019. [Online]. Available: https://machinelearningmastery.com/transfer-learning-for-deep-learning/. [Accessed: 27-Dec-2020].

[11] F. 7 and S. Martin, “What Is Transfer Learning?: NVIDIA Blog,” *The Official NVIDIA Blog*, 14-Feb-2019. [Online]. Available: https://blogs.nvidia.com/blog/2019/02/07/what-is-transfer-learning/. [Accessed: 27-Dec-2020].

[12] *OpenCV*, 13-Oct-2020. [Online]. Available: https://opencv.org/. [Accessed: 27-Dec-2020].

[13] “Convolutional Neural Networks,” *Coursera*. [Online]. Available: https://www.coursera.org/learn/convolutional-neural-networks. [Accessed: 27-Dec-2020].

[14] W. Shi, J. Caballero, F. Huszár, 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,” *arXiv.org*, 23-Sep-2016. [Online]. Available: https://arxiv.org/abs/1609.05158. [Accessed: 08-Feb-2021].

[15] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced Deep Residual Networks for Single Image Super-Resolution,” *arXiv.org*, 10-Jul-2017. [Online]. Available: https://arxiv.org/abs/1707.02921. [Accessed: 08-Feb-2021].

[16] *JavaFX*. [Online]. Available: https://openjfx.io/. [Accessed: 27-Dec-2020].