Vision``

Mohannad Atmeh
Abdel-aziz Hamdan
Mohammad Ibrahim Abu-Amara

Functional Requirements

Denoising Image

Function ID: V-00

Denoising images, especially when these images are taken at night, is crucial to our system, we can enhance the image as far as we can, but without the denoising, it'll still be not so clear. There are various sources of noise in an image, it can be through compression, network transmissions, and the way the image is saved from hardware components to zeros and ones can cause noise as well. So how are we going to represent noise in an image, the answer is, drum rolls please….., MATH, as scary as the word MATH sounds, it's actually provide a simple and neat definition of how noise can be represented in an image,

$$m(x) = o(x) + u(x), x \in \Omega$$

where $o(x)$ is the original image without noise, $u(x)$ is the noise that was applied through some source, or various sources, to the image, and finally, $m(x)$ is the input image with noise. x is a set of pixels, and $\Omega$is a collection of pixels, which is the entire image basically.

With basic algebra math we can see that we can obtain the original image accurately by subtracting the noise $u(x)$ from the input image $m(x)$. But as the saying goes "Names are not always what they seem." ~Mark Twain, unless there is some magic way to figure out where the noise(s) came from, we can't just invert them, not to mention the fact that seeing if an image has noise or not is hard, we as humans think it's intuitive, but computational wise, it's really not.

How are we going to solve it ? How are we going to implement it ? This is such a complex problem oh my god!!!!!!!. Convolutional Neural Networks, CNNs, to the rescue. It just makes sense to use Artificial Intelligence in general to solve these kinds of problems, and it even makes more sense to use CNNs as a method to implement this problem, we are mapping a known input to a known output. How do we know the output?, one might wonder, easy, and simple answer: we know how to add noise to an image, so we get a bunch of datasets that have high quality images in it, add some noise to it, and *Voila*! We have a bunch of noisy images, plus, we can also use this to evaluate our model output, because as I said we know the output.

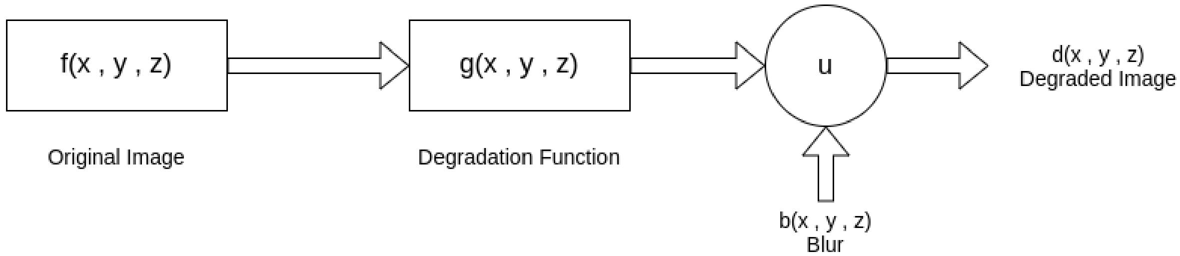
To train this network (sometimes called model) we need a dataset, fortunately, there are many open source datasets that we can use, we used many of them from different categories, faces, normal natural images, and many more, like for example we used the SIDD dataset which has 30,000 pairs of noisy images that have different lighting conditions using different cameras, and the original image. As well as the FFHQ dataset, which has more than 70,000 high quality human faces images with different variations in terms of age, ethnicity, background, glasses, hats, and more.

Enhance Image

Function ID: V-01

This is quite literally half of the core to our system, we heavily rely on image enhancing to get the job done for our users. For that reason, we need to be careful when we solve the super resolution problem, sometimes called the Single Image Super Resolution Problem or SISR for short, because there are high standards that are set for this kind of feature. Some of the standards that are set for this feature is, producing a high quality image with the same details in the original image, delivering single-digit millisecond performance at any scale, having low memory footprint in both the user side and the server side, and many more.

At the SISR problem it is also a mathematical model, just like any other computer science problem, to understand how to derive a mathematical model we first have to understand some concepts that relate to it. First is, Degradation Modeling, degradation modeling is an effective reliability analysis tool for things with failures caused by degradation; images degradation can come in different forms such as noise, camera misfocus, motion blur, …, etc. To give an example, a simple image degradation model can be like this one,



In our problem, the Single Image Super Resolution, the linear degradation model is formulated by,

$$i = A_s Bx$$

Where $i \in R^{M \times N}$ is the input image, $x \in R^{M_s \times N_s}$ is the high resolution image that we want. $B \in R^{MNs^2 \times MNs^2}$ is the linear operator that blurs the high resolution image, x, with the decimation in a factor of s in each axis, which is the outcome of the multiplication by $A_s \in R^{MN \times MNs^s}$.

We are solving for the unknown x, which is the high resolution image, and recover it from the known measurement i, which is the input image. We also have to pay attention that the model doesn't always have to be linear, for example: due to compression, noise, or even some other unknown(s).

We are going to use an Enhanced Super-Resolution Generative Adversarial Networks, ESRGANs for short, we talked about SRGANs in Phase one, it was from the Twitter Cortex research group, and it showed some really interesting results. Yet a group of chinese computer scientists won first place in the PIRM2018-SR Challenge and showed even better results, and guess what they called their model ? that's right, ESRGAN, they found some flaws in the original model, fixed them, enhanced the model, and gave better results, simple right ?, well yes but actually no, there are more behind how they actually enhanced the model and kept consistency with the old model.

The normal SRGANs was very influential in the Super Resolution problem; which is an important part of the computer vision industry, and it showed very promising capabilities of creating and generating realistic image textures.
However, there were some details in the image that didn't improve, so the group of chinese computer scientists found some areas to improve and enhance, which is the network architecture, the adversarial loss, and the perceptual loos, so they added the Residual-in-Residual Dense Block (RRDB) without batch normalization as the building unit of the neural network, which lower the computational complexity of the network. As well as got some influence from other types of GANs, and from there they were able to enhance the SRGANs.

For the training data we used a high quality dataset, the DIV2K dataset that contains 800 images. As well as some other datasets to make the model diverse, such as the Flickr2K dataset, which has more than 2000 2K images.

Image to Data

Function ID: V-02

This is an extra feature to make use of images, and to provide our users with some insights from the image. Some of these insights can be licence plates recognizing, faces detections, object detections, and more to come. It can also assist users to see details that they can't recognize, or some object they didn't pay attention to.

We are going to use Amazon Rekognition, for its simplicity and ease of use, in addition to having a well documented APIs. Nevertheless, if we need to, we are going to implement some features that the API doesn't have.

Denoising Video

Function ID: V-03

Just like in images, denoising in video has the same degree of importance. In addition to that, most of our system data will be videos, that's what the project description said. Nonetheless, we showed that it's not efficient if we process videos frame by frame, or a batch of frames, to be treated like images. So we simply treat videos as they are and process them as a whole.

We also showed that Conventional Neural Networks, CNNs, can solve complex problems where we map from a known input to a known output, so we are going to use it. Why not use GANs? GANs can also be used to solve this problem, but it would be tedious to use it, as it is too complex for this kind of problem, along with, the state-of-the-art models right now also use CNNs to solve this problem.

Training was done using the DAVIS dataset, in addition to other datasets that help the model with diversity.

Enhance Video

Function ID: V-04

This is the other half of our core, which puts the cherry on top of the cake for our system, and it's also the most complex problem in our system, yet. The name of the function itself is scary, imagine what the model will look like.

There was an interesting paper that we reviewed, iSeeBetter, this paper was a grad student project in fall 2019, in the CS230 Deep Learning course in Stanford University led by Andrew Ng to be exact, and it showed some really amazing results. It makes practical and effective use of GANs, along with the concept of Recurrent Back-Projection Networks, RBPN for short, as its generator, and its known for the RBPN to solve the video super resolution problem. It also uses a four-fold loss function (Adversarial, Perceptual, Mean Squared Error, and Total-Variation) to avoid the problem of misrepresentation of perceptual quality. With this solution it easily set a new bar for the other state-of-the-art models.

Training was done using the Vimeo90K dataset, and many other datasets to make sure we have a general model, so that we can enhance all kinds of videos, no matter the background, the object, and all the different details in the video.

Non-Functional Properties

Usability

Ease of use is an essential part of every system, you can't afford bad user experience because the system isn't intuitive for some users. Ease of use does not relate to only user experience but also to developers and software engineers who are going to make the system better, it takes less time when the system is easy to understand and use.

When we talk about user experience in our system, then our system is simple, meaning that anyone can use it intuitively, and get their desired results in no time. Why is that?, you might ask, because of the fact that uploading an image or a video to an application is easy, plus our user interface will make sure that everything is straightforward.

When we talk about the developer or the software engineer side of view, our system has simple problems to understand, but hard to solve, like for example, most of the ways we use to solve problems is AI, Machine Learning, and Deep learning, as well as some data engineering concepts, cloud computing, and more computer science topics. So it might not be very straightforward to understand how the system was built. However, the layers of abstractions that we provide will make sure that we make it easier to understand, like for example some data visualization so it's easier to look at, without knowing much of how things happen behind the scenes, plus some APIs to handle communicating with the models and other components for the system, and more.

Performance

Performance is key to all systems nowadays, users want their response as soon as possible, and software engineers want their system to have the capability to handle more load as the system grows. It's a very important quality attribute that can tell alot about any system. Furthermore, it affects many other non functional properties, we can't have an ease of use with the system taking too long to respond, we can't have reliability of a system talking too much memory from every single request, and many more properties that won't hold under poor performance.

Our system will hold performance easily, we use state-of-the-art Deep Learning models that provide single-digit milliseconds responses, along with relying on cloud computing to not worry about extra overhead from poor server configurations, as well as to distribute our system to also distribute the load into many machines. Not to mention that we will have a small memory footprint which helps with many operations and reduce the size of our running processes.

Security

Let’s assume that security wasn’t important, whether it’s user data, or the system as a whole. Users images and videos would be just visible to everyone who can look at the network, moreover, we, the software engineers would also have access to user’s data, some engineers say that “it’s okay” so we can make the product/system better for the user. Now throw that assumission out of the window, burn it, and walk away like you did not see anything.

User’s data MUST be visible to the user only, we have to make sure that no one is allowed to view that data, unless it’s theirs of course. So, our plan is to have our deep learning model run on the client side; to prevent network calls and data transmissions. However, for now we will deploy our models on the cloud and secure our network, until we find an efficient way of running deep learning models on the client side; because they might be heavy to load in every client device (web, mobile, desktop).

Overall, we are going to make sure that we secure our system, and that our cloud provider has a secure network, as well as not having a registration system for our users, How are we going to know our users?, you might ask, the answer is, that is the whole point, we should not know which image belongs to who, the request response API model will handle who to route these without having to store which user is which. Caching might expose some data, so we also won’t use it, plus caching won’t help us make that much sense figuring out if two images are the same is a whole problem itself.

Reliability

Reliability isn’t a very clear concept when you compare it to performance, security and these kinds of things in the system; meaning that it’s not a very initiative property to understand. So to lay some groundwork and make sure we are on the same page of what reliability is we will start with a definition,

Reliability refers to the probability that a system performs as it's designed, correctly, during a specific time duration, or a period of time, and during this time no repair should be performed, and the system adequately follows the defined performance specifications.

Reliability is a must in any system in this day and age, our system is expected to perform at its top performance at all times, even when the load is at its peak. That is one of the many reasons we chose to be cloud native, which means run everything on the cloud, so we don't need to handle server maintenance, network recovery and what not. This way we can almost guarantee that we have high reliability probability and operate as we should.

Availability

This property is sometimes confused with the reliability property, although all properties are connected in some way or another, but still every one of them has its own definition and how they relate to others; meaning that they are not the same of course.

Availability is the probability that a system performs as it's designed, correctly, at a specific point of time, time instance. Some things might go wrong before or after the point of time (where we calculated the availability), but the system MUST be operational, and adequately satisfy the defined system specifications.

So, Availability is a very critical concept in any system, especially in our system, users can't afford having the system down when they're enhancing a video or an image, the same way we can't afford bad user experience.

Scalability

Having a system that is built for a big user base is very important, however, not all products are Google products where the starting number of users are more than a million users. Instead, normal products start with zero users, or the number of developers who are testing them, so having a system that is already scalable can be expensive, both in time and money.

Vision, the system, is not build for a big user base from the start, but is build to grow as the user base grows, I like to call it the infinite ballon scale, where the system is a balloon that can get as big as you want (infinite), and as the load grows, the balloon grows, and as the load decrease the balloon decrease its size as well. This way we make sure we can handle our system peaks like a piece of cake, and handle when the system doesn’t need to handle a lot of requests and reduce system resources such as CPU power, memory, disk I/O, and more.

Maintainability

Maintainability is something that many software engineers leave behind in many projects, it doesn’t get that much attention. Nonetheless, we will make sure our system is maintainable to make sure that if we want to extend some extra features, or make sure that other people understand how the system works and add their own touch of creativity to the system.

It’s also important to make sure the system itself is maintablite, meaning that deployment, bug fix, and stuff like that should also be easy to maintain in the future. This way we handle the technical side of maintainability as well as the development side of it.

Use Cases

Image Enhancer

Use Case	Image Enhancing
Actors	User
Purpose	Improve the quality of the given image by boosting its resolution.
Overview	The user has an image, and decides to use our product in order to enhance the image. So the user opens our application, or our website, and uploads the image anonymously, the system sends this image to our back-end, processes it, and responds to the user with an enhanced image.
Type	Primary
Cross references	Functions: **V-01**

Video Enhancer

Use Case	Video Enhancing
Actors	User
Purpose	Improve the quality of the given video by boosting its resolution.
Overview	The user has a video, and decides to use our product in order to enhance the video. So the user opens our application, or our website, and uploads the video anonymously, the system sends this video to our back-end, processes it, and responds to the user with an enhanced video.
Type	Primary
Cross references	Functions: **V-04**

Image Denoising

Use Case	Image Denoising
Actors	User , System Processes
Purpose	Remove noise from a given image.
Overview	The actor has an image, and decides to use our system in order to remove noise from the image. So the image is uploaded to the system, the system sends this image to our back-end, processes it, and responds with a denoised image.
Type	Primary
Cross references	Functions: **V-00**

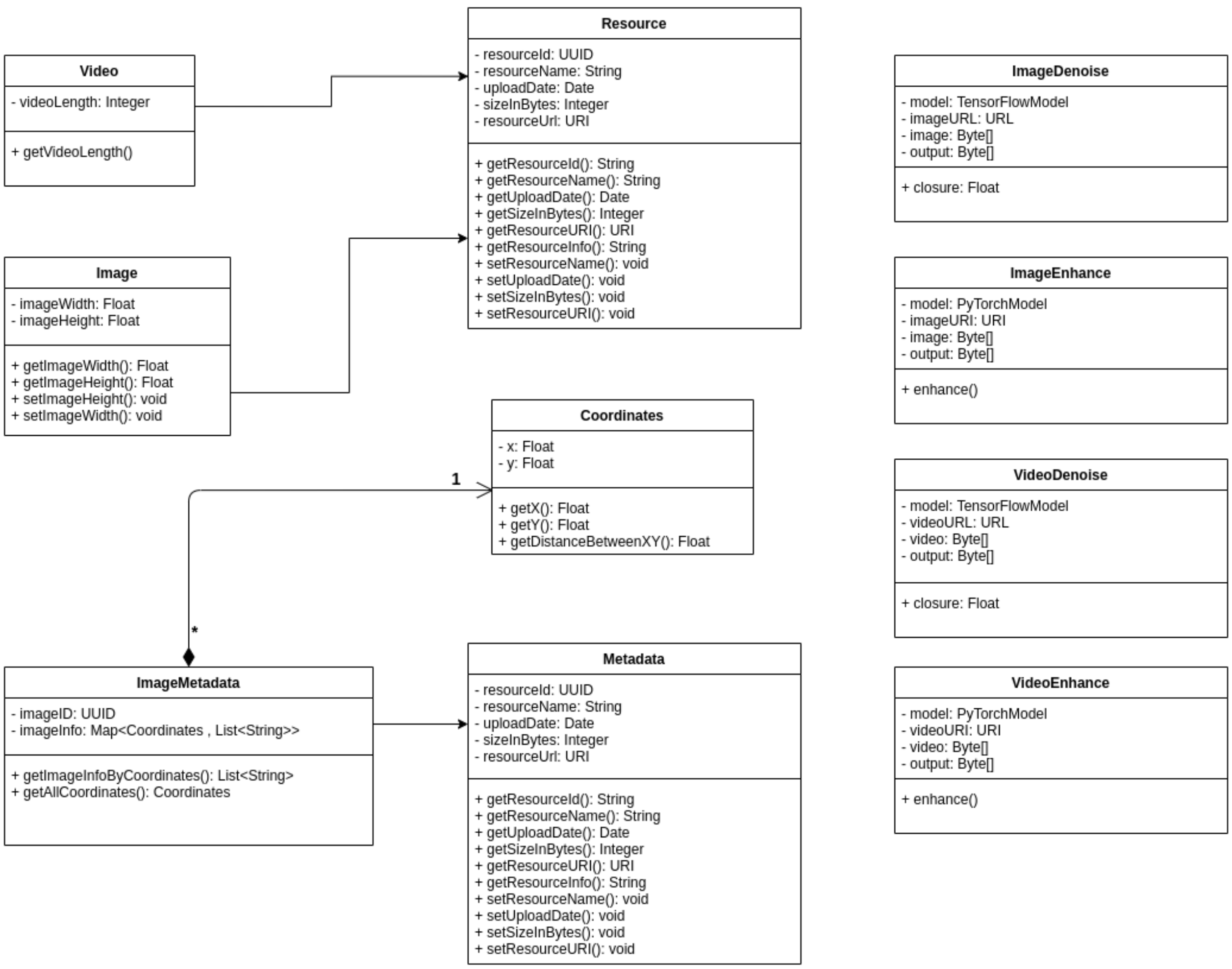
Video Denoising

Use Case	Video Denoising
Actors	User , System Processes
Purpose	Remove noise from a given image.
Overview	The actor has a video, and decides to use our product in order to remove noise from the image. So the user opens our application, or our website, and uploads the video anonymously, the system sends this video to our back-end, processes it, and responds with a denoised video.
Type	Primary
Cross references	Functions: **V-03**

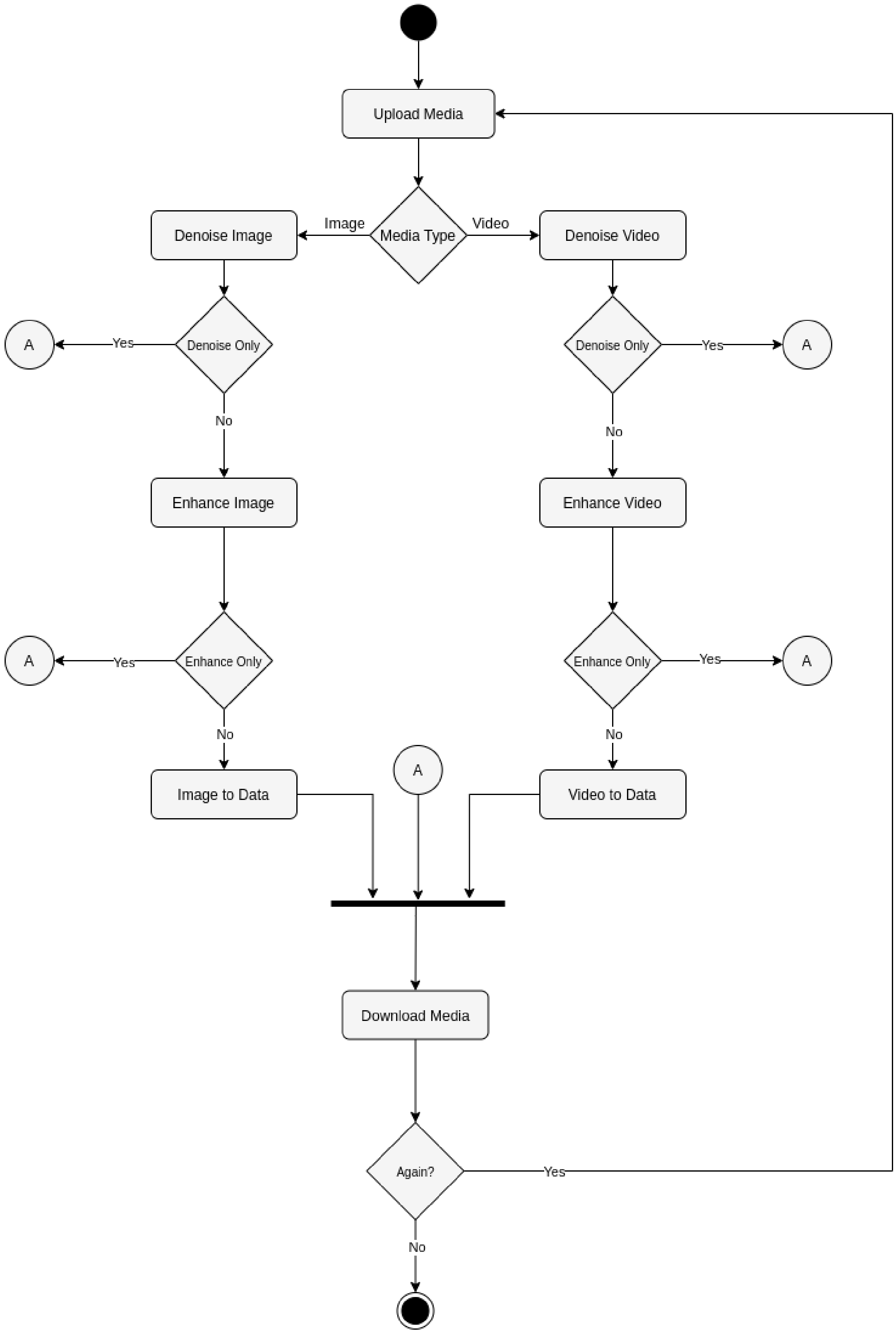
Image to Data

Use Case	Information from an Image (Image to Text)
Actors	User
Purpose	Get extra information from a given image.
Overview	The user has an image and wants to get extra information about the image, i.e: what kind of objects are there in the image, licence plate detection, face detections, and more. So the image is uploaded to our system by the user, anonymously of course, the system sends it to our back-end, processes it, and returns with extra information about the given image using JSON format.
Type	Primary
Cross references	Functions: **V-02**

Class Diagram



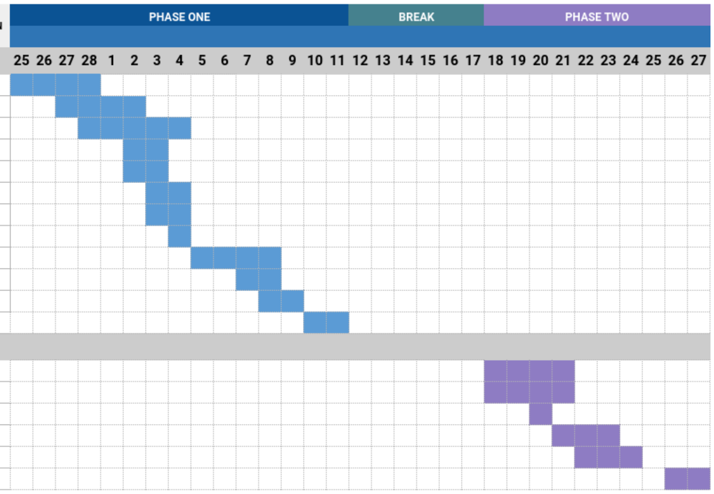
Activity Diagram



Gantt Chart

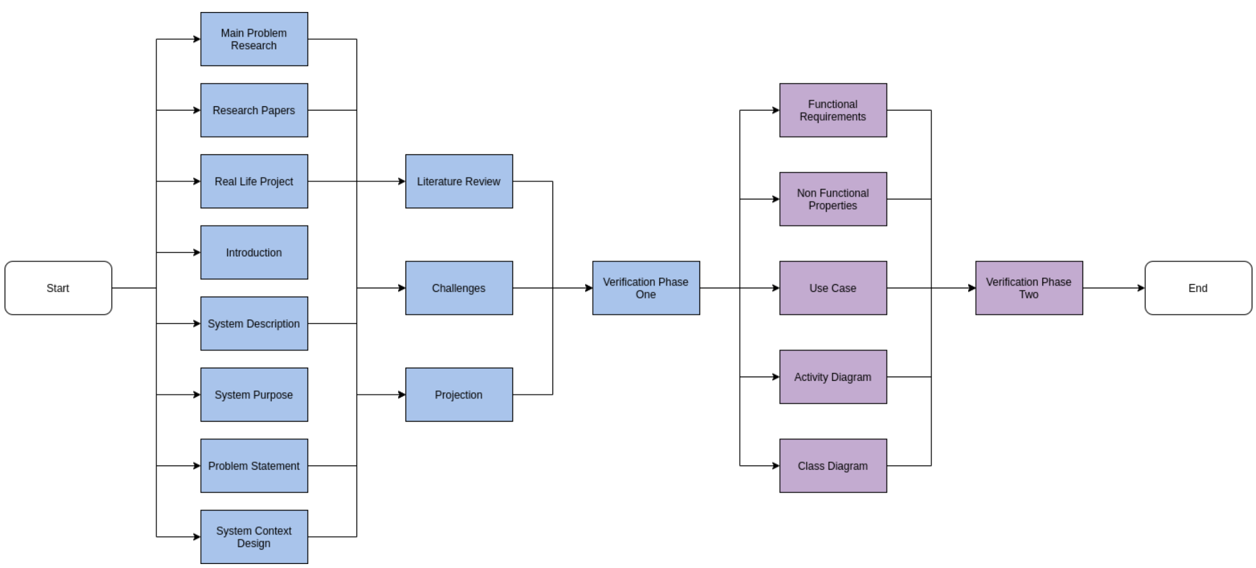
Gantt Chart

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
1	**Project Research**			
1.1.0	Main Problem	25-Feb-2021	28-Feb-2021	4
1.1.1	Research Papers	27-Feb-2021	02-Mar-2021	4
1.1.2	Real life projects	28-Feb-2021	04-Mar-2021	5
1.2.0	Introduction	02-Mar-2021	03-Mar-2021	2
1.2.1	System Description	02-Mar-2021	03-Mar-2021	2
1.2.2	System Purpose	03-Mar-2021	04-Mar-2021	2
1.2.3	Problem Statement	03-Mar-2021	04-Mar-2021	2
1.2.4	The System Context View	05-Mar-2021	05-Mar-2021	1
1.2.5	Literature Review	05-Mar-2021	08-Mar-2021	4
1.2.6	Challenges	07-Mar-2021	08-Mar-2021	2
1.2.7	Projection	08-Mar-2021	09-Mar-2021	2
1.2.9	Verification	10-Mar-2021	11-Mar-2021	2
2	**Design and Analysis**			
2.1.0	Functional Requirements	18-Mar-2021	21-Mar-2021	4
2.1.1	Non-Functional Properties	18-Mar-2021	21-Mar-2021	4
2.1.2	Use Case	20-Mar-2021	20-Mar-2021	1
2.1.3	Class Diagram	21-Mar-2021	23-Mar-2021	3
2.1.4	Activity Diagram	22-Mar-2021	24-Mar-2021	3
2.1.5	Verification	26-Mar-2021	27-Mar-2021	2



Vision

Network Diagram



Referencing & Citation

- Tassano, Matias, Julie Delon, and Thomas Veit. "Fastdvdnet: Towards real-time deep video denoising without flow estimation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.
- Fan, L., Zhang, F., Fan, H. et al. Brief review of image denoising techniques. Vis. Comput. Ind. Biomed. Art 2, 7 (2019). Brief review of image denoising techniques | Visual Computing for Industry, Biomedicine, and Art | Full Text
- Convolutional neural network
- SIDD-Full Dataset
- NVlabs/ffhq-dataset: Flickr-Faces-HQ Dataset (FFHQ)
- DIV2K Dataset
- Wang, Xintao, et al. "Esrgan: Enhanced super-resolution generative adversarial networks." Proceedings of the European Conference on Computer Vision (ECCV) Workshops. 2018.
- Romano, Yaniv, John Isidoro, and Peyman Milanfar. "RAISR: rapid and accurate image super resolution." IEEE Transactions on Computational Imaging 3.1 (2016): 110-125.
- Generative adversarial network
- Ian Goodfellow
- Amazon Rekognition – Video and Image
- Computer vision
- DAVIS: Densely Annotated VIdeo Segmentation
- https://cv.snu.ac.kr/research/EDSR/Flickr2K.tar
- Video Enhancement with Task-Oriented Flow
- Chadha, Aman, John Britto, and M. Mani Roja. "iSeeBetter: Spatio-temporal video super-resolution using recurrent generative back-projection networks." Computational Visual Media 6.3 (2020): 307-317.
- Haris, Muhammad, Gregory Shakhnarovich, and Norimichi Ukita. "Recurrent back-projection network for video super-resolution." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
- SULTANA, ARIFA & Sarma, Kandarpa. (2021). WEINER FILTER AND SUB-BLOCK DECOMPOSITION BASED IMAGE RESTORATION FOR MEDICAL APPLICATIONS.

- Linear degradation model for estimating remaining useful life - MATLAB
- TensorFlow.org