CARNEGIE MELLON UNIVERSITY

School of Architecture
College of Fine Arts

Thesis

Submitted in Partial Fulfillment of the requirements for the degree of

Master of Science in Computational Design

TITLE:

Image-based Light Modification
A Machine Learning Approach to 'Content-aware' Image Processing

AUTHOR:

Zheng Luo

ACCEPTED BY ADVISORY COMMITTEE:

Daniel Cardoso Llach [redacted]		May 16, 2019
Professor NAME	Principal Advisor	DATE

Image-based Light Modification

A Machine Learning Approach to 'Content-aware' Image Processing

by Zheng Luo

Abstract

Designers translate their ideas into images via visualization tools. Most of these tools provide some level of editing capabilities. Although these tools have very powerful and comprehensive capability in editing, they require users to have a strategy before using and decide the combination of commands he/she needs to apply. For some simple task, for instance, moving the position of the light source in an image, these tools will have very limited power (image processing) or require great amount of computational resources (render). Even though the user may have a clear idea of the expected result, current tools will have a hard time to accomplish the task, mostly because they are 'content-blind'. To test the idea of a 'content-aware' image processing tool, in this research I used machine learning methods to develop a proof of concept tool that is able to rapidly relight an image based on content traits in the image (depth, shape, reflectiveness, etc.) that relate to the re-lighting task. By doing this, we can have the light weight convenience of an image processing tool as well as the content-awareness of a rendering tool.

Acknowledgements

First, I would like to thank Prof. Daniel Cardoso Llach not only for advising my thesis but more importantly, for the support and mentorship over the past two years. It was incredibly fortunate to be your student and to have you as my thesis and academic advisor.

Second, I would like to thank all my cohort. CodeLab is such a warm place with all the talented people. I will never forget all the brainstorm, late night working, and mostly all the fun we had.

Finally, I would like to express my deep gratitude for the support from my family and friends, especially my wife. It was impossible for me to go through all the difficulties and frustrations for the past year working on my thesis without you.

Table of Contents

1. Introduction: 6
   1.1 Limitation of current visualization tool. 7
   1.2 Smart filter 8
   1.3 Relighting task 9
2. Background: 10
   2.1 Light in architecture design and the light modification challenge. 10
   2.2 Visualization before computers. 10
   2.3 Relighting challenge 13
   2.4 Light simulation in render 14
   2.5 Image-based editing 16
   2.6 An old problem with a new approach, deep learning 17
   2.7 Deep learning in image process 20
3. Hypothesis: 23
4. Method. 25
   4.1 Data Pipeline 25
   4.2 The Simple Box Task 26
   4.3 What is GAN? 27
   4.4 Sphere with bump surface 31
   4.5 New dataset 33
   4.6 Single decoder model 34
   4.7 U-net(Ronneberger, Fischer, & Brox, 2015) 36
   4.8 Generalization: 41
5. Evaluation: 43
   5.1 Image quality: 43
   5.2 Limitation: 44
6. Conclusion 45
   6.1 Contribution: 45

6.2 Future work: 46
6.2.1 Modulate data pipeline: 46
6.2.2 Improve accuracy: 46
6.2.3 Extend the approach to other application: 46
Reference: 48

1. Introduction:

This thesis explores the possibilities of machine learning, deep learning to be more specifically, into visualization process, specifically focusing on image-based light modification which allows the user to modify light in an image without creating 3D digital model of the scene or using complex image processing commands.

In mainstream design processes, designers use 3D modeling tools and post-processing tools to receive visual feedback about theirs design decisions and modify their design either in model space or directly on images. Modern image processing tools, for example Photoshop, provide great number of operations and almost infinite number of combinations. Designers are assumed to be familiar with most of these operations and have the ability to construct a clear combination of correct operations for the tasks they are facing. For some tasks this assumption may not be true even if the designer has a clear understand of the expected outcome. Under this circumstance, the powerful image processing tool becomes useless.

Therefore, I would argue that introducing deep learning into the process is necessary. With rapid development in recent years, especially convolutional neural network architecture, deep learning has been used in many image related tasks such as image classification, object detection, image segmentation. This shows that proper deep learning architecture has the potential to abstract information from the content in the image, and apply modifications based on that.

Essentially, deep learning, as a universal function approximator, has the potential to find non-linear relationships between input and output. This provides possibilities to skip the step of planning existing image processing commands.

In this thesis, I propose an image processing tool and show the steps to solve a relighting problem that is hard to do with existing image processing tools, using deep learning. Through this demonstration, we can see that using deep learning could bring new possibilities in visualization. Compared with simulation and mainstream image processing tools, we will see how deep learning can be 'content-aware', abstracting information directly from image. We can also see that how users can use this workflow to create their own 'smart filters'.

1.1 Limitation of current visualization tool.

Visualization processes in architecture can be generalized into three steps: 3D modeling, rendering and post-processing. Each of these steps is in charge of a specific goal in the visualization. 3D modeling collects the data from design inputs, and translates them into a visible spatial model. This step gives spatial feedback to designers, also prepares parameters for the following stage. The render uses model data (geometry, material, environment setup, etc.) and translates these to high resolution, most of the time photo-realistic images. These images will then be used in post-processing as input. The post-processing, most of the time is image processing, gives the final touch to the image. Image processing provides modifications that are hard to do in rendering. Typically, render provides the baseline image to work with and image processing provides more on the expression and modification that are more specific and characteristic.

The modeling and rendering usually consume a great of computational power. Even with advanced GPU, rendering typically takes long. Even if a slight modification after the rendering is finished, the process has to start from the beginning. An architecture image rendering task along can easily last for hours, which makes the process extremely expensive.

On the other hand, even though image processing comprises mostly real-time operations, it doesn't provide much flexibility as the render. The quality of operations depends on the user's understanding of image, commands to use and sequence of using them. Most of the operations do not analyze anything in the image but rather simply apply mathematical transformation on pixels.

1.2 Smart filter

The concept of smart filter is essentially an image-based modification algorithm. Similar to the filters in Photoshop, smart filters work directly on an image, provide certain transformations on pixels. Compared to regular filters, smart filters have the ability, in some extent, to understand the content in the image and provide different modification to different pixels or regions. This provides us the possibility of avoiding running simulations, yet still able to make modifications based on different materials in the image. Notice that the information of materials used to be accessible only in the simulation, but now the smart filter has the ability to 'understand' the materials directly from image. In this way, the filter can stay computationally light weight and provide more flexibility in modification at the same time.

1.3 Relighting task

To show how smart filters work differently with current image processing tools and simulation tools, I set up a relighting task with different scenes. The task is to move the light source in an image to a new position and show the corresponding new image. Lighting is essential but complex in design and often requires understanding of the design content. We will see differences between simulation tools and 'content-blind' image processing tools handling this task, as well as the analysis of the advantages and disadvantages of mainstream methods. We will also see how deep learning fits in this task and how being 'content-aware' can be beneficial.

2. Background:

2.1 Light in architecture design and the light modification challenge.

> *"Architecture is the masterly, correct and magnificent play of masses brought together in light. Our eyes are made to see forms in light; light and shade reveal these forms; cubes, cones, spheres, cylinders or pyramids are the great primary forms which light reveals to advantage; the image of these is distinct and tangible within us without ambiguity. It is for this reason that these are beautiful forms, the most beautiful forms. " – Le Cobusier, (Cobusier, 1986)*

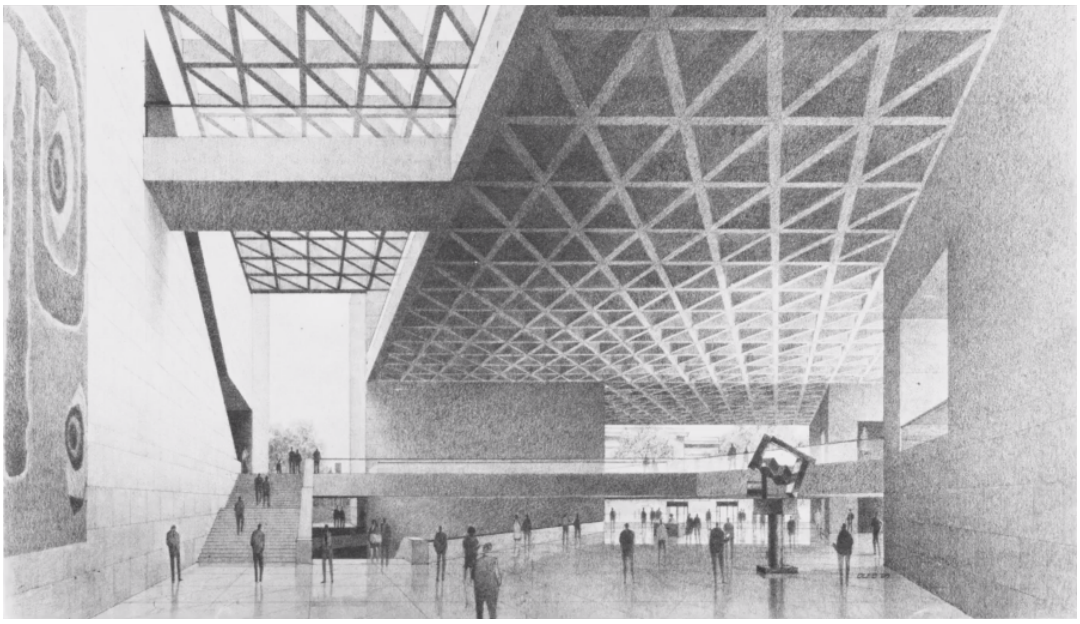
The study of light in architecture has a long history and it is commonly considered as one of the essentials in not only architecture but also design in general. Yet light is one of the most complicated components to express. We don't see light directly, rather we receive it by the reflections on surrounding objects. This makes it even more complicated since different shapes, materials and transparencies will lead to different responds. Blurry, translucent, gloomy, relucent, polychrome… All these words show how dynamic light can be and how hard to define it.

2.2 Visualization before computers.

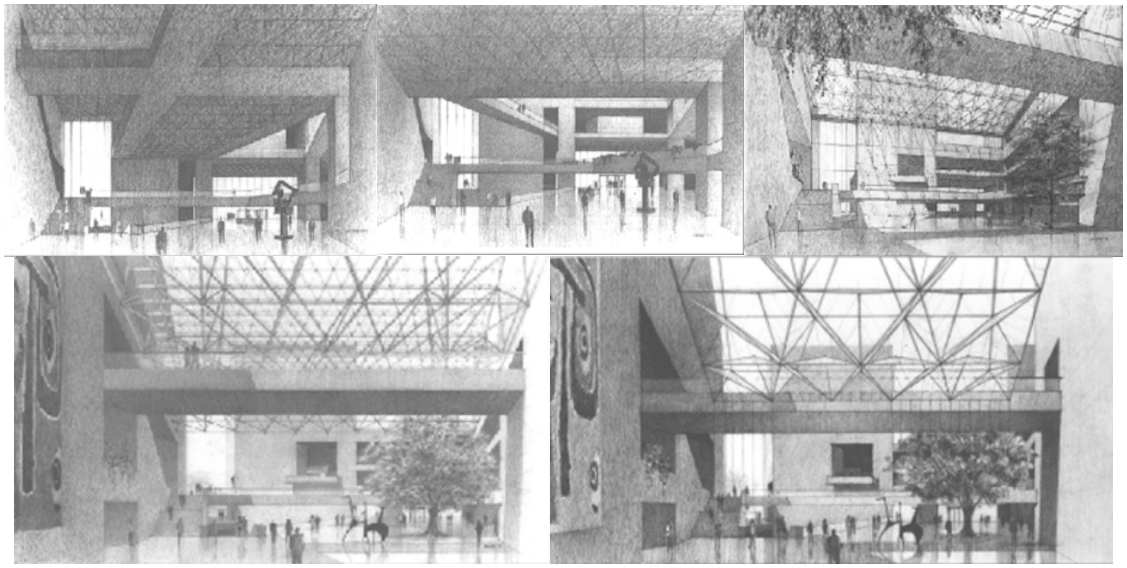
Before the age of computers, visualization was mainly done by drafting, hence it is obvious that good drafting skills and the ability to image the design in real-world environment is crucial to an architect. In fact, some architects are famous for their drafting skills, that they can 'foresee' what the building looks like. One of the examples would be Paul Steven Oles.

> *"An architectural project always began as a very vague and amorphous idea in architect's head. The task of architectural illustrator, in general, is to take all of the material that is available at the time, and translate that into something that is representational, photorealistic if you will, something that people could understand and the architect him/herself can understand as well."* – Paul Steven Oles(Wit House Productions, n.d.).

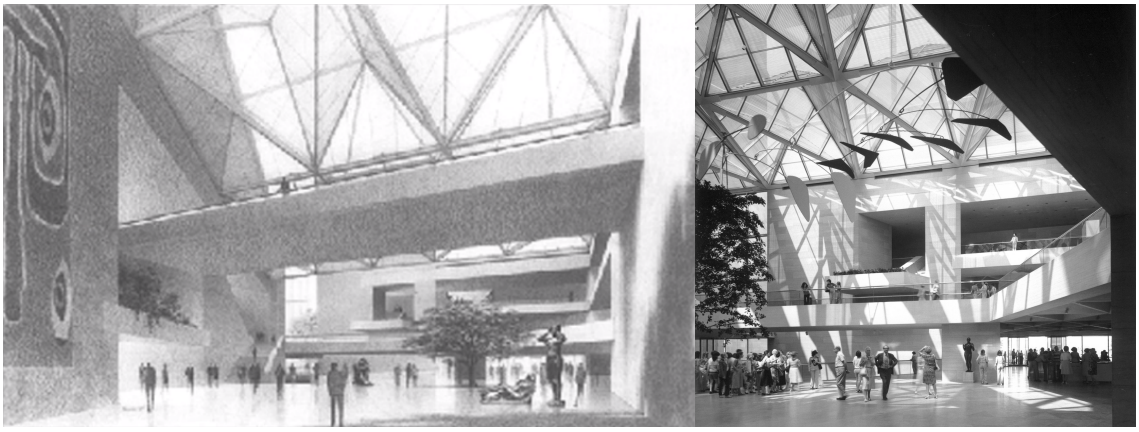
Oles is a professional architect and mostly famous because of his drafting works in the project National Gallery of Art, East Building designed by I. M. Pei. In 1969, soon after Oles joined the project he delivered his first sets of drawings of the building. Even though the clients were very satisfied with the design, Pei was not happy with the lighting presented in the drawings. This led to a series of changes in the design and ended up with the enormous skylights in the lobby.



Initial lobby drawing, 1969(Oles, 1979)



Drawings in the design process(Oles, 1979).



Left: Final drawing, 1972. Right: Photo of the lobby after built(Oles, 1979).

Comparing with the photograph on the right, we can see that how accurate Oles was able to present the interior of the lobby and why his work was essential in the process of design iterations. Visualization confirms the design intention and from there directs the changes. It is easy to realize that as a designer, he/she would want the visualization process to be fast enough to get the feedback sooner, and flexible enough to be accurate, photorealistic or expressive if necessary.

Nowadays most of the visualizations are done using digital tools. Here we compare two possible ways to tackle the relighting task, simulation and image processing.

2.3 Relighting challenge

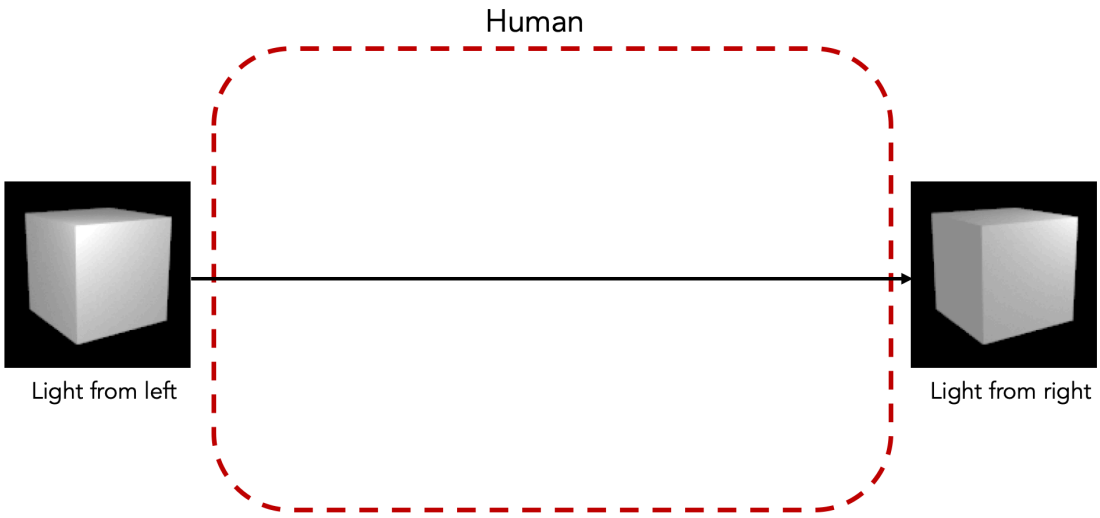
To see how lighting in visualization can easily become a complicated problem, and to compare typical strategies designers use to tackle these problems with modern visualization tools, I created a relighting challenge. The challenge is as follow: given an image of a scene, with clear definition of the light source, how would one create an image of a novel light source.



Relighting challenge: The left image shows a cube in the scene and given constraint of light as "from the left of the cube". The challenge is to modify this image to the target constraint as "from the right of the cube".

As we can see from the example above, the original image contains a scene of certain lighting setup. In this scene, we have a cube placed in the center of the view. The light setup is given as 'from the left of the box'. Although we cannot see the light source directly, we can prove it by observing the two surfaces in the

front that is lighted up. The challenge is to give the correct image of the new lighting: 'from the right of the box'.



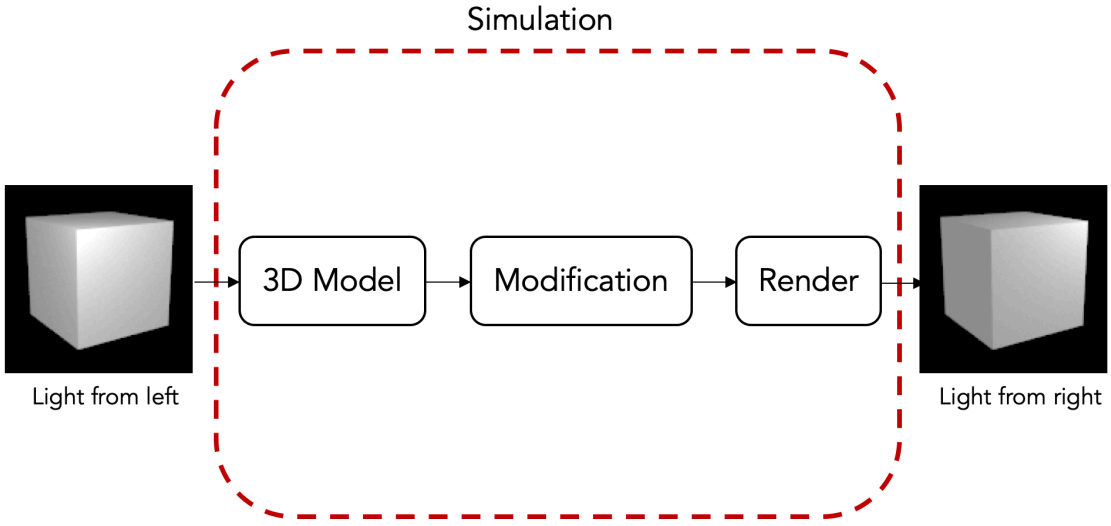
Human can process this image intuitively

Human beings have the ability to intuitively understand the content of the image and create the image of the correct lighting almost instantly in their heads. If one is trained, he/she might easily draw out this image. As mentioned in Oles' story, drawing was the dominant way of visualization before the existence of digital tools.

2.4 Light simulation in render

To tackle the relighting challenge given above, one of the possible ways is through simulation. The mainstream method to construct a simulation is to use 3D modeling and rendering tools. 3D modeling tools collect data of the scene and display it through the monitor. For the given relighting problem, the user needs to create a cube object in the model space, along with a light on the left of the cube. To get the desired image, one only needs to move the light in the

scene to the right side of the cube. Then the digital model will be passed to the rendering engine to produce the final image.



Steps to take using simulation to modify a light in an image.

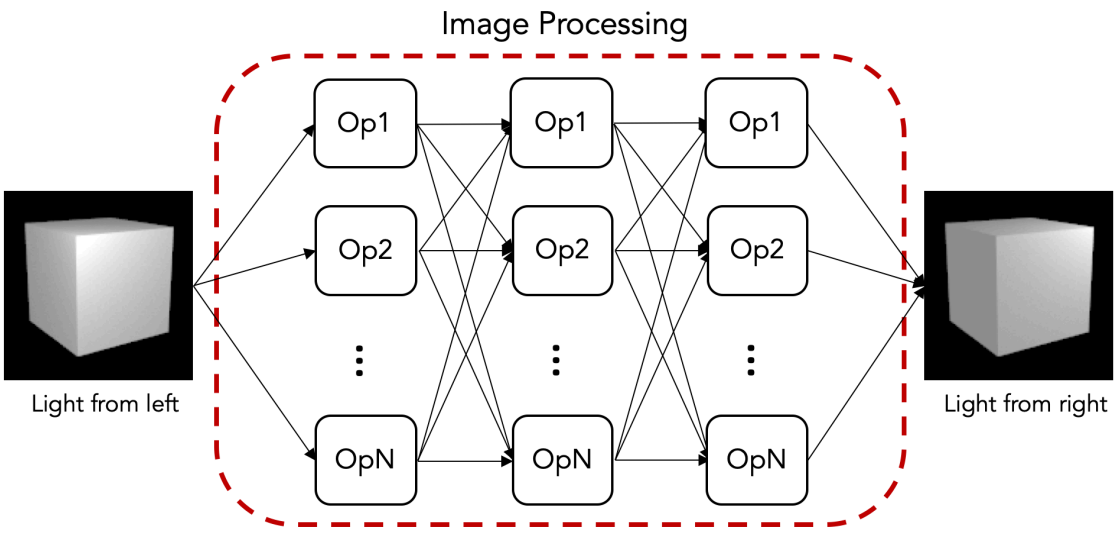
The advantage of this process is flexibility. Because we have the 3D model, we can apply any modifications as needed. Keep in mind that the simulation process typically serves many purposes at the same time and not only just for light modification: researching, prototyping, visualization, documentation, etc. It's clear that the process is an overkill for a simple challenge of relighting, but it's capable of accomplishing this task and indeed the mainstream method in design to study or modify light.

Two disadvantages can be observed in this process. First, this is a complex and computationally heavy process. This is understandable given the many tasks it serves and the flexibility it can provide. Software and hardware companies are providing more convenient modeling tools to use for faster prototyping and more powerful GPU to shorten the time of render. Second, this process assumes the model's existence. The 3D model can be acquired in two ways: either the user

could build it from scratch or import it from other resources. But the entire process is built upon the 3D model. If the model doesn't exist and we only have an image as the input, simulation has no use in this circumstance.

2.5 Image-based editing

The other possible way to do this challenge is to use image processing tools. Photoshop, as the most commonly used image processing software, provides a great number of commands to apply to an image. Most of these software use 'layers' and 'masks' to select the regions the user wants to modify, and apply built-in filters on these regions. The huge collection of filters is one of the key reasons that make these tools powerful. However, for the purpose of relighting, no single filters are suitable to tackle the task.



Steps to take using image processing software to modify a light in an image.

The above process shows one possible way to finish the task using image processing software. Inevitably, one would have to 'draw' some of the pixels in

order to create the correct reflections and shadows on each surface. This 'drawing' action, to some extent, brings us back to the time before computers, only except we are now drawing using digital tools rather than pen and paper.

Inherently, filters can be seen as a mathematical transformation applied to a region of pixels. Therefore, most of these filters are 'content-blind'. Any of these filters don't have the ability to recognize information in the image (shape of object, materials, bumpiness, etc.). The tool is designed in the assumption that the user knows where to apply these filters and how to apply them appropriately. This transfers the task of analyzing to the user. In order to use these to get the correct image, user needs to have some level of understanding of the consequences of most of the filters, and also processing the image in his/her head and translate this process into a series of operations to take using the software provided filters. Even for an intuitive task like this one, the quality of the outcome still relies on the user's understanding not only to the image, to lighting physics, but also to the tools he/she is using. This becomes a drawback for image processing.

2.6 An old problem with a new approach, deep learning

Simulation is computationally expensive but most flexible, while image processing is lightweight but 'content-blind'. Using deep learning, we can find a way to combine the advantages of both and create 'smart filters' that are 'content-aware' but still lightweight as existing filters.

Deep learning has two advantages for this task: first, deep learning is capable of finding connections between input and output with little interference from

humans, even when the connections are vague. In the relighting task, the input is the original image, the original light position, and the target light position. The output is the image corresponding to the target light position. The relationship between them is a transformation function that maps each pixel value in the original image to the new one. This relationship is hard to describe or calculate, yet it exists; second, it has the ability to grasp information from the image directly and apply modification accordingly. This is mainly due to the convolutional operation which will be discussed in detail further in this thesis. This advantage brings the possibility that the algorithm can automatically find the correct way to modify input image into the target image, whereas before it depends on human to design the strategy of modification.

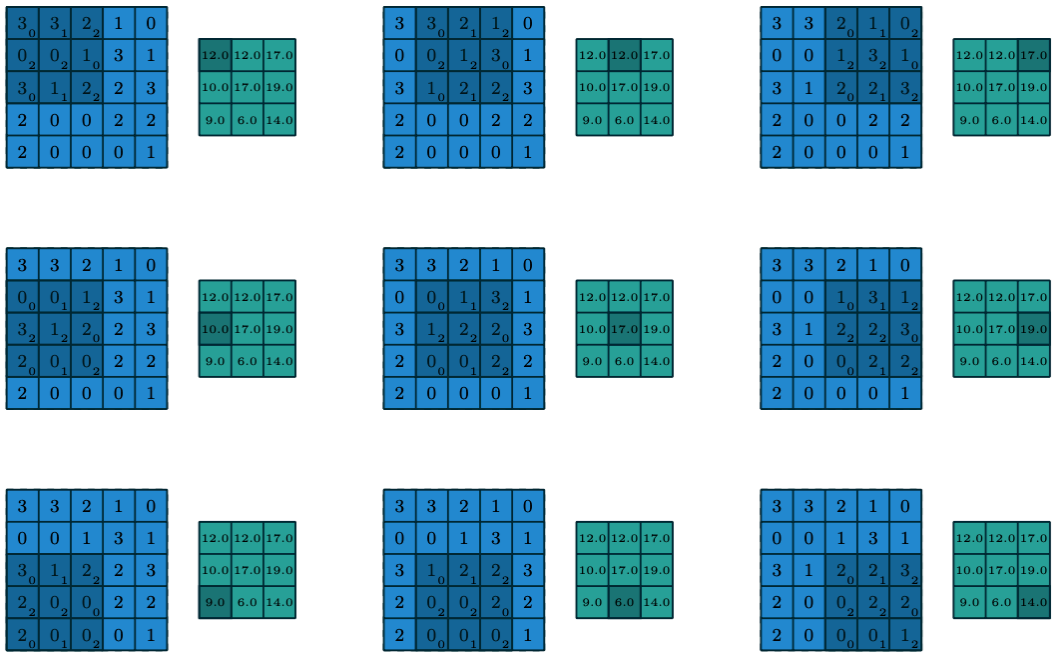
A deep learning model is essentially a universal function approximator. We can understand the relighting filter as a function. This function takes in an image, its original lighting setup and its target lighting setup as inputs. Through some calculations, this function will generate a new image as output. To be more specific, this function constructs a non-linear relation between the original pixel values and the new pixel values, given the target light position as a condition. This function exists in a high dimensional space (often known as latent space) and is hard to define. A deep learning model will learn this function through iterations.

$$I_{c'} = F(I_c, c, c') = F(\mathcal{P}, c, c')$$

Hypothesis transformation function F(•). The function takes in an image under constraint c, the original constraint c and the new constraint c' as input, yield new image as $I_{c'}$. The original image is represented as a collection of pixel value P.

The learning starts with the model generating an image under a given light position. At the early stage of training, this image is often wrong. In the training process, the image will then be compared with the correct image, the ground truth, and calculate how much difference they made. This difference, known as the loss, is calculated by the loss function. It is obvious that at the beginning the loss is quite large. This loss is caused by the wrong parameters of the model and each parameter contributes a certain portion of this loss. Given that, we can distribute this loss to each parameter by how much they contribute and change the parameter accordingly. Eventually, we play and tweak these parameters until they make the least loss. This is the essence of how supervised learning works.

A common component in most modern neural networks is convolution. The idea of convolution came from signal processing, where a carefully designed feature filter is applied and scan through the data, and the regions that match with the feature will be selected. In deep learning, this process is done by convolutional layers. The parameters in the convolutional layers are trainable, meaning that in a convolutional network we don't need to design the filters. Through training, the model will learn what are the most useful features and use them to grasp information from the image. This is inherently how deep learning is able to be 'content-aware'.



Demonstration of convolutional operation on image(Dumoulin & Visin, 2016)

2.7 Deep learning in image process

The use of deep learning has been growing rapidly for the past few years and quickly became dominant in all kinds of image tasks. In 2012 AlexNet[1] was published and won the ImageNet classification competition. It showed the potential of model learning the feature by itself rather than human feature engineering. The paper opened the door of utilizing deep convolutional architecture. After that, deep learning has been used in different areas including image classification, image segmentation, image generating, style transfer. Due to the fast-growing computational power and massive data, some of the deep learning models can go beyond human-level performance.

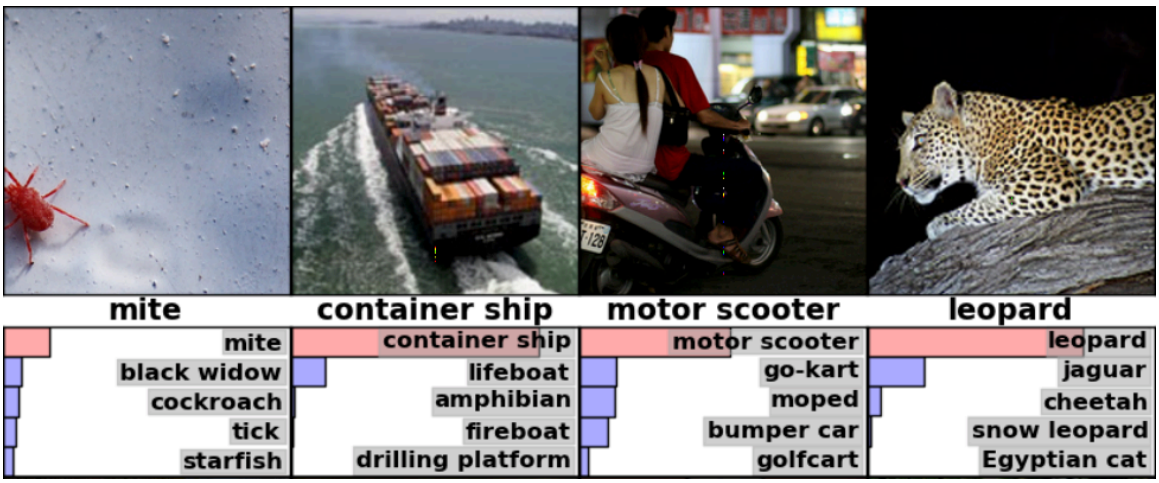


Image classification(Krizhevsky, Sutskever, & Hinton, 2017)

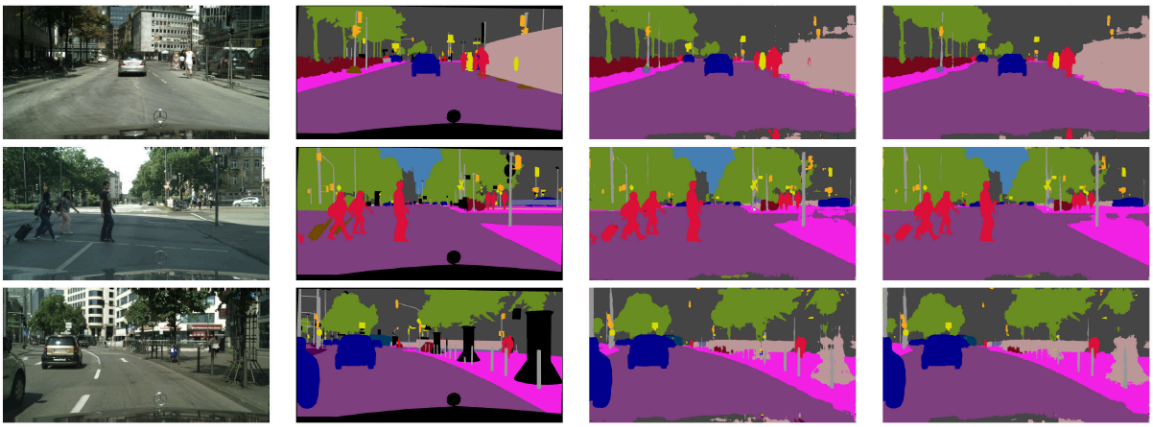


Image semantic segmentation(Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2016)

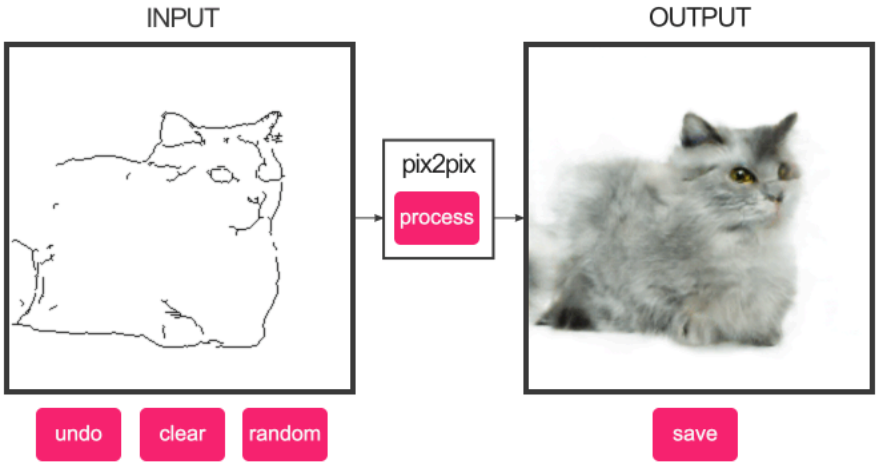
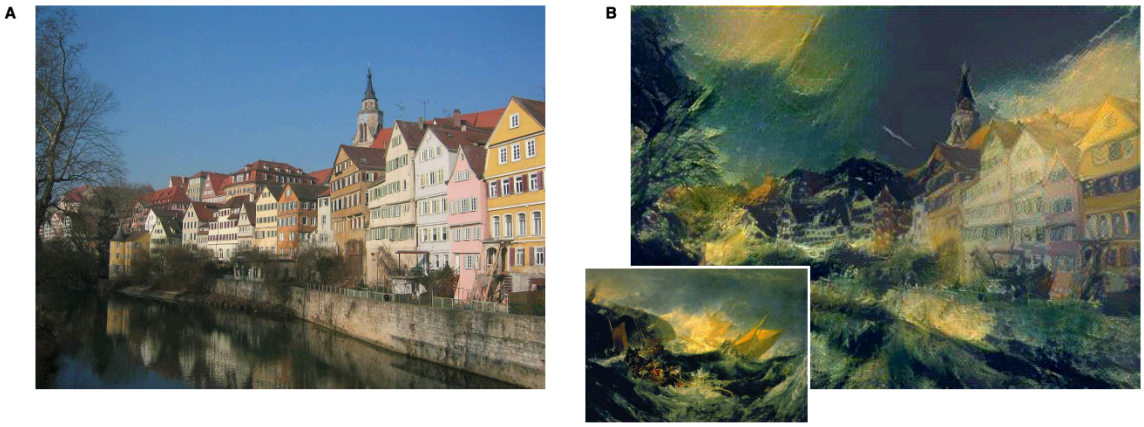


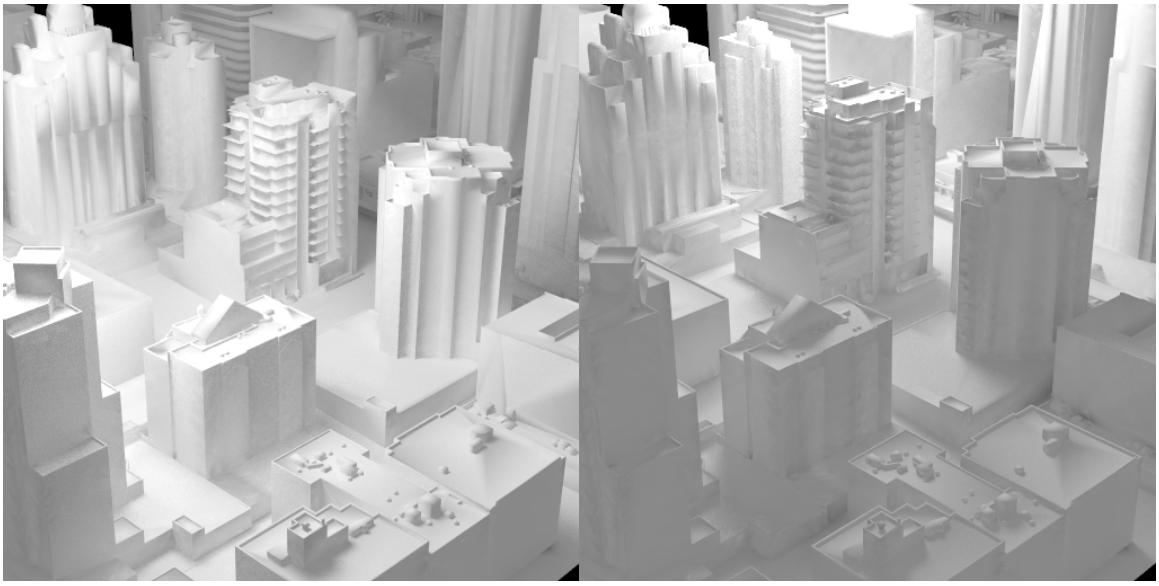
Image generation(Isola, Zhu, Zhou, & Efros, 2016)



Style transfer(Gatys, Ecker, & Bethge, 2016)

3. Hypothesis:

My hypothesis is that the relighting challenge can be accomplished using deep learning. The vision of this thesis is to implement a system that uses machine learning to modify the light condition in an image based on user-provided constraints. This is a fairly large task to tackle and can be further divided into two parts. First, we can build a machine learning algorithm that modifies the light condition in a particular image, in our case, modify the light position. Further, we can generalize this model so that theoretically it can be applied to an arbitrary image.



Example of apply light modification to the image.
Left: light from east. Right: light from west.

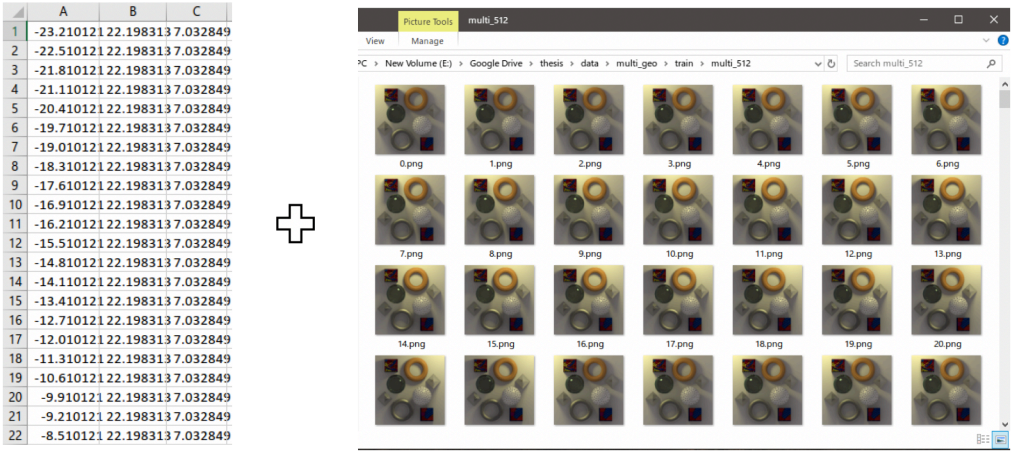
The generalization mainly depends on the diversity of the dataset. In other words, the more scenes the model has 'seen', the better the generalization will be. This requires a fairly large and diverse dataset to be built and typically will be conducted by firms or research organizations. In this thesis, I was more focusing

on the first part, how to build a model to move the light in a particular image. I demonstrated some attempts for generalization as well in order to show the potential.

4. Method.

4.1 Data Pipeline

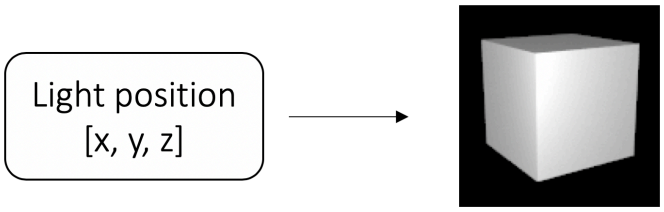
To train the machine learning model properly, I first need to build a dataset that is suitable for training a relighting task. Given the task is to move the light, we need the dataset to have the images which contain a source of lighting, as well as the light information from the scene, in this case, the position of the light. Some of the open-source datasets might be a good resource for this study. Yet there are some extra requirements for selecting a proper dataset besides solely images: it needs to have a large amount of data; the images are all for a fixed scene; if not a fixed scene, each scene needs to have at least two images for two different light conditions; the information of light is accessible (time, position, etc.) In order to quickly iterate through different designs of the machine learning model, I decided to use synthetic data. This gave me more flexibility in not only training time, but also evaluating the performance of the model.



Generated synthetic data. On the left are the light positions. On the right are the corresponding images.

The pipeline mainly contains three components: A Grasshopper point generator, VRay render engine and Python script. The Grasshopper script was used to generate any number of points in a designated region inside Rhino model space, either randomly or in a grid. In this way, I was able to control the total number of positions of the light, as well as how dense the points are distributed in the designated region either randomly or on the grid. The coordinates of these points will then be stored in a CSV file to feed into the machine learning model for training and indication of the position of the light when creating render. The render is executed by a Python script. First, the light position is read from the CSV file. The light in the scene will then be moved to that specific position. A render is then triggered, and the rendered image will be saved automatically. In this way, the order of the coordinates in the CSV file is the same order of the images so we can load them correspondingly for training.

4.2 The Simple Box Task



This task tries to find the connection between input light position and image

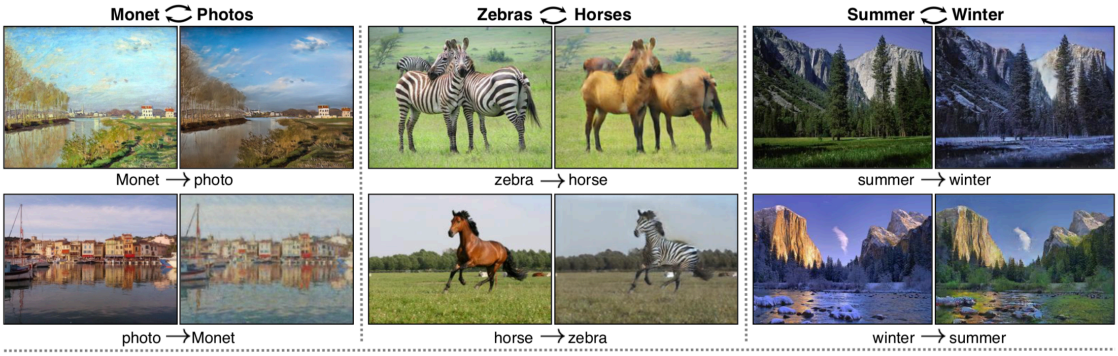
The first task I tried was to relight a box. In this task, there was only a simple box with no material or texture. The light will be randomly placed around the box. In this task, my main goal was to see if the relationship between the light position and corresponding reflection and shadow could be captured by the model. The model I chose was conditional deep convolutional GAN.

4.3 What is GAN?

Generative Adversarial Network(GAN) is a deep learning architecture introduced by Ian Goodfellow in 2014(Goodfellow et al., 2014). Ever since it's been invented, GAN has sparked many new innovations in the domain of deep learning. One of the advantages of GAN is its great ability to generate images. Researchers have been applying GAN in many tasks including but not limited to face generating, style transfer and so on(Junginger et al., 2018)(Zhu, Park, Isola, & Efros, 2017).

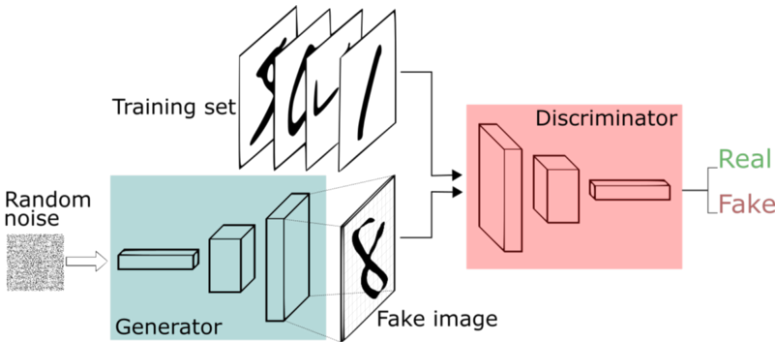


Faces generated using ProgressiveGAN(Karras, Aila, Laine, & Lehtinen, 2017)



Style transfer using CycleGAN(Zhu et al., 2017)

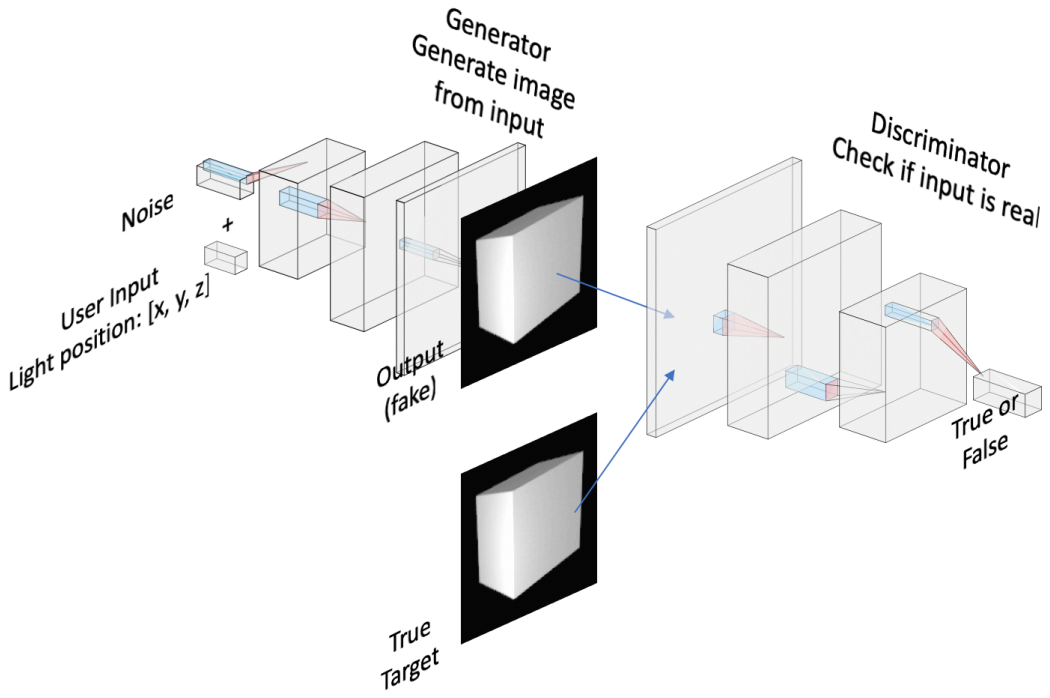
GAN consists of two deep neural networks that are often time referred to as the generator and the discriminator. The generator is a type of deconvolutional neural network that will create new images from the input noise vector, and the discriminator is a type of convolutional neural network that will determine its authenticity, or whether or not it belongs in a dataset. In order to fool the discriminator, the generator will have to learn to produce more realistic results, and the discriminator will also increase its ability to detect whether this result is a true image or a synthetic one. Thus, by competing with each other, two networks become more and more powerful. Ideally, we would want to end up with a generator that could trick the discriminator to have 50% of accuracy, which indicates that the discriminator can do nothing but randomly guessing whether the image is a true one or not.



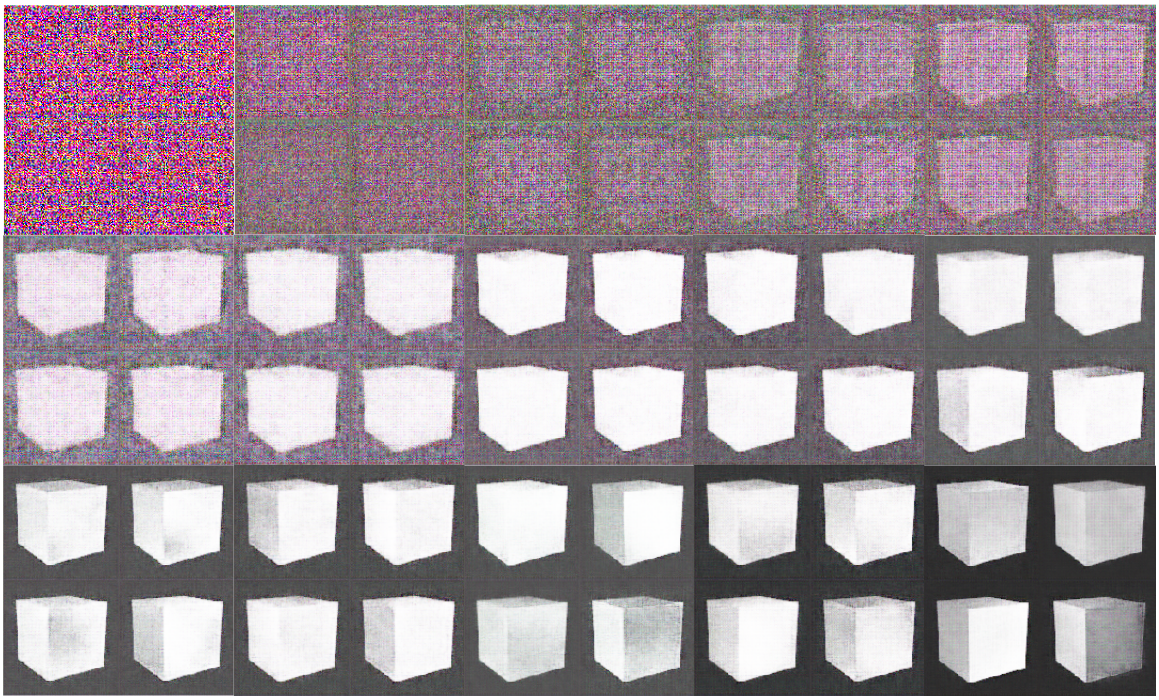
Demonstration of the original Generative Adversarial Network architecture used in generating handwritten digits. Image by Thalles Silva.

In the original GAN model, the input for the generator is usually a noise vector. Conceptually, this noise is the latent features of the images generated. One could think of this vector as a series of dials and knobs that control the content of the image. Although the meaning of each value is typically impossible to understand by humans, the machine learning model can learn it through training. Typically, the values of this noise are generated randomly, and the generator will create a random image as well, for example, random faces for a face generating GAN.

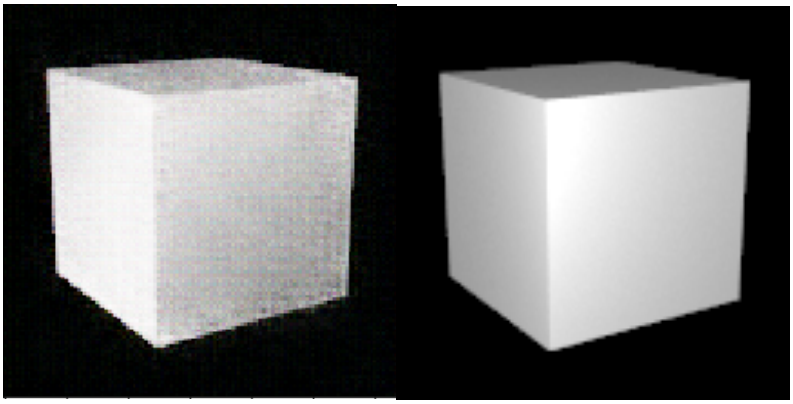
It is intuitive to realize that the light position vector is part of the latent vector. Thus, one of the modifications of my GAN model is to combine the light position with the noise vector. After proper training, the model should be able to generate the correct image by the given light position. In this way, we have the control of part of the latent vector, so that we can control the model to generate images that we want. In such a case, this position vector is often called 'condition', since our model is running a task 'under the condition of a specifically given directive'. Given that both the generator and discriminator are using multiple convolutional layers to encode and decode images, the architecture of the model is called conditional deep convolutional generative adversarial network(Radford, Metz, & Chintala, 2015).



The conditional DCGAN model I used in the box task. The light position is concatenated behind the noise vector.



The training process of the box task.

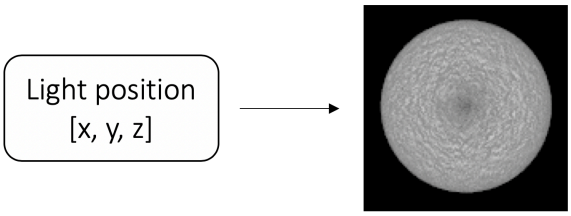


Left: Generated image. Right: Rendered image with same light position (ground truth)

After training on the box dataset, the model was capable of generating roughly accurate images. Compared with the ground truth image on the right, we can see that the model was able to generate correct light and shadow relations on different surfaces. We can see that since the light came from the front left of the cube, the front two surfaces were slightly lighter than the top surfaces. We can also observe that the generated image was very noisy. These noises were mainly due to the architecture of GAN, since it tried to generate all information purely

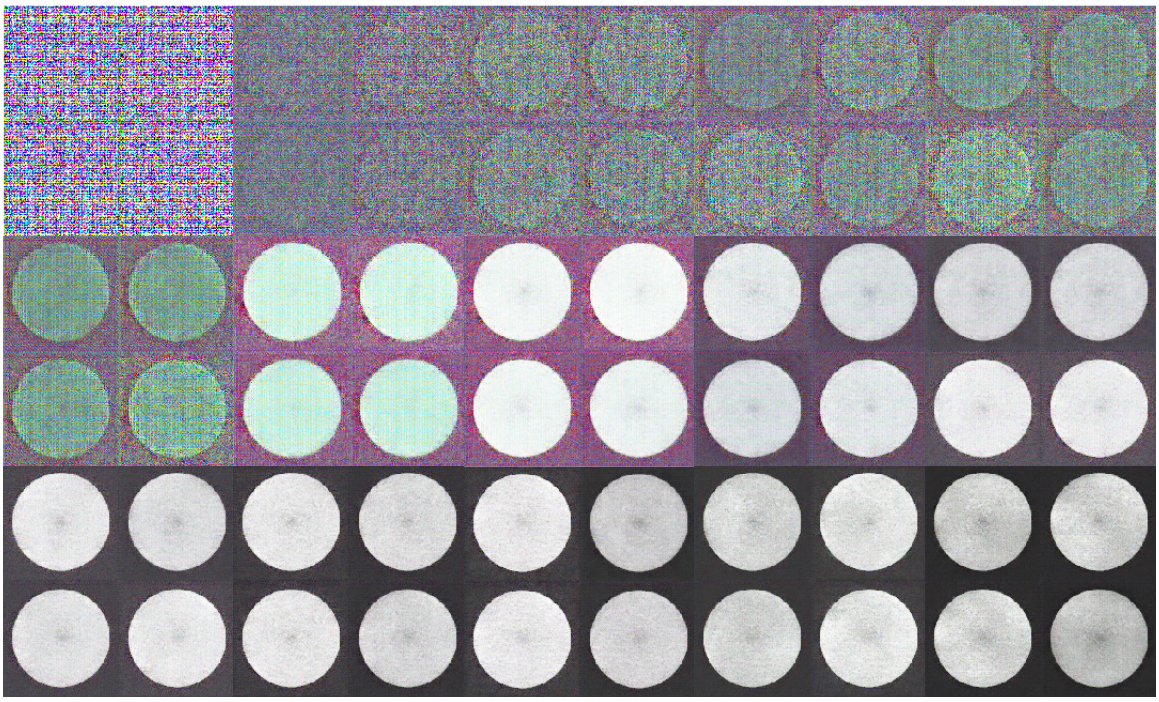
from a simple vector. Since this task is relatively simple, one of the concerns was that the model may not do well when the object has complex geometry or texture. Thus, I tested on another task.

4.4 Sphere with bump surface

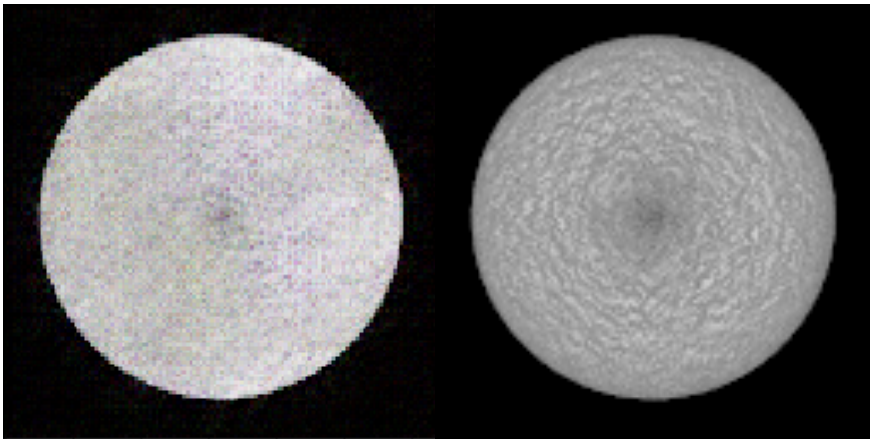


This task is similar to the box task, except that the sphere has a bump map.

The task was very similar to the box task, except the sphere had bumpiness texture attached. One of the concerns I had was that the model could not tell the difference between noise and bumpiness, and the generated image would end up being a white circle with random spots inside.



The training process of the sphere task.

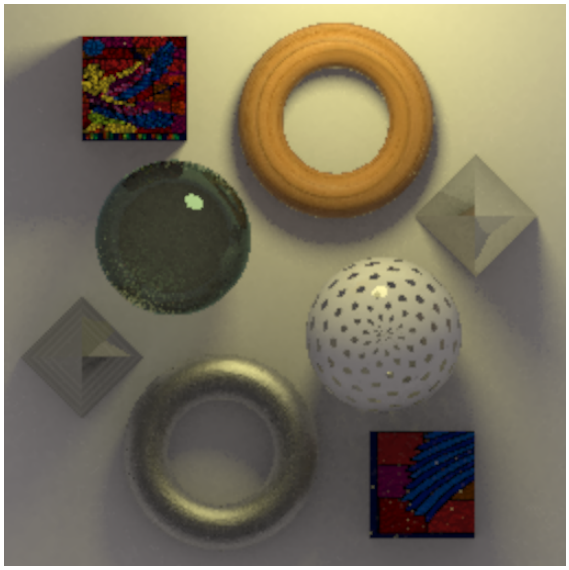


Left: Generated image. Right: Rendered image with same light position (ground truth)

We can observe that the generated image has almost no detail of the bumpiness texture. This was mainly due to the inherent drawback of the GAN architecture as we can observe the same issue in the box task.

GAN, in general, is hard to train. It's a semi-supervised learning algorithm and it depends on large amount of data to be trained well. The discriminator used in this architecture does not check the correctness of details, but rather tries to determine whether the image is true or not based on the overall information in the image. These caused the poor performance of the two tasks. To improve the performance, one could try to design a special loss function that takes into account the accuracy of reconstructed details. This is typically challenging and could easily take months of research. The other approach is to try other deep learning architecture.

4.5 New dataset

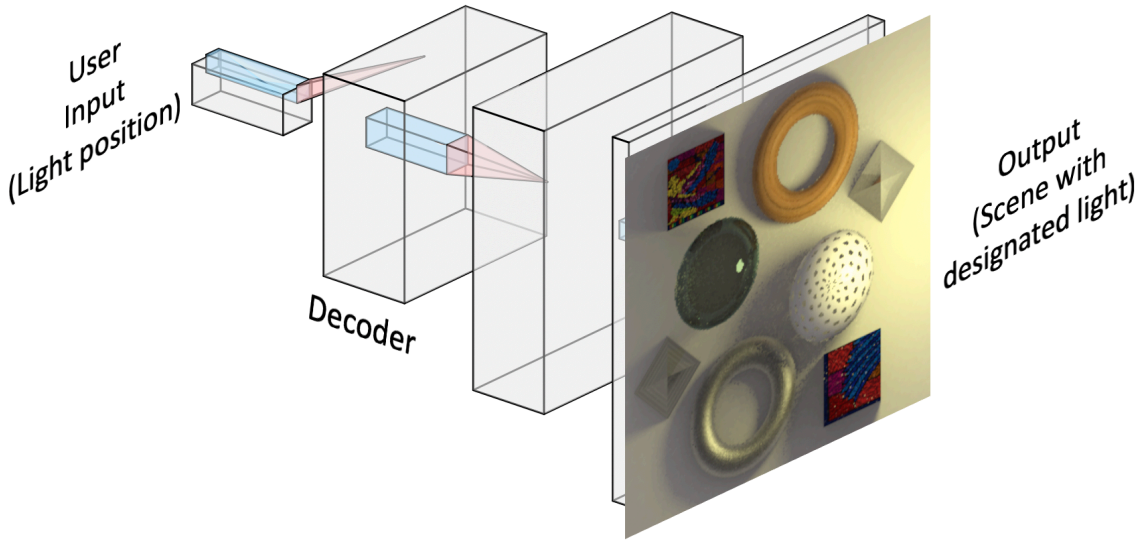


The new scene that contains different textures for testing purpose.

To evaluate the ability of image reconstruction, I created another dataset with a new scene. This scene contains multiple objects with different materials, each of which is intended to show a specific aspect of the reconstruction quality. For example, the transparent sphere is to show whether the underneath content can be seen, and whether the reflection spot is moving correspondingly with the light; the pattern on top of the cube is to show whether patterns will be distorted by the model; the two reflective pyramids are to show whether the reflected image is correct; the coarse donut shape is to show if the bumpiness is reconstructed properly.

4.6 Single decoder model

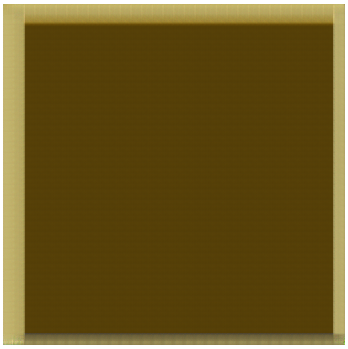
One of the concerns for the previous model was that the gradient may not be large enough for the generator to learn when it comes back from the discriminator. This is a very common situation for most of the deep learning models as the layers of the model increases. One intuitive way to check whether the relationship between the input and output could be learned is to try a relatively shallow network.



The single decoder architecture.

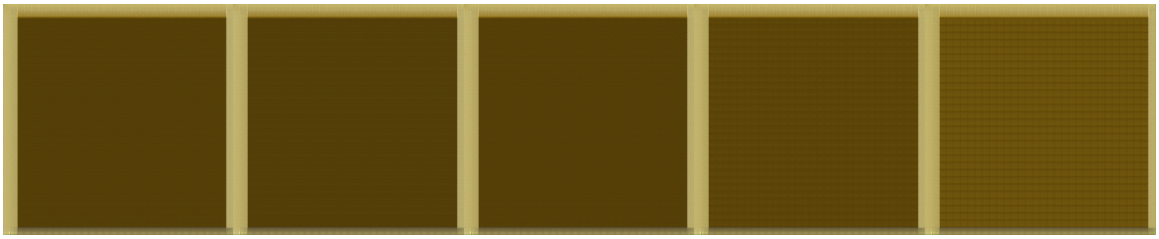
With this concern, I built a single decoder model. This architecture was extremely naive. It tries to build a connection between the light position and the corresponding image by force. All the information it received was the latent vector. The decoder's job was to not only reconstruct the image, but also move the light properly in the image, which could be challenging for the decoder even it has

very deep architecture. After the training, I was expected to see the light changes with different input vector, even though the quality of the image may be poor.



The training result of the single decoder model. The model was not able to reconstruct the image.

From the above image we can observe that the reconstruction of the image was failed. Most of the content in the image was wiped out. The decoder learned to 'average' all the pixel value which led to the yellow block in the image.



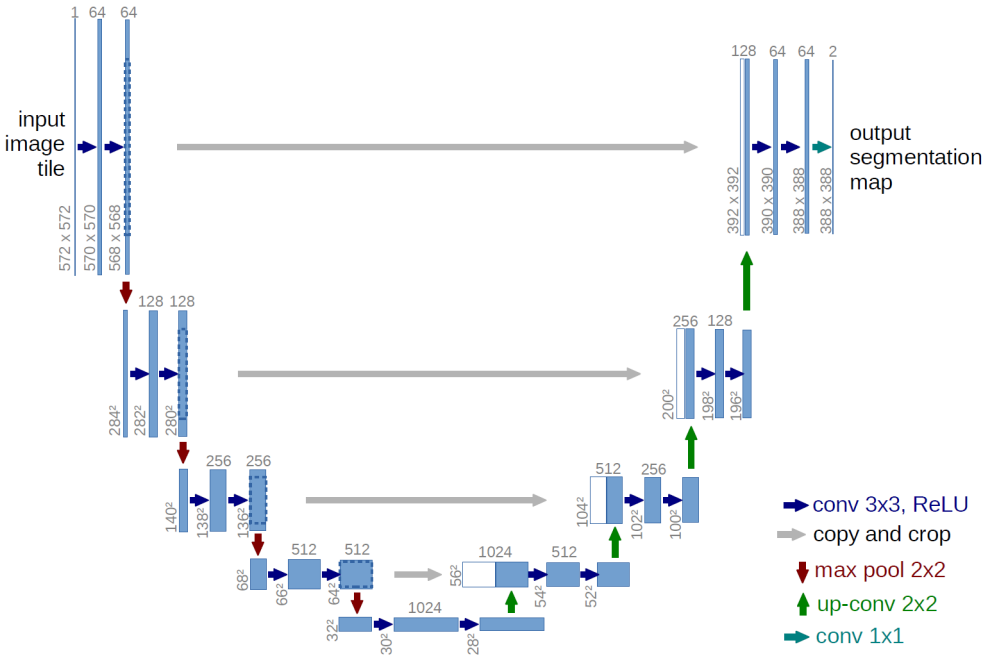
It can be easily observed that the brightness of the images is changing while the light is moving.

We did see that the relation between light position and the generated image was established. If we gave different light positions as input, the brightness of the image would change correspondingly.

This result may be improved by using a deeper network and feeding with more data. In general, the larger and deeper the network is, the larger the amount of data is required to train the model.

So far, all the previous three models take in light position vector as their only input, suggesting that this is all the information they can receive. The content of the image has to be 'memorized' through training, which means that if we would like to generalize our usage of the model to other images once it's been successfully trained, this will not work since the model is 'scene-specific'. We need a way to introduce the content of the image into the latent vector, so that the model would have less stress on reconstruction and more focus on the light modification.

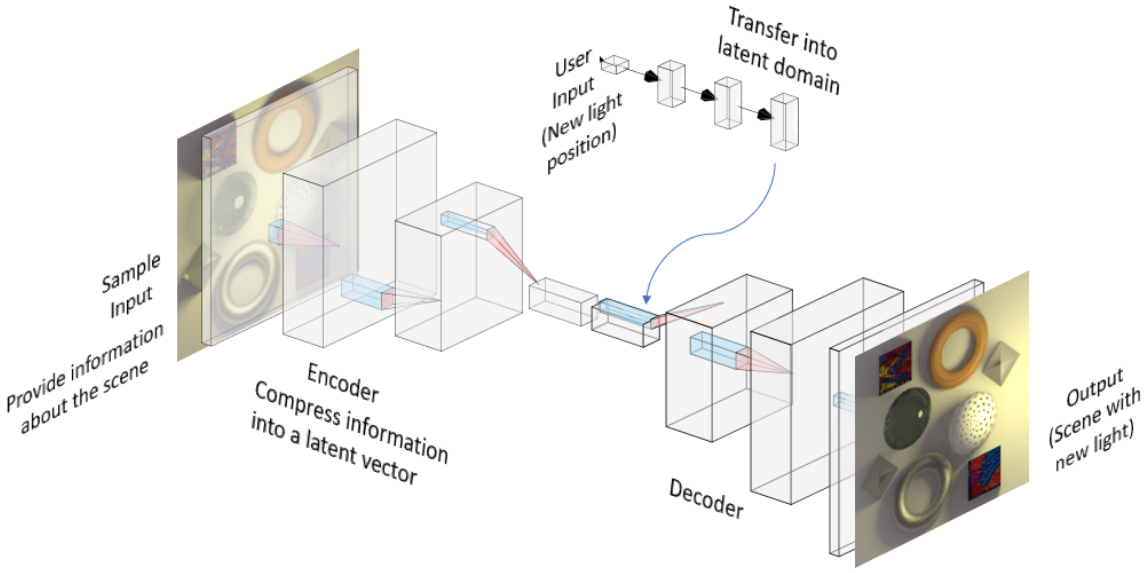
4.7 U-net(Ronneberger, Fischer, & Brox, 2015)



The original U-net architecture.

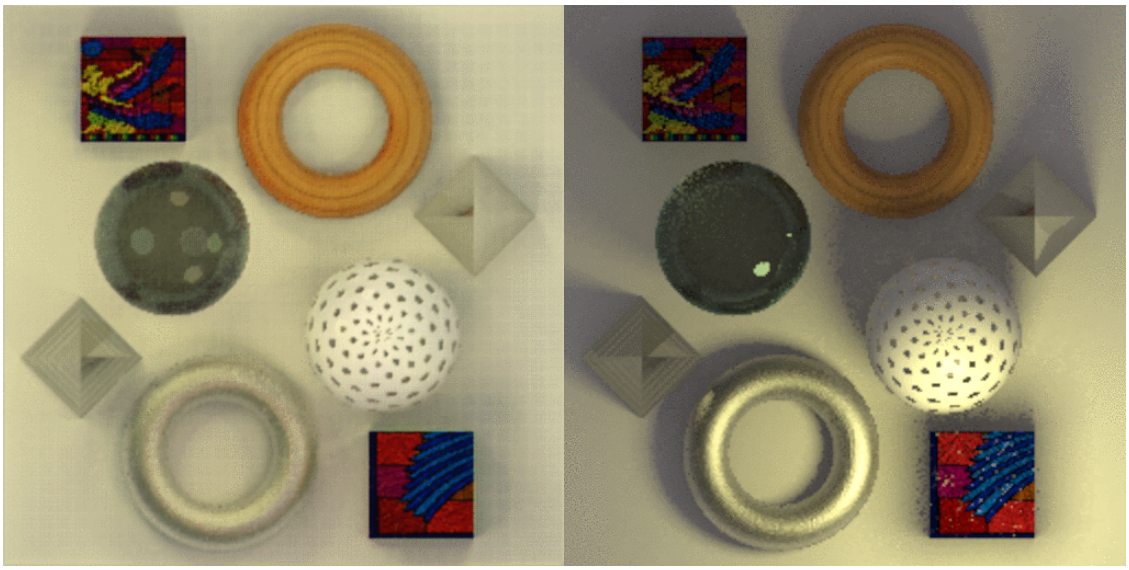
The U-net architecture is a convolutional neural network that was developed for biomedical image segmentation. The typical usage of U-net is to generate masks

of a target region given an input image, for example, tumor cells in an MRI scan. The architecture looks like a 'U' which justifies its name. This architecture consists of three sections: The contraction, The bottleneck, and the expansion section. The contraction section is very close to a traditional convolutional neural network which compresses the image into a vector, which is the latent vector that represents this image. For classification tasks, this vector is usually then be connected to other fully connected layers for binary output. In U-net, this vector is connected into a decoder (expansion section) so that it will be reconstructed into an image. This combination, an encoder directly connected to a decoder, is typically called an 'autoencoder'. What makes U-net different is that it introduces skip connections between encoder and decoder so that the information from the encoder can be passed to the decoder. This not only makes the decoding task easier, but also makes the gradient flows back easier, which will help in training very deep architecture.



The modified U-net architecture with conditional vector.

For our task, besides the image input, we also need to introduce the light position into the model. Similar to conditional DCGAN(Radford et al., 2015), this information is mapped into the latent space by some non-linear transformation (done by connecting it to several fully connected layers). Thus, the latent vector supposedly should have all the information it needs to do the relighting.



Left: generated image. Right: ground truth.

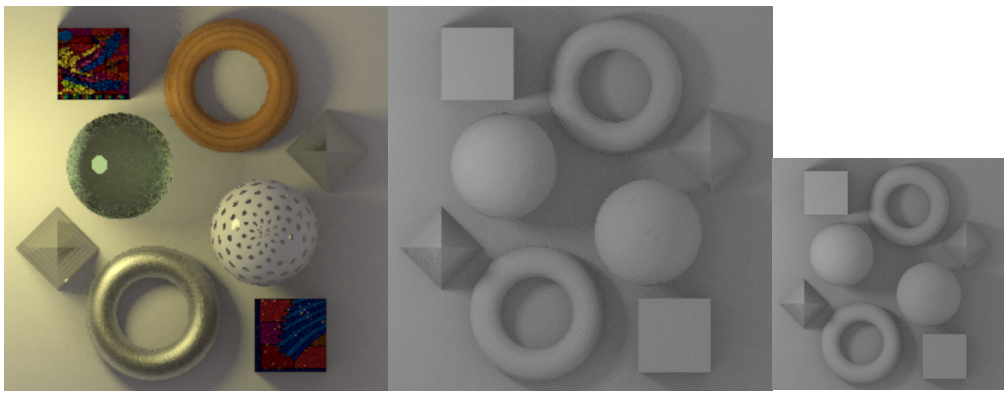
The above image shows the generated image after training. The content of the image was successfully reconstructed. We could observe that most of the texture was correctly generated: the pattern on top of the box was not distorted or blurred; the diamond-shaped opening on the white sphere preserved its shape; the reflection on the pyramid is clearly show. This indicated that the model was capable of tackling the reconstruction task.

There were some obvious errors in the result. The first thing to notice was that the image was much brighter than the ground truth, without a clear indication of where the light source was. Secondly, we could observe that there were multiple

reflection spots on the transparent sphere. This is a clear indication that the result is a simple combination of multiple images, which also explains why the image overall was brighter. The model did not learn to modify the image based on the moving light source. In fact, it ignored the target position input and the subtle changes between the sample input image and ground truth image. It generated an 'average' of all sample images. Since this result only caused a fairly small loss, the model stopped learning.

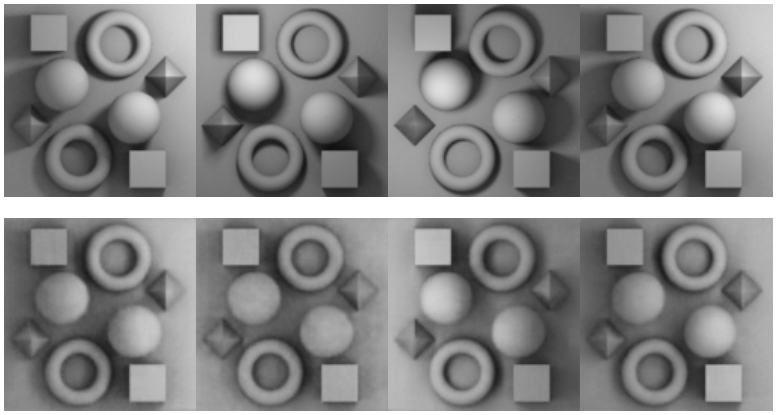
The progress stopped at this stage for a very long time, since I wasn't able to find the reason for the problem. I tried to fix the problem by tuning hyperparameters and use other loss functions and many other attempts. Eventually, I realized that the problem was not about the model but about the dataset.

There were mainly two problems that caused the failure of learning. First, the size of the image was too large. 512 by 512 is typically considered as high resolution in most of machine learning problems. As the size increases, the number of parameters needed grows exponentially and the number of training images grows exponentially as well. This is a common problem in machine learning known as 'the curse of dimensionality'. Second, the RGB values were too dominant compared to the subtle changes that the light makes. The model sees the input image as signals, and encode them using the encoder pipeline. The signals contained information about the color as well as the response to the light changes, all in the form of RGB values. Compared to colors, the light signals were way too weak to be captured by the filters and thus were lost in the process of encoding.



To solve the training problem, two modification is applied to the dataset. 1. Remove color information. 2 Reduce size.

To fix these problems, two modification was applied to the dataset. First, I removed the texture for all geometries in the scene and change the scene image into grayscale. Then I resized the image to 128 by 128. Having done that, the lighting changes became more obvious to the model.



First row: Ground truth image. Second row: Predictions.

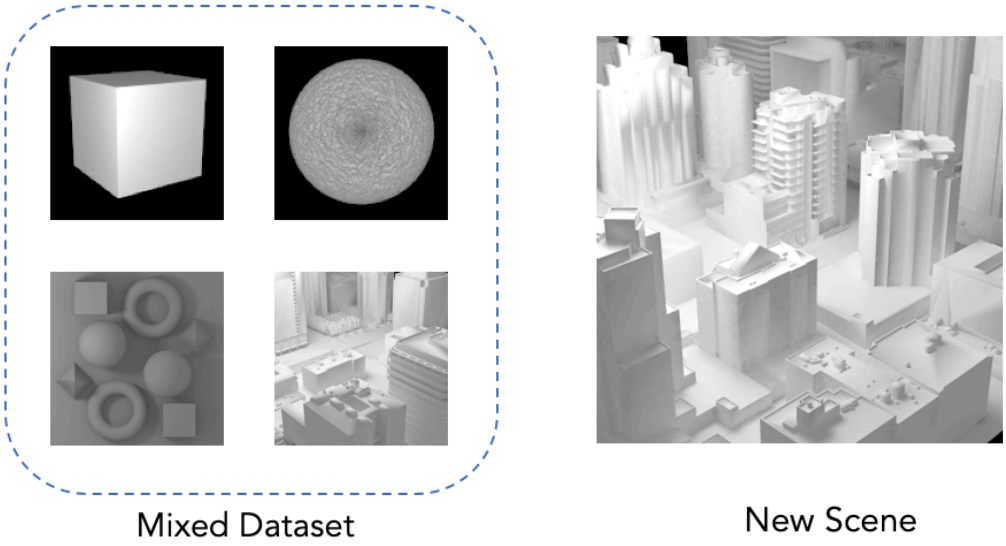
As we can see, the model trained on the modified dataset was capable of showing response to different lighting positions. Compared to the ground truth images, the predictions still suffered some accuracy issues, as the generated images were blurry. But the result was good enough as a proof of concept.

4.8 Generalization:

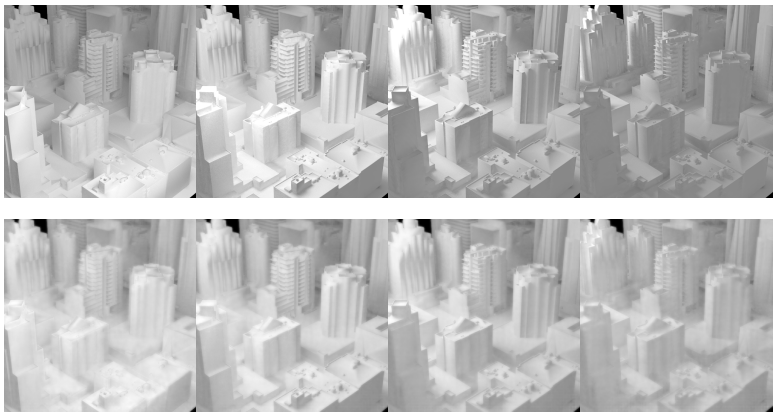
As mentioned previously, the encoder was used to receiving information from the scene. The model doesn't need to be trained specifically for each scene, rather it can receive this information through the encoder. Thus, the model has the potential to be later generalized and used on any image. The encoder can be replaced by other encoding architecture for other purposes and better performance as well, which I will not elaborate here.

The other benefit of the Unet structure is that it's fully convolutional. The entire model can be seen as a 'lens' and applied to a high-resolution image by a scanning motion. This gives the potential to be generalized to a large image without retraining the entire model. However, one should be aware of the size of the receptive field that the model has and compare it to the original image size to make sure it is not too small.

To show the potential of this architecture in generalization. I mixed the datasets that were previously used, along with a new scene. This new dataset, compared to the previous ones, were more diverse. Then the dataset was used to train on the model. Notice that in this training, the model did not focus on a fixed scene. Although the dataset had only four scenes in it, this should still be considered as a more generalized dataset. Moreover, after the training, the model was tested on another scene that was different and was not included in the training dataset.



Left: the mixed dataset. Right: the new test scene



First row: Ground truth image. Second row: Predictions.

The test results are shown above. We can observe that the prediction showed corresponding light movements. The brightness of different surfaces on each building showed some changes and they were in accordance with the same region in the ground truth image. The prediction lost a lot of details and contained some noise, yet this result was a clear indication that using diverse datasets with current architecture is generalizable.

5. Evaluation:

5.1 Image quality:

For numerical evaluation, I chose Peak Signal-to-Noise Ratio(PSNR) and Structural Similarity(SSIM). PSNR is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. For color images with RGB values per pixel, the definition of PSNR is the same except mean squared error is the sum over all squared value differences divided by image size and by three. PSNR is typically used to measure image reconstruction quality. It uses dB as units and ranges from 0 to infinity, the larger the better. The model trained on the modified dataset got a PSNR about 17.3. SSIM is used for measuring the similarity between two images. It is designed to improve on traditional methods such as PSNR or MSE. SSIM is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena. In short, SSIM mimics the way how humans compare two images, not by pixel, but by the structure of the image. It ranges from 0 to 1, the larger the better. For the last model, it has a SSIM of 0.7.

Both of the metrics indicate that the predicted image quality was only fair. These can also be observed visually. There are a lot of rooms for improvement, but it won't be the focus of this thesis. One thing to be aware is even if PSNR and SSIM scores are high does not necessarily indicate accurate prediction. For example, the model trained on 512 by 512 RGB images actually received a score of 23 for

PSNR and 0.985 for SSIM. Yet it still behaved poorly on the relighting as discussed before.

5.2 Limitation:

There are some limitations to the method shown in this thesis. The method assumes a clear definition of the light position in the input image. In this thesis, the definition is position. I used synthetic data which allows me to easily acquire the coordinates of the light position in each image. However, this is usually not true for most of the cases, where we typically have only an image. This limits us to use many large, diverse and organized open source datasets.

Moreover, the current model only works with grayscale images. Proper modification of the model in order to work with RGB images is definitely one of the important goals to achieve in the future.

Lastly, if one would want to further generalize this architecture to be able to control multi-constraints, for instance changing the position and the brightness of the light at the same time, this can become challenging. This requires the dimension of the latent vector to be increased which could easily lead to a very large-scale model that is incredibly hard to train.

6. Conclusion

6.1 Contribution:

In this thesis, I have demonstrated my exploration of the proof of concept of utilizing deep learning to create content-aware smart filters. I showed that this approach has the potential to combine both advantages from simulation and image processing to create lightweight filters that have the ability to do modification based on the content of the image.

First, I reviewed the mainstream methods to modify light in the visualization process. Two general categories were compared: simulation and image processing. I summarized their limitations as follows: simulation is computationally heavy and image processing is 'content-blind'.

Second, I designed a simple relighting task and demonstrated how simulation and image processing methods tackle it. This step showed the advantage and disadvantage of both methods, and further, revealed the potential problems both methods could have when the complexity of the task increases.

From there I made my claim of using deep learning to create content-aware smart filters that could potentially combine the benefits of the two methods. I started with the explanation of how deep learning works, both conceptually and mathematically. Then I demonstrated a detailed process of how to create the smart filter to tackle the relight task. The process contained data preparation, machine learning model design, model debugging and dataset modification.

Finally, I showed the final model that could work as a smart filter and apply image-based content-aware modifications, specifically for the relighting task. I briefly discussed about the evaluation of the model and limitation of the overall approach.

6.2 Future work:

6.2.1 Modulate data pipeline:

One of the key steps to create a smart filter is to prepare a proper dataset for training. The dataset defines the function of the smart filter. One could collect his/her own data and use them to train for a specific purpose. A modulated data pipeline would be helpful to automate, regulate and organize the process of preparation. Protocols of constraint definition, image size, format should be established.

6.2.2 Improve accuracy:

As one can see the prototype in this thesis still needs a lot of improvement in image quality. This might need to research on better loss function, additional preprocess module, multiple encoders or decoder or other means.

6.2.3 Extend the approach to other application:

This exploration process can be applied to create other applications as well. For example, the change of shape, color, or even content in the image with respect to time, position, force and so on. One could use similar machine learning

architecture and follow the step to prepare data, training and debugging to develop his/her own smart filters.

Reference:

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2016). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *ArXiv:1606.00915 [Cs]*. Retrieved from http://arxiv.org/abs/1606.00915

Cobusier, L. (1986). *Towards A New Architecture*. New York: Dover Publications.

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *ArXiv:1603.07285 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1603.07285

Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2414–2423. https://doi.org/10.1109/CVPR.2016.265

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Networks. *ArXiv:1406.2661 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1406.2661

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *ArXiv:1611.07004 [Cs]*. Retrieved from http://arxiv.org/abs/1611.07004

Junginger, A., Hanselmann, M., Strauss, T., Boblest, S., Buchner, J., & Ulmer, H. (2018). Unpaired High-Resolution and Scalable Style Transfer Using Generative Adversarial Networks. *ArXiv:1810.05724 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1810.05724

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive Growing of GANs for Improved Quality, Stability, and Variation. *ArXiv:1710.10196 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1710.10196

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Oles, P. S. (1979). *Architectural illustration: the value delineation process*. Retrieved from https://books.google.com/books?id=lfxPAAAAMAAJ

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ArXiv:1511.06434 [Cs]*. Retrieved from http://arxiv.org/abs/1511.06434

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv:1505.04597 [Cs]*. Retrieved from http://arxiv.org/abs/1505.04597

Wit House Productions. (n.d.). *Our Friend, Steven Oles*. Retrieved from https://vimeo.com/108906162

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *ArXiv:1703.10593 [Cs]*. Retrieved from http://arxiv.org/abs/1703.10593