LEARNING LIGHTING MODELS WITH SHADER-BASED NEURAL NETWORKS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Qin He

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

ACKNOWLEDGMENTS

I wish to gratefully acknowledge my thesis committee members - Dr. Tim McGraw, Dr. Esteban Garcia and Dr. Christos Mousas, for providing valuable suggestions and comments for this thesis. Special thanks to my advisor, Dr. McGraw for his guidance and comments on the research and thesis writing. I also wish to thank my friends and all staff in CGT department.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

BRDF	Bidirectional Reflectance Distribution Function
NDF	Normal Distribution Function
ANN	Artificial Neural Networks
FCNN	Fully Connected Neural Networks
MSE	Mean Squared Errors
DoF	Degrees of Freedom

ABSTRACT

He, Qin M.S., Purdue University, May 2020. Learning Lighting Models with Shader-Based Neural Networks . Major Professor: Tim McGraw.

To correctly reproduce the appearance of different objects in computer graphics applications, numerous lighting models have been proposed over the past several decades. These models are among the most important components in the modern graphics pipeline since they decide the final pixel color shown in the generated images. More physically valid parameters and functions have been introduced into recent models. These parameters expanded the range of materials that can be represented and made virtual scenes more realistic, but they also made the lighting models more complex and dependent on measured data.

Artificial neural networks, or neural networks are famous for their ability to deal with complex data and to approximate arbitrary functions. They have been adopted by many data-driven approaches for computer graphics and proven to be effective. Furthermore, neural networks have also been used by the artists for creative works and proven to have the ability of supporting creation of visual effects, animation and computational arts. Therefore, it is reasonable to consider artificial neural networks as potential tools for representing lighting models. Since shaders are used for general-purpose computing, neural networks can be further combined with modern graphics pipeline using shader implementation.

In this research, the possibilities of shader-based neural networks to be used as an alternative to traditional lighting models are explored. Fully connected neural networks are implemented in fragment shader to reproduce lighting results in the graphics pipeline, and trained in compute shaders. Implemented networks are proved to be able to approximate mathematical lighting models. In this thesis, experiments are described to prove the ability of shader-based neural networks, to explore the proper network architecture and settings for different lighting models. Further explorations of possibilities of manually editing parameters are also described. Mean-square errors and runtime are taken as measurements of success to evaluate the experiments. Rendered images are also reported for visual comparison and evaluation.

CHAPTER 1. INTRODUCTION

To produce a realistic appearance for computer-generated objects, numerous illumination models have been proposed for shading. Reflectance models which describe the color and the distribution of the reflected light (Cook & Torrance, 1982), play an important role in the display of most surfaces. Most surfaces are neither ideal specular reflectors nor ideal diffuse reflectors but perform a combination of both specular and diffuse reflection. In computer graphics models, diffuse reflections distribute equally in all directions for all materials (Blinn, 1977; Cook & Torrance, 1982). Therefore, specular reflections play a major role in indicating the materials.

Phong (1975) firstly models the specular reflections in computer graphics based on the earlier fundamental techniques. The proposed model divides reflected light into three components, specular reflection, diffuse reflection, and environmental diffuse reflection (ambient light) to perform realistic shading which cannot be accomplished using basic Lambert's cosine law (Phong, 1975). Then Cook and Torrance (1982) proposed a model that introduces more physical theories and takes consideration of spectral energy distribution to expand the reflectance models to the reproduction of specific real materials. The Cook-Torrance model and models after it also adopt the concept of normal distribution function (NDF), which describes a surface consisting of microfacets. Only microfacets with a specific normal contribute to the specular reflection. The distribution of microfacet orientations then becomes a basic component in producing specular reflections in physically based shading (Burley, 2012). Different material parameters and microfacet distributions are explored to achieve better presentation of materials (Walter, Marschner, Li, & Torrance, 2007; Ward, 1992). With better hardware and data used, the presentation of materials becomes more realistic, but also more dependent on computation and measured data like material parameters and textures. When more physically valid formulas get involved in the calculation of reflection, however, the explicit physical meaning of each component in modern lighting models makes it difficult to add artistic creation into the reproduction of materials. Therefore a proper approximation with tunable parameters may be useful to offer ideas for creation in lighting from a new perspective.

Neural networks have been proved to have the ability to approximate arbitrary functions (Cybenko, 1989), and have been used for similar tasks like image-based relighting (Ren, Dong, Lin, Tong, & Guo, 2015; Xu, Sunkavalli, Hadap, & Ramamoorthi, 2018). Some unique effects created by neural networks have also been used for artistic exploration (Broad & Grierson, 2017). But the application of neural networks on lighting models themselves hasn't been explored yet. Manipulating network parameters after training offers possibilities to achieve novel effects. Currently popular neural network platforms and libraries are difficult to be combined with modern graphics pipeline, while graphics libraries offered general-purpose computing solutions(General-Purpose Computing on Graphics Processing Units, GPGPU). This research explore the possibility of shader-based neural networks approximating lighting models, to provide a new perspective of presentation and creation of lighting effects. Compute shaders are adopted to implement and to train the networks. To better fit neural networks into modern graphics pipeline, neural networks are also implemented in fragment shader to produce lighting results. Recommended model hyperparameters and settings for fitting specific lighting models and effects of manipulating network parameters are explored and reported,

1.1 Scope

This research will mainly explore the performance and possibility of shader-based neural networks, by the means of using neural networks to approximate traditional lighting models, and the proper network structure and settings for the task. Experiments will be done to examine the performance of shader-based neural networks on lighting models, with different type of activation functions and different network parameters. This research will also try to explore the effects of network parameter matrices in neural networks on achieving novel effects. Aesthetics of the rendered results is difficult to quantify and compare, thus will not be taken in consideration in this research. Other aspects of lighting models like scattering and global illumination, specific formulas of deterministic lighting models and complex network structure that are impractical to implement in shaders will be out of the scope of this research. There is only direct lighting and relatively low dimensional inputs and outputs in most mathematical lighting models. Thus screen-space shading is used in this research, where the input and output of each pixel will be seen as one sample for the neural network. Fully connected neural networks (FCNN) are used in the implementation and no surrounding pixel is considered.

1.2 Significance

Lighting is a critical aspect of presenting materials since it is directly related to the pixel intensity. Earlier lighting models contain empirical components that allow artists and technicians to adjust appearance, but these models are limited in the range of effects that can be presented. To get further in the realistic reproduction of reflections, modern lighting models contain more physically-based parameters that are not directly related to artistic control but only physical properties of materials and light. A new perspective of lighting models can be useful widening the artistic exploration while keeping the plausible appearance.

Neural networks are good at dealing with complex tasks and have been proved to have the ability of approximate arbitrary functions (Cybenko, 1989; Hornik, 1991). Neural networks have been applied in many areas in computer graphics to assist current tasks and are proven to be successful. For instance, Zhang, Starke, Komura, and Saito (2018) design and implement neural networks to control animation, and Tompson, Schlachter, Sprechmann, and Perlin (2017) use neural networks to accelerate fluid simulation (Lucy, 1977). Neural networks have also been used for lighting-related tasks such as global illumination (Ren et al., 2013), relighting (Ren et al., 2015; Xu et al., 2018) and screen-space shading effects (Nalbach, Arabadzhiyska, Mehta, Seidel, & Ritschel, 2017). But most previous similar works focus on the complexity of lighting environment and acceleration and take neural networks as experimental tools independent of graphics pipeline. The possibility of neural networks as a part of modern graphics pipeline, and an alternative of creative lighting models has not been explored yet. Taking network weights and biases as general model parameters, neural networks can provide much more degrees of freedom (DoF) than general mathematical models. It is worthwhile further exploring the appliance and performance of shader-based neural networks to approximate lighting models, and manipulating network parameters to achieve novel effects.

1.3 Research Questions

- What are the best settings of FCNN to learn traditional lighting models?
- What is the performance of a shader-based implementation?
- For different lighting models, how closely do certain neural network architectures approximate them?
- What effects can FCNN achieve manipulating network parameters?

1.4 Assumptions

The assumptions for this research include:

- the visual appearance of most materials can be presented by reflections;
- reflections can be seen as a combination of specular reflection, diffuse reflection, and environmental diffuse reflection;
- the calculation of reflection can be modeled as a continuous function using finite inputs;
- the feasibility and performance of FCNN approximation for lighting models can be evaluated via visual judgment and mean square error measurement.

1.5 Limitations

The limitations for this research include:

- the materials to be explored will be limited depending on mathematical lighting models that can be accessed,
- minor differences between the approximation and the ground truth functions or normal maps cannot be avoided,
- unexpected artifacts can be found in the approximated results.

1.6 Delimitations

The delimitations for this research include:

- the lighting models to be tested are limited to several typical ones,
- the materials to be tested are limited to the ones which can be well represented by these models, such as plastics and metals,
- refraction won't be discussed or investigated,
- anisotropic models won't be discussed or investigated,

- detailed production of components in empirical models like ambient color won't be discussed or investigated,
- capture, measurement and/or production of new data won't be included.

1.7 Definitions

The following words and phrases will used throughout the thesis:

Lighting model: refers to reflectance model, which describes the color and the distribution of the light reflected from a surface (Cook & Torrance, 1982).

Bidirectional reflectance distribution functions: bidirectional reflectance distribution functions (BRDFs) describe the (geometrical) reflecting properties of most surfaces. It is a concentration of reflectance given a pair of vectors - the light and view directions. (Nicodemus, Richmond, Hsia, & Limperis, 1977)

Normal distribution functions: normal distribution function (NDF), or facet slope distribution function represents the fraction of the facets that contribute to the specular reflections. (Cook & Torrance, 1982)

Neural networks: neural networks, or artificial neural networks (ANN) are mathematical models inspired by biological neural networks that can be used to approximate functions and models (Funahashi, 1989)

Shader: a specific type of program running on graphics processing unit, originally used for the rendering of 3D scenes, as a substitute of fixed graphics pipeline. But now it can also perform a variety of specialized functions in various fields.

1.8 Summary

This chapter provided an overall statement of the research, the scope, the significance, research questions, limitations, delimitations, and definitions to be used for the research project.

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

2.1 Literature Review

To correctly reproduce the appearance of different objects in graphics applications, numerous lighting models have been proposed over the past several decades to calculate reflection and refraction. These models are among the most important components in modern graphics pipeline since they decide the final pixel intensity shown in the images. They are also among the main causes of heavy computation in computer graphics because of the huge amounts of vertices and fragments to be shown. The basic models were first proposed and rapidly developed in 1970s and 1980s (Cook & Torrance, 1982; Phong, 1975). Different variations, from simple empirical models to models that are more theoretical, are proposed to balance the mathematical simplicity and physically validity since then.

Artificial neural networks, or neural networks are famous for their ability to deal with complex data and to approximate arbitrary functions (Cybenko, 1989; Hornik, 1991). Various kinds of neural networks have been used for tasks in computer graphics and proven to be effective (Holden, Komura, & Saito, 2017; Tompson et al., 2017). They have also been used for lighting related tasks, such as image-based relighting (Xu et al., 2018) or global illumination (Ren et al., 2013), and show the possibility of being used for lighting models.

This literature review provides an overview of different lighting models and the application of neural networks in computer graphics.

2.2 Earlier Empirical Lighting Models

Given information about visible surfaces, light sources and view directions, lighting models calculate the light received by the viewer, or pixel intensity on the screen. Reflectance models which describe the color and the distribution of the reflected light (Cook & Torrance, 1982), play the major role in the display of most surfaces and are most popularly discussed by earlier lighting models. The very first model used was the "cosine law" (Phong, 1975, p.312):

$$S_p = C_p cos(i)$$

where C_p is the reflection coefficient, i is the incident angle and S_p is the reflected light from the surface. "Cosine law" describes the shading of each point on a polygon as the product of a shading coefficient for the polygon and the cosine of the angle between the polygon normal and the direction of incident light (Phong, 1975, p.312). It assumes that all surfaces are diffuse reflectors that reflect light equally in all directions, and only use a constant C_p to present the reflection of different materials. But most surfaces are neither ideal specular reflectors nor ideal diffuse reflectors but perform a combination of both specular and diffuse reflection. In lighting models for computer graphics, diffuse reflections are generally similar for all materials and specular reflections are more important to indicate the materials (Blinn, 1977; Cook & Torrance, 1982). Therefore, the "cosine law", which doesn't allow for any of the specular properties of the material, fell out of favor.

Phong (1975) first modeled the specular reflections in computer graphics, taking consideration of the position of viewer. The author divided the light received by the eye into three components, specular reflection, diffuse reflection, and

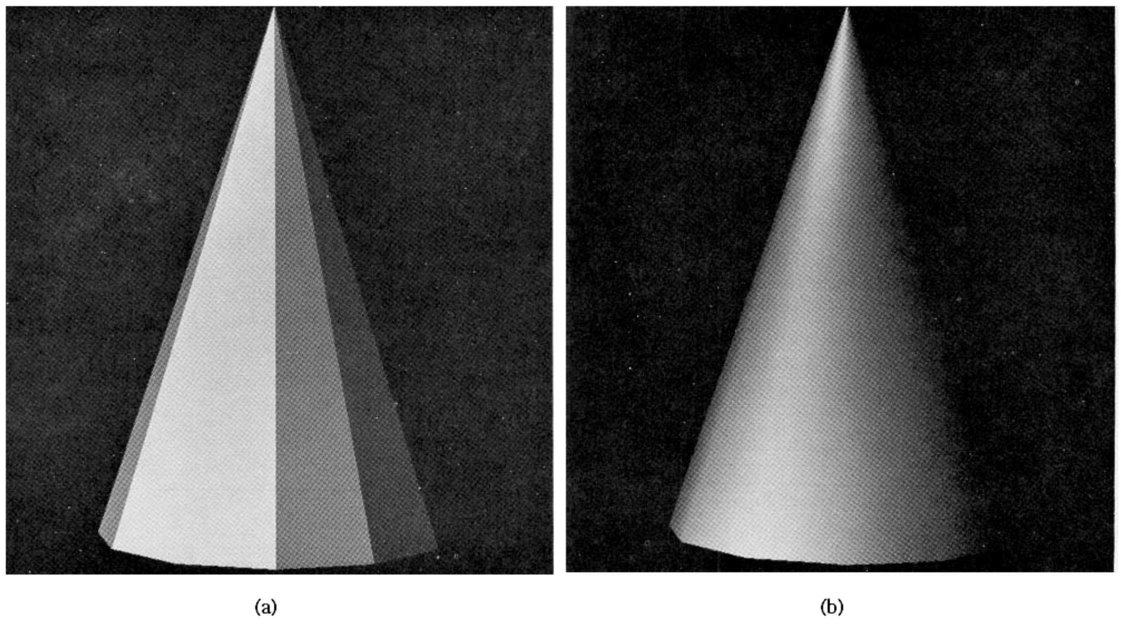

Figure 2.1. Shading results using "cosine law" (a) and the Phong lighting model (b). (Phong, 1975)

environmental diffuse reflection (ambient light) to perform realistic shading. The Phong model models the reflection as (Phong, 1975, p. 315):

$$S_p = C_p[cos(i)(1-d)+d] + W(i)[cos(s)]^n$$

where d is the environmental diffuse reflection coefficient that describes how much ambient light is considered, $W(i)$ is a function of incident angle which gives the ratio of specular reflection and incident light, s is the angle between the direction of reflected light and view direction. In practice, C_p, as well as $W(i)$, d and n, are set by artists when creating the scene. In this model, the ambient component C_pd represents reflected light that comes from incident light in the environment. It is then reflected equally in all directions just like the diffuse reflection $C_pcos(i)(1-d)$. The specular component represents highlights that are concentrated around the mirror direction of incident light (given by the ideal reflection law), the specular reflection characteristics of a material is expressed by the function $W(i)$ and the exponent n.

The Phong model was modified by Blinn (1977) replacing the angle between the reflected light direction and the view direction with the angle between surface normal and the direction of maximum highlights (the half vector) which will be described in next section. These earlier methods were all based on basic geometrical optics, and are often stated as empirical models (Yan et al., 2014) dependent on parameter settings. The results generally look plastic because these models can only generate specular reflections with the color of light sources (Cook & Torrance, 1982). These models represent the very first steps to modeling realistic reflections and are still used by some applications because of their mathematical simplicity and efficiency (Ward, 1992).

2.3 Bidirectional Reflectance Distribution Functions (BRDF) and Microfacet Models

Earlier empirical models can generate realistic reflections for some materials like plastic or porcelain, but not for materials like metals. These earlier models didn't take into consideration the properties of materials themselves and ignored the structures that are smaller than a single fragment. Instead, fixed constants were used to represent complex phenomena like attenuation and roughness, assumptions that do not hold for real-world materials. Therefore more theoretical lighting models were proposed to extend the range of materials can be represented.

Bidirectional reflectance distribution, which gives the basic quantity that characterizes the reflecting properties of surfaces (Nicodemus et al., 1977), was introduced to model reflections in computer graphics by Torrance and Sparrow (1967), to solve the problem of specular bump (Blinn, 1977) and the deviation between predicted and actual peak specular reflection. Torrance and Sparrow (1967) also introduced the important concept of the microfacet, which models surfaces as a collection of small, randomly oriented mirror-like faces. Blinn (1977) further developed the method and proposed bidirectional reflectance distribution function (BRDF) for computer graphics implementation with clear definitions of the direction of maximum highlights (the half vector) and three commonly used components of BRDF models. According to basic geometrical optics, the incident ray to a point is reflected at the same angle to the surface normal as the incident ray. Therefore, specular reflections are intensest, that is, the surface is in maximum highlight when the view direction coincides with or distributed around this reflection direction, as shown in Fig.2.2 . Blinn (1977) took the bisector of the angle between incident ray

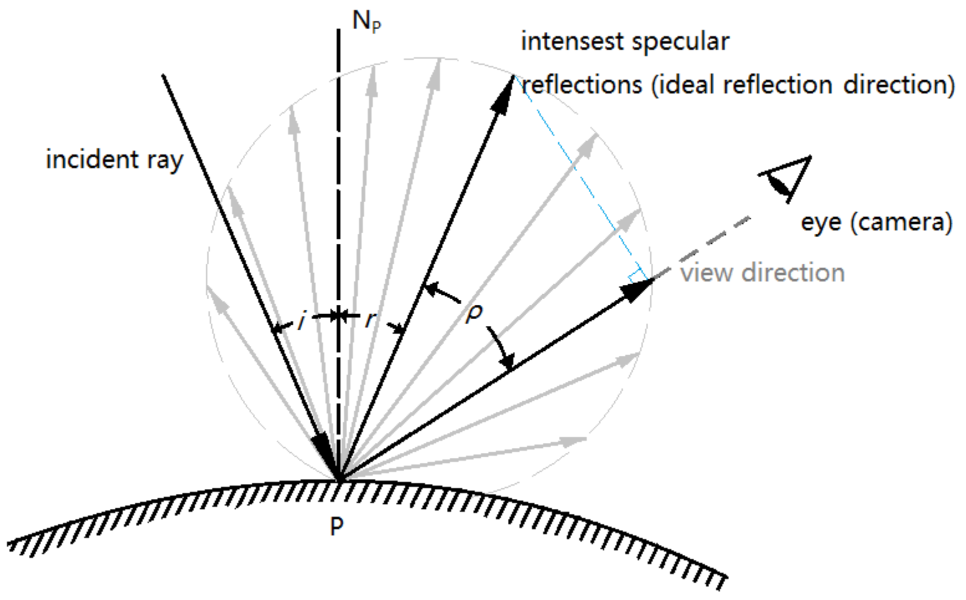

Figure 2.2. A simple example of specular reflection lobe in computer graphics lighting (Phong model with exponent $n = 1$). Specular reflections are intense when the view direction is around the ideal reflection direction.

and view direction as “the maximum specular direction”(p. 193), for the strongest specular reflections stated above can be observed when the surface normal coincides with this angular bisector. The direction is stated as (Blinn, 1977, p. 192):

$$H = \frac{L + E}{||L + E||}$$

where L is the light source direction, and E, can also be expressed using V, is the eye direction or view direction. H is also called "the half vector", for it is "the half way" from the incident ray to the eye, and widely adopted as one of the basic geometrical parameters. Blinn (1977) used the concept of H to modify the original Phong model as stated in previous section, and also factorized the specular reflectance into three components (Blinn, 1977, p. 193):

$$s = \frac{DGF}{(N \cdot E)}$$

where D is the distribution function of microfacets, G is the amount by which the facets mask each other, or shadowing-masking function, and F is the component that represents the Fresnel reflection law (Born & Wolf, 2013). With the clear definition of H, D can be also stated as the proportion of microfacets where the normals of the facets oriented H, since only these facets contribute to the final specular reflection. All the three components can be approximated with some empirical functions of geometrical parameters in the reflection, to balance the efficiency and accuracy. Torrance and Sparrow (1967) adopted a simple Gaussian for D, Blinn (1977) explored two more distributions and compared the accuracy and efficiency of all the three distributions.

Later, Cook and Torrance (1982) proposed a model that incorporates more physical properties. The Cook-Torrance model is also based on the concept of BRDF, and gives the widely accepted formula of specular bidirectional reflectance (Cook & Torrance, 1982, p. 11):

$$R_s = \frac{F}{\pi} \frac{DG}{(N \cdot L)(N \cdot V)}$$

where D, G, F are microfacet distribution function, corresponding shadowing-masking function and the Fresnel term, N is normal of the macro surface, L and V are the incident direction and view direction, separately. The authors re-derived their computer graphics approximate lighting model from the models that are more complex and physically valid (Nicodemus et al., 1977; Torrance & Sparrow, 1967), taking further consideration of spectral energy distribution, to expand the model to the representation of specific real materials, especially metals. They introduced the spatial distribution of electromagnetic radiation in different materials and the law of conservation of energy into the model through the derivation, and treat BRDF as a function of energy distribution. In this way, the authors emphasized specular and diffuse reflection in a different manner from the previous models, where specular reflections are taken as the components that reflected from the surface and diffuse reflections are mainly resulting from internal scattering, while previous methods modeled diffuse reflections as a result of rough surfaces and specular reflections as simple changes of direction of the source light. This assumption introduces the conception of "the reflectance spectrum of the material" (p. 23) into the Cook-Torrance model. Therefore specular reflections in the Cook-Torrance model perform not only changes of direction but also color shifts on the incident light, which allows the model to reproduce specular reflections from both homogeneous materials like metals and non-homogeneous materials like plastics, and to fix the problem of earlier empirical models of generating only plastic appearances for virtual objects.

Moreover, the research of Cook and Torrance (1982) also extended the exploration of distribution function of microfacets, i.e. normal distribution function (NDF) or slope distribution functions. The authors further compared multiple NDFs proposed for computer graphics after the work of Torrance and Sparrow (1967) as well as multiple distributions measured in the real world that describe the scattering of radar or infrared radiation. With the BRDF model being widely adopted by researchers and artists (Ren et al., 2013; Walter et al., 2007), the

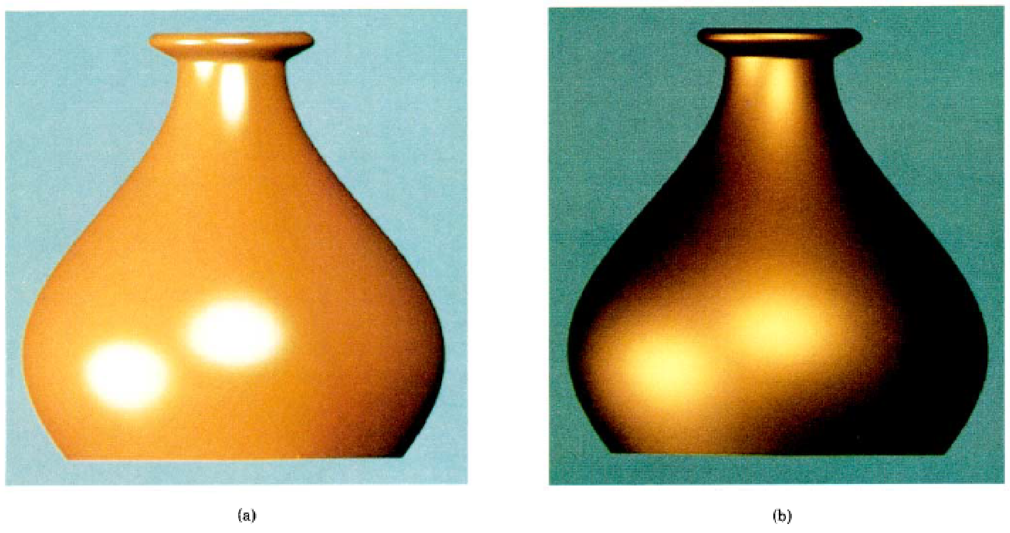

Figure 2.3. The non-homogeneous material (a) and homogeneous material (b) presented by Cook-Torrance model. (Cook & Torrance, 1982)

exploration of better and faster approximations for normal distribution functions and corresponding shadowing-masking functions has become a popular topic in the research of lighting models. The consideration of the reflectance spectrum lead to the usage of measured thermophysical properties of materials (Matusik, Pfister, Brand, & McMillan, 2003; *Thermophysical properties of matter database TPMD.*, 2003). The Cook-Torrance model became the basis of the upcoming modern methods of physically based rendering (Burley, 2012).

Ward (1992) made use of physically measured distribution data to help modify the Cook-Torrance model for faster execution and more realistic appearance. Instead of introducing more electromagnetic theories, the literature pointed out the differences between goals of theoretical models and computer graphics models and simplified the latter. In the Ward BRDF, the expression of three components in specular reflection are replaced with some empirical factors and functions that can be sampled and calculated more easily (Ward, 1992, p. 268):

$$s = \rho_s \cdot \frac{1}{\sqrt{cos\theta_i cos\theta_r}} \cdot \frac{\exp(-tan^2\delta/\alpha^2)}{4\pi\alpha^2}$$

where θ_i and θ_r are incident and view direction, separately, δ is the angle between surface normal and the half vector, calculated using incident and reflected angles. Specular reflectance ρ_s and standard deviation of the surface slope α are parameters that control the effects of specular reflection, decided by measured data. To simply visualize the effect of different reflectance model, two-dimension (as shown in Fig. 2.2) or three-dimension lobe shapes are used to describe the distribution of reflected light. In this context, ρ_s controls the magnitude of, and α controls the width of the specular lobe (Walter, 2005) As shown in the equation, the Cook-Torrance model was simplified with more empirical components that can be adjust more intuitively, while being more physically valid using measured data. Realism of the appearance was assessed by comparing to measured reflectance data. Ward (1992) then extended the model to present surfaces with anisotropic slope (normal or height) distributions by simply adding another principal direction into the specular reflection (Ward, 1992, p. 268):

$$s = \rho_s \cdot \frac{1}{\sqrt{cos\theta_i cos\theta_r}} \cdot \frac{\exp[-tan^2\delta(cos^2\phi/\alpha_x^2 + sin^2\phi/\alpha_y^2)]}{4\pi\alpha_x^2\alpha_y^2}$$

where α_x and α_y control the width of the lobe in the two principal directions of anisotropy, ϕ is the azimuth angle between the half vector and its projection into the surface plane. A computationally cheaper approximation is also given in the literature. To implement the Ward model in computer graphics applications, an appropriate sampling method is needed to find geometrical parameters δ and ϕ. Since the model is an empirical approximation of real world reflectance, Ward (1992) adopted hybrid deterministic and stochastic ray tracing technique to avoid bias caused by deterministic technique and high variance caused by Monte Carlo approach. The sampling method was then modified by Walter (2005).

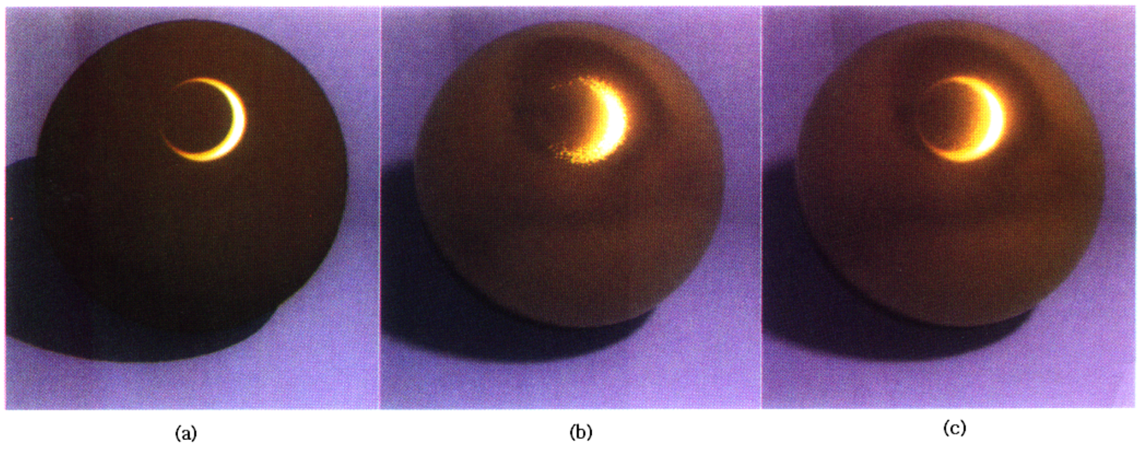

Figure 2.4. Rendering results using the Ward anisotropic reflectance model (Ward, 1992). (a) shows the result of deterministic technique, (b) shows the result using strict Monte Carlo sampling approach, (c) shows the result using the hybrid method proposed by Ward (1992).

On the basis of BRDF models, Walter et al. (2007) took a further step and used the microfacet model to describe refraction. They proposed a more comprehensive bidirectional scattering distribution function (BSDF) which includes bidirectional transmittance distribution function (BTDF) to simulate both reflection on and transmission through rough surfaces, because in their experiments,microfacet models demonstrated the ability to successfully predict the shift in the peak of transmittance away from the refraction direction predicted by Snell's law, and other effects that rough transmission could show. BSDF is described as sum of BRDF and BTDF. The bidirectional reflectance basically has the same function with the Cook-Torrance model except for a difference factor of 4 in the denominator instead of π that previous methods usually use. The bidirectional transmittance is also a function of incident direction, view direction and surface normal. To make the BSDF model better match the measured transmission data, the authors proposed the widely accepted GGX distribution for microfacet distribution function, where the exponent component in standard Beckmann distribution (Spizzichino & Beckmann, 1963) is replaced with a simpler polynomial, and corresponding shadowing-masking function. The GGX distribution has a narrow peak and stronger tails which allow GGX to provide closer match to the measured data for some materials, especially when the incident and refraction direction are close to grazing angles.

In microfacet models, a detailed microsurface is replaced by a simplified macrosurface with a modified scattering function (BSDF) that matches the aggregate directional scattering of the microsurface (i.e. both should appear the same from a distance). This assumes that microsurface detail is too small to be seen directly, so only the far-field directional scattering pattern matters. (Walter et al., 2007, p. 197). This assumption is safe when the viewer and source light are at a distance. But with the development of display hardware, the demands on lighting in high definition images also increase. Detailed glitches or scratches may be required on certain surfaces, which cannot be realised by empirical or mathematical assumptions of distribution functions. Therefore high-resolution normal maps and deterministic calculation are also used for new solution for realistic reflection (Yan et al., 2014). Instead of calculating an approximated proportion of reflections from empirical distribution functions, Yan et al. (2014) got the proportion of microfacets where normals coincide with the half vector directly from a normal map. In these normal maps, resolutions are significantly higher than rendering needs so that pixel footprint can be used on the maps to get normal patches within a single pixel. Then the reflection intensity can be calculated summing up the contributions of chosen microfacets.

2.4 Applications of Neural Networks in Computer Graphics

Artificial neural networks (ANN), or simply neural networks (NN) are a solution that allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, where complicated concepts can be learned by building them out of simpler ones (Goodfellow, Bengio, & Courville, 2016). Neural networks are designed to solve problems that are hard to describe formally with a list of mathematical rules. The research into neural networks can be dated back to early 1940s (McCulloch & Pitts, 1943), but became very popular after the "third wave" of research started around 2006, where neural networks were widely used for those hard-to-describe tasks, such as image classification (Krizhevsky, Sutskever, & Hinton, 2012), speech recognition (Hinton et al., 2012), or machine translation (Mikolov, 2012), and were proven to be effective with the help of increasing computational resources and dataset sizes (Goodfellow et al., 2016). Neural networks can also be used for regression tasks, since they solve complex problems training a combination of multiple linear and nonlinear units to fit existing models with sufficient data and have been proved to have the ability to approximate arbitrary functions (Cybenko, 1989; Hornik, 1991). They are now widely used for fitting complex or unknown mathematical models, and demonstrate the capability of generalization (Dahl, Jaitly, & Salakhutdinov, 2014).

Neural networks have also been used for various tasks in computer graphics. Because of the ability of fitting complex models with sufficient data, neural networks now serve as data-driven approaches, to reduce human labor (Zhang et al., 2018), to approximate and accelerate complex calculation (Tompson et al., 2017), or to serve as a bridge between low-dimensional inputs and high-complexity outputs (Holden et al., 2017), or between ambiguous inputs and specific outputs (Wang, Kang, & Li, 2015) which are typical in computer graphics tasks, and are proven to be effective.

To reduce the demands of inputs from animators, neural networks have been used in the data-driven approaches for character animation. The production of character animation can be a exhausting and tedious work. Especially in the cases of crowd animation or character control in video games, where the amount of generated motion, rather than the performance of the character, is the core concern of animators. Walking, hopping and many other kinds of motions in character animations can be much more complex and abstract than simple vertex translations or rotations which can be usually represented with one single matrix. These motions can hardly be described with mathematical rules. Therefore, as attempts of solving the problem using data-driven approaches, much research effort has been put into the production of locomotion and the transitions between them, for smoother character control in video games, or rapid prototype for character animation production. Most earlier research is based on linear bases such as principal component analysis (PCA) (Liu & Schisterman, 2004), where motion data can be projected to lower-dimensional space as a model for further processing and synthesis. But non-linearity will be introduced into these complex motions when multiple linear movements are integrated as a whole. A longstanding goal of data-driven approaches for generating motions is to make use of existing databases to reduce the demands of inputs from animators or users, while keeping the results smooth, natural and controllable. It makes computer animation a challenging task for traditional regression methods. Approaches using neural networks (Holden et al., 2017; Zhang et al., 2018) found solutions, since neural networks perform well dealing with complex data and high-level information, and can introduce non-linearity into the processing. Another important advantage of neural network is that the limit for the amount of input parameters is higher. Because existing motion data is no longer directly used in the synthesis of new motions, but used as the guidance to train parameters for regression, this significantly broadens the types of processing that can be introduced into the research and production. For example, aside from the effort to generate animation with less input from user, another import goal of researchers is to generate adaptive animation automatically according to specific condition limits, such as terrain or route. In this endeavor, Holden et al. (2017) provides a successful example for the application of neural network in character animation production. Instead of dividing the system into several parts, the authors designed an end-to-end neural network which takes the user command as well as terrain data and character states in the previous frames and forms an integrated input. The resulting character can smoothly move along a user designed route as well as following their real-time control. Zhang et al. (2018), based on the core idea this work, proposed a new neural network for quadruped animals. Motion data captured with animals are usually ambiguous and hard to control or label. To solve the problem, the authors design a gating network which can dynamically update the weights of the original motion prediction network. Then the system can learn from even non-periodic and unstructured motion data, and allow users to skip the tedious and annoying labeling work in data preparation (Zhang et al., 2018).

In addition to being a bridge between user inputs and the data, neural networks are also used for accelerating complex computation in computer graphics, such as fluid simulation (Tompson et al., 2017) and global illumination (Ren et al., 2013). Simulation can be one of the most explored and critical areas in computer graphics technology. And the simulation of fluids is among the most popular areas of simulation. The movements of and interactions between these different lifeless materials are strictly controlled by physical laws. Therefore, there is not much artistic performance in fluid simulation as in character animation, but heavy computation caused by huge numbers of particles or interactions. Some researchers focus on the simplification of representation of the parameters, while some other researchers believe that the computations themselves can be approximated using regression, and then the simulation can be further sped up. Tompson et al. (2017) used deep convolutional neural networks to solve the problem. A new convolutional neural network architecture was proposed to replace existing Eulerian-based solver (Foster & Metaxas, 1996) with approximation after training. Like most of other researchers working on deep learning, the authors also proposed series of domain-specific optimizations to improve both the quality and speed of the network, and to achieve a balance between the efficiency and the performance.

Ren et al. (2013) designed an acyclic feed-forward neural network to solve the problem of global illumination or indirect illumination (Sillion & Puech, 1994) and got impressive results. The authors separated the indirect illumination components from the reflectance equation, and took it as an independent function of geometrical parameters and other parameters in BDRF. Then Radiance Regression Functions (RRF) was proposed to approximate this calculation of indirect illumination. RRF describes indirect illumination as an R^{12+n_p} to R^3 regression problem, where 12 is the dimension of geometrical parameters, n_p is the number of other adopted parameters from BRDF, R_3 represents the 3 channels of RGB color space. As the symbol R^{12+n_p} suggests, the authors designed input vector for the network which includes material properties (n_p) as well as geometrical parameters (normal, position, lighting and view vectors), and the dimension of material properties can be decided according to the needs for regression. An acyclic feed-forward neural network (Fig. 2.5) was then used to fit the function and then directly maps the designed input parameters to indirect illumination values.

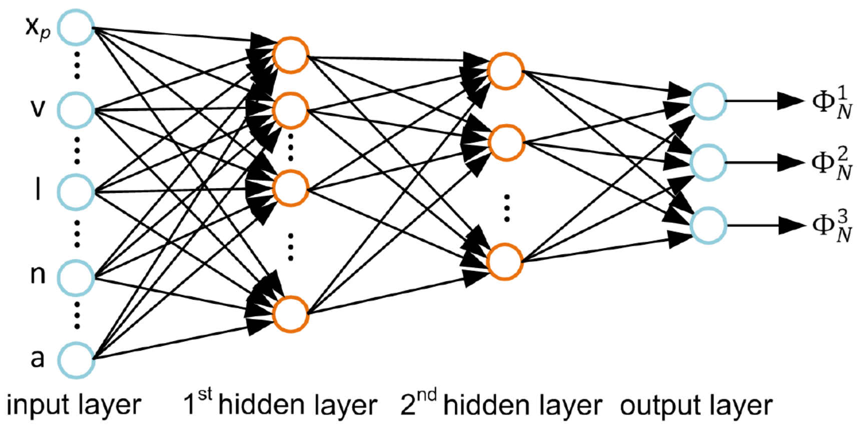

Figure 2.5. The network structure used by Ren et al. (2013). (Ren et al., 2013)

In addition to the impressive result, an important contribution of the research of Ren et al. (2013) is the indirect proof of applicability of neural networks in lighting models. Though the feed-forward neural network is used to fit RRF, the inputs and outputs of the network can also be used in BRDF or BSDF. The basic idea of R^n-to-R^3 regression sets the connection between those parameters and caters to the application of neural networks. Therefore all the lighting models using the same inputs and outputs should be approximated using neural networks as well. It is worth mentioned that Ren et al. (2013) described that position view vector and incident vector describe a minimal set of factors that determine the indirect illumination, while they adopted an input vector that is much more complex (R^{12+n_p}). No experimental results or mathematical proof were given in the literature, and that may set a future direction for research on neural networks application in lighting models.

More similar to this research, Nalbach et al. (2017) used convolutional neural network (CNN) to perform screen space shading. Attributes were stored in deferred shading buffers, including various of geometric and material parameters, and reflectance calculated using traditional lighting models. These attributes were then used as the inputs of CNN and rendered screen space effects such as ambient occlusion, directional occlusion, sub-surface scattering and depth-of-field. The authors adopted a U-shaped network structure and an autoencoder fashioned for the task. A total of 12 buffers and 48-dimension inputs are adopted and structural similarity was taken as the loss function. The implementation could perform a wide range of shading effects in real time. But the training time and the arrangement of pipeline weren't thoroughly discussed. The focus of the research is still on the efficiency of heavy computation and the matching between information and complex effects.

As the basis and inspiration of this research, McGraw and Garcia (2019) implement shader-based neural networks and use the implementation for creative works and image processing. The implemented neural networks demonstrate ability of synthesizing images and patterns from scratch or manipulating given images using weights in different layers and activation functions. McGraw and Garcia (2019) implement fully connected neural networks of different depth with data structures and functions defined completely in fragment shader, and then examined the effects of different hyperparameters and activation functions. As shown in Fig. 2.6, it is observed that parameters of different depth have different effects on the result, where shallower layers has an effect related more to the input, pixel coordinates, while deeper layers has an effect related more to the output, color intensities. Activation functions, on the other hand, can introduce abstractions into the result according to its own characteristics. The implementation and experiments have proved the possibility of neural networks to be integrated into graphics pipeline and to be used as a creative tool, which can be the prerequisites of this research.

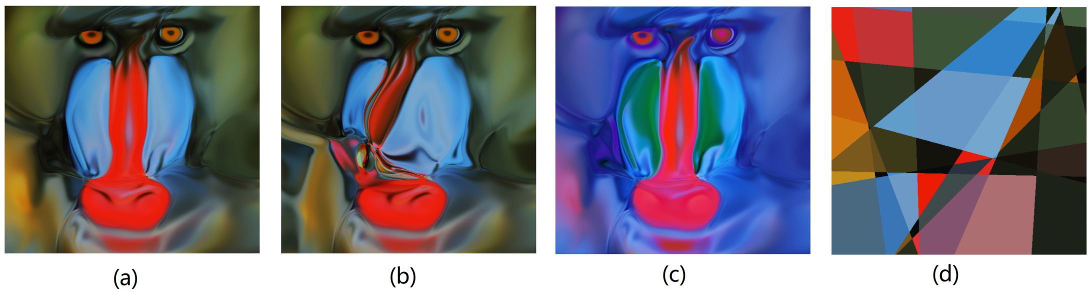

Figure 2.6. Image manipulation results of shader-based fully connected network. (a) shows the fitting result of a 20-layer fully connected network, (b) shows the result of manipulating 60th submatrix, (c) shows the result of manipulating 320th submatrix and (d) shows the result of using step function as activation function.

Although neural networks have been used for accelerating global illumination (Ren et al., 2013) and for image-based relighting (Ren et al., 2015; Xu et al., 2018), none of the previous research involved the exploration and discussion of fitting process itself. The appropriate network structure for the fitting, the number of necessary parameters, the lighting models that can be approximated using neural networks and corresponding precision haven't been discussed before, while the complexity and diversity of lighting models make it worthwhile to be explored. In addition to the traditional usage of classification or regression, neural networks are recently used for creative works as well (Bricman & Tudor Ionescu, 2018; Broad & Grierson, 2017). Therefore another potential research topic is to give a new perspective of lighting models through neural network approximation. The size of network thus will be limited for interactive purposes, so that the network parameters can be new handles or interfaces for artists to create interesting lighting effects.

CHAPTER 3. FRAMEWORK AND METHODOLOGY

This thesis describes a quantitative research project which is assessed in terms of computation time, prediction errors of shader-based fully connected neural network structures, and different parameter settings. This chapter provides the framework and methodology to be used in the research, including hypotheses, variables, and experiment procedures.

3.1 Research Design

Experiments are conducted to assess the impact of

- different neural network settings (hyperparameters and activation functions) for fully connected neural network,
- approximating different lighting models (Phong model, GGX BRDF model),
- with different output color spaces.

Experimental results of different settings are evaluated using mean squared error (MSE).

3.2 Hypotheses & Variables

The following sections will discuss the hypothesis, variables, and the measure for success.

3.2.1 Hypothesis

The hypotheses for this research are:

- Phong lighting model and GGX lighting model can be approximated by shader-based FCNN.

3.2.2 Variables

This subsection discusses the variables in the experiments.

Independent variables of the experiments are the hyperparameters and settings of FCNN:

- Number of layers in FCNN;
- Width of each layer;
- Activation functions adopted in FCNN;
- Color spaces for outputs.

To measure the accuracy and performance of neural networks approximating lighting models, the dependent variables will be:

- Fitting errors (mean squared errors) between results from neural networks and mathematical models;
- Runtime measurement.

3.2.3 Measure for Success

This subsection discusses how the results of the experiments will be evaluated.

The goal of this research is to discuss how well the adopted neural network can perform with certain settings and data. Experimental results are evaluated using mean squared errors of all the fragments between the outputs of traditional lighting models and neural networks. Resulting images are shown to conduct visual comparison.

3.3 Experiment Procedure

The goal of this research is to examine the performance of neural networks approximating lighting models. Therefore, the experimental procedure will include the training phase: generating training data by rendering with traditional lighting models, training and validating the network; and testing phase: rendering new objects using traditional lighting models and neural networks and evaluating the results.

First, the program using traditional lighting models calculates the adopted outputs (intensity or reflectance) of large amounts of virtual objects, under different settings of the camera, and records the results. Then, about 80% results from these traditional lighting models are used to train the neural network with different architecture, using part of or all the inputs. The remaining 20% results are used to validate the neural networks using adopted evaluation approach in each epoch.

In the testing phase, rendering results using traditional lighting models and neural networks will be rendered simultaneously and used to evaluate the performance of neural networks with different structures and settings.

3.4 Environment

Experiments will be conducted under the following environment:

- Intel Core i7-7700HQ CPU @ 2. 80GHz
- NVIDIA GeForce GTX1060 6GB
- 16GB Memory
- Windows 10 Home 64bit Operating System

3.5 Implementation and Preliminary Experiments

An OpenGL client program written in C++ and shaders written in the OpenGL Shading Language will be used in this research. Implementation of the main part will be in shaders, including necessary data structures, backpropagation and network structures. Traditional lighting models and experimental neural networks will be implemented in fragment shader. FCNN will be trained in the compute shader and tested in the fragment shader Data reading and writing, coordination of different processes and interactive functions will be implemented in client C++ program.

During the training phase, the running of program will be divided into three stages. In the first stage, training data will be sent into shaders implementing traditional lighting models to render a scene, and the input parameters and rendering results will be recorded in several buffers separately. In the second stage, shaders-based neural networks will read the data from different buffers and take them as inputs and outputs for training the network parameters and calculating training errors. The parameters will be passed to the client program for further processing. After several alternations of stage one and two, in the third stage validation data will be sent into shader-based neural networks. Then results from neural networks will be recorded and sent into shaders implementing traditional lighting models, together with the original input parameters to calculate validation errors. Then after multiple epochs of training, the parameters will be recorded and used for testing phase, where testing data will be used to render images and to conduct visual comparison. Preliminary results can be seen in the work of McGraw and Garcia (2019).

3.6 Summary

This chapter provided the framework and methodology to be used in the research research.

CHAPTER 4. EXPERIMENTS

To prove the possibility of FCNN learning lighting models and to find suitable network hyperparameters and settings, groups of experiments are conducted in this research. This chapter reports the experimental settings and corresponding results of these experiments, including the comparison of losses and loss curves. Images rendered using mathematical lighting models and using FCNN with different settings are also reported to conduct visual comparison.

A total of 64149 samples (pixels) are used in all quantitative experiments in this chapter. Samples are randomly split at 4:1 ratio into training and validation sets. Dimensions of inputs and outputs are controlled to be the same for all experiments in same section. Mean squared errors (MSE), as validation loss are reported to indicate the results. For testing, samples from different meshes are input into trained FCNN. Images are reported to indicate the testing results.

4.1 Model Hyperparameters Settings for Different Lighting Models

Traditional lighting models gets more complex as more theories and factors are included. The growing complexity of these models includes more parameters and more nonlinearity in calculation. Therefore, if ground truths are provided using different lighting models, suitable network structures can be different to have higher efficiency or even better results in practice.

Experiments are described in this section to explore the suitable network structures for two different lighting models. Since FCNN are used in the experiments, two model hyperparameters: the number of hidden layers (or the depth of FCNN) and the numbers of neurons in each hidden layer (or the widths of each hidden layer) are taken to describe the network structures.

It is worth mentioning that with sufficient training time and iterations, FCNN should be able to approximate mathematical models to any degree of desired accuracy (Cybenko, 1989). All the results reported in following sections are meant to indicate the difference between and characteristics in results with different experimental settings in practice.

4.1.1 Phong Lighting Model

Phong lighting model is a concise model and can be good for testing simple forward feed neural network structures. The traditional lighting model only have 4 constant parameters and 2 dot product. It is also representative and still widely used, thus in this research, it is taken as an example of concise lighting model that focuses on the reflectance intensity. Although color is not the primary consideration of original Phong lighting model, RGB values are still taken as the outputs of the network to better indicate results with color shifts and to align with the experiments in other subsections. Inputs of 6 dimension, with world-space normals and view directions, and outputs of 3 dimension with RGB values are used in experiments. As independent variables, numbers of hidden layers varying from 2 to 12, and widths of hidden layers varying from 2 to 12 are adopted in 36 groups of experiments separately.

Mean squared errors, as validation losses are given in Table 4.1, to show the results of shader-based FCNN approximating Phong model. Minimum validation losses of 20 epochs are adopted. Vertical rows show the MSE of FCNN with different numbers of hidden layers, and horizontal rows show the results of FCNN with different widths of hidden layers separately. Medians of 10-run-groups are reported as results for each experimental settings.

According to Table 4.1, with same training rates and epochs, FCNN with more parameters can generally be trained with higher efficiency. Validation loss decreases as the width of hidden layers increases until 10-12 neurons are applied in each hidden layer. It also decreases with the increase of the depth of hidden layers, but the results are not significantly different, especially when the widths of hidden layers are small. An inconspicuous saddle can even be observed around 8-10 hidden layers, when less than 4 neurons are applied in each hidden layer. Considering the time consumption of training, FCNN of a balance scale (with about 6 neurons in each hidden layer and 4 hidden layers) could be a better choice for fitting Phong lighting model.

Table 4.1.

Mean squared errors of different model hyperparameters learning Phong lighting model. Same learning rate ($\alpha = 0.00025, \beta_1 = 0.8, \beta_2 = 0.9$), same batch size of 8 and same number of training epochs (20) are set for all experiments.

Number of Neurons in Hidden Layers	**2**	**4**	**6**	**8**	**10**	**12**
2 Hidden Layers	0.002158	0.000736	0.000713	0.000569	0.000696	0.000565
4 Hidden Layers	0.001785	0.000693	0.000472	0.000428	0.000395	0.000380
6 Hidden Layers	0.002558	0.000657	0.000502	0.000512	0.000383	0.000353
8 Hidden Layers	0.001992	0.000497	0.000509	0.000476	0.000356	0.00351
10 Hidden Layers	0.002526	0.000480	0.000534	0.000460	0.000347	0.000325
12 Hidden Layers	0.002595	0.000571	0.000467	0.00042	0.000352	0.000367

However, the results cannot be fully indicated with only MSE. Since FCNN in this research is used for learning graphics-related models, images rendered are also given in Fig. 4.1 and Fig. 4.2.

In Fig. 4.1 results can be observed that without sufficient training, more hidden layers than necessary can cause convergent outputs, that is, blurry and low contrast images as rendering results. In Fig. 4.1 (a) clear structures of clothes fold can be observed, while Fig. 4.1 (b) indicates less structures and Fig. 4.1 (c) can only shows a gray image of rough shade. It is worth mentioning that MSE in Table 4.1 doesn't change much as the number of hidden layers increases. But the resulting images cab be different.

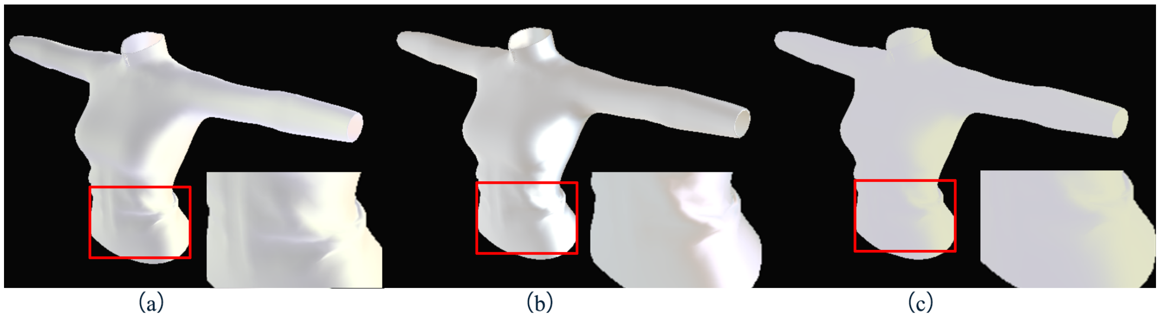

Figure 4.1. Example images rendered using FCNN with different numbers of hidden layers, width of 6. (a) shows the image using 8 hidden layers, (b) shows the image using using 12 hidden layers, and (c) shows the image using 16 hidden layers.

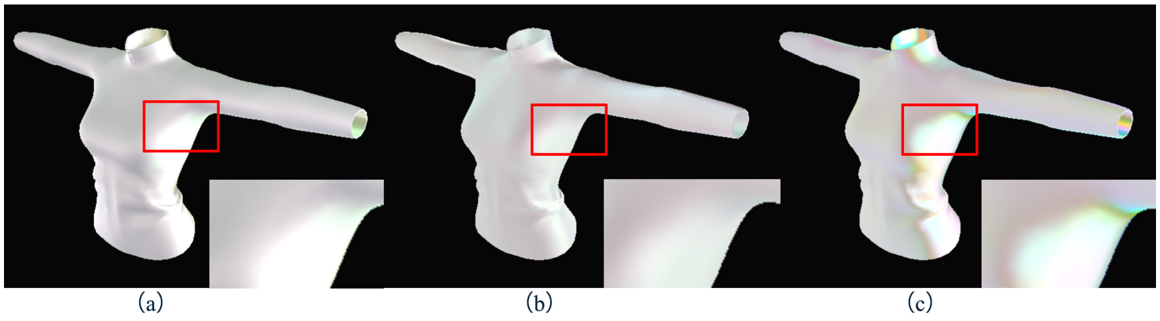

Figure 4.2. Example images rendered using FCNN with 6 hidden layers but different width. (a) shows the image using width of 8, (b) shows the image using width of 10 and (c) shows the image using width of 16.

In Fig. 4.2 results can be observed that hidden layer with more neurons than necessary may cause color shifts in resulting images. Fig. 4.2 (a) shows an image with clear gray scale, while in Fig. 4.2 (b) some red can be observed and Fig. 4.2 (c) shows an image with both red and green color shift. According to the research of Nalbach et al. (2017), color shifts can be weakened by increasing the dimension of input. Therefore it is reasonable to take color shift as an artifact caused by unmatched dimensions of data and parameters, either insufficient dimension of data, or excessive dimension of parameters. In this research, this artifact appears as overfitting caused by excessive width of hidden layers.

Testing results using a different mesh are reported in Fig. 4.3. Rendered images indicate that trained FCNN acts as an alternative of properly as Phong lighting model, not just a mapping between training inputs and outputs.

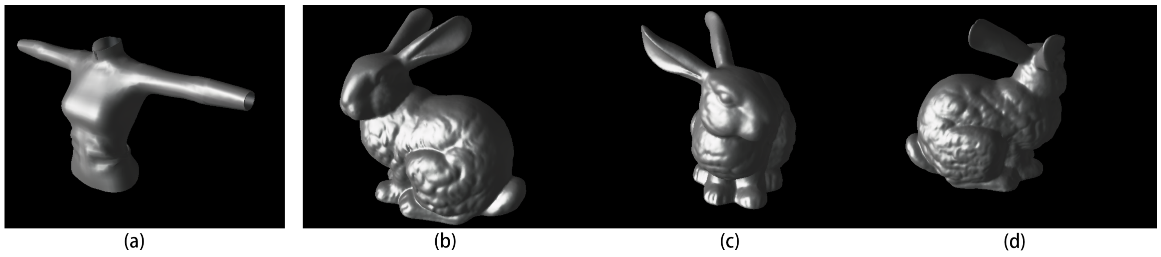

Figure 4.3. Example images rendered using FCNN with 6 hidden layers, width of 6. FCNN approximates Phong model using samples from mesh in (a), and (b),(c),(d) show testing results using a different mesh in different view angles.

4.1.2 Bidirectional Reflectance Model with GGX Distribution

BRDF models have more parameters and more non linear computations than simple Phong lighting model. Due to its difference from Phong model on both computation process and final outputs, BRDF model with GGX distribution is adopted as an example lighting model that focuses on color and materials in this section. Since color shift is an important part of material representation of BRDF models (Cook & Torrance, 1982), the outputs should be able to indicate the color of reflected light. In this section, RGB color space is adopted as output space. To focus on the effects of model hyperparameters, the input dimension is aligned with the experiments using Phong lighting model. Inputs of 6 dimension, with world-space normals and view directions, and output of 3 dimension with RGB values are used in experiments. As independent variables for this section, numbers of hidden layers varying from 2 to 12, and widths of hidden layers varying from 2 to 12 are adopted in the experiments in 36 groups of experiments separately.

Median mean squared errors of 10-run-groups are reported in Table 4.2, to show the results of FCNN approximating BRDF model with GGX distribution. Minimum validation losses of 20 epochs are adopted. Vertical rows show the MSE of FCNN with different numbers of hidden layers, and horizontal rows show the results of FCNN with different widths of hidden layers separately. Overall shader-based FCNN performs relatively worse fitting GGX model than fitting Phong models under most experimental settings. But validation loss decreases significantly with more parameters in FCNN, especially when 10 or more neurons are applied in each hidden layer. It can be expected since both the calculation and outputs are more complex than Phong model. Considering the time consumption and the gain of more parameters, FCNN of a balance scale (with about 10 neurons in each hidden layer and 6-8 hidden layers) could be a better choice for learning GGX lighting model.

Table 4.2.

Mean squared errors of different model hyperparameters fitting BRDF lighting model with GGX distribution. Same learning rate ($\alpha = 0.00025, \beta_1 = 0.8, \beta_2 = 0.9$), same batch size of 8 and same number of training epochs (20) are set for all experiments.

Number of Neurons in Hidden Layers	**2**	**4**	**6**	**8**	**10**	**12**
2 Hidden Layers	0.008342	0.007557	0.004520	0.003851	0.003769	0.004038
4 Hidden Layers	0.005567	0.003634	0.001836	0.002153	0.000945	0.000457
6 Hidden Layers	0.004549	0.003171	0.002882	0.001214	0.000467	0.000376
8 Hidden Layers	0.004451	0.002962	0.003205	0.001141	0.000467	0.000194
10 Hidden Layers	0.005627	0.003138	0.002678	0.000909	0.000365	0.000259
12 Hidden Layers	0.006864	0.003725	0.001571	0.000843	0.000256	0.000235

Resulting images are reported in Fig. 4.4 and Fig. 4.5. Fig. 4.4(a) and Fig. 4.5(a) show the ground truths, a metallic surface rendered using BRDF model with GGX distribution.

In Fig. 4.4 results can be observed that neural networks with fewer hidden layers can generate images with dimmer specular reflections, while images generated

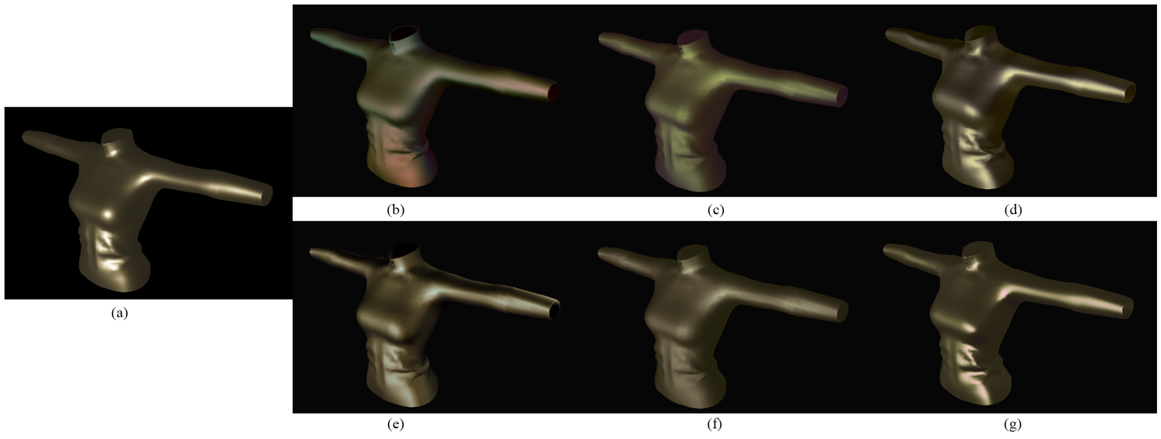

Figure 4.4. Example images rendered using FCNN with different numbers of hidden layers with width of 6. (a) shows the ground truth image, (b)∼(g) show the images rendered using neural networks with 4, 6, 8, 10, 12, 14 hidden layers separately.

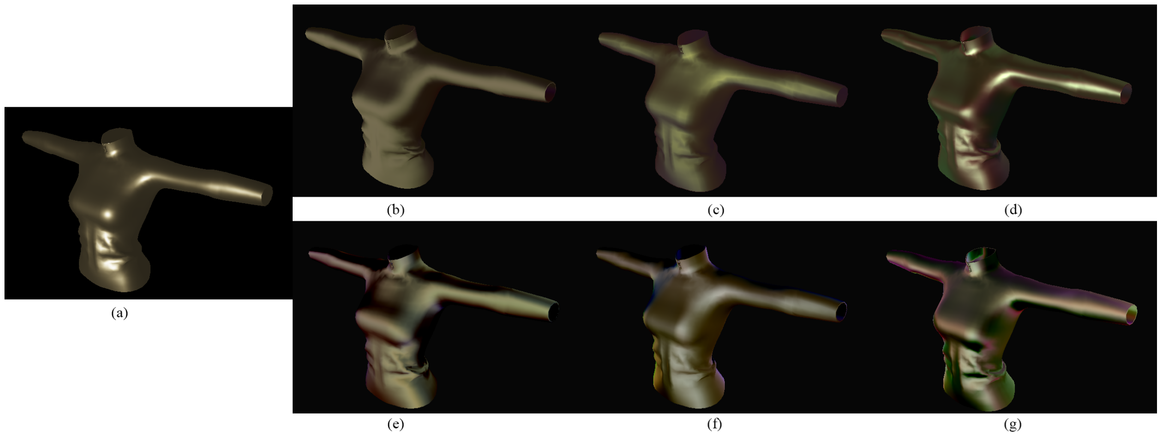

Figure 4.5. Example images rendered using FCNN with 6 hidden layers but different width. (a) shows the ground truth image, (b)∼(g) show the images rendered using neural networks with hidden layer widths of 4, 6, 8, 10, 12, 14 separately.

using deeper networks are not significantly different from the ground truth. It can also be observed that in Fig. 4.4 that the number of hidden layers doesn't affect color rendering. Color can be reproduced correctly no matter how deep FCNN is.

In Fig. 4.5 results can be observed that the rendered images change significantly as the width of hidden layers changing. In Fig. 4.5.(b), hidden layers

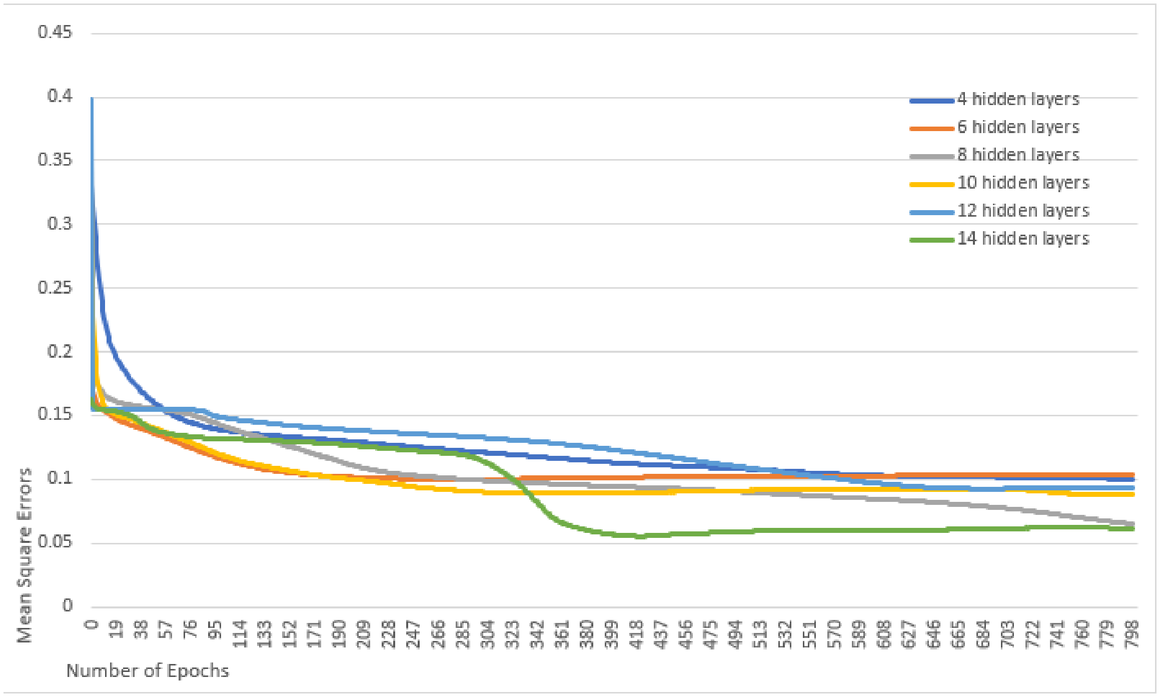

Figure 4.6. Example Validation loss curves of FCNN with different numbers of hidden layers. Same learning rates ($\alpha = 0.00003$), same width of hidden layers (8) and same number of training epochs (800) are applied.

with insufficient dimension (lower than the input dimension) generate images with incorrect specular reflections. The resulting image can still indicate the basic color and general direction of the lighting source. But the network also fails to indicate metallic surfaces. On the other hand, hidden layers with excessive neurons than necessary will generate images with random color shifts. It is worth mentioning that slight color shifts can also be observed in Fig. 4.5. The balance between the reproduction of metallic specular reflections and the Suppression of color shift should a point to consider when choosing hyperparameters for FCNN learning BRDF models.

Examples validation loss curves are reported in Fig. 4.6 and Fig. 4.7. A larger max training epochs of 800 and a smaller learning rate of 0.00003 are adopted to better indicate the difference caused by experimental settings. It can be easily

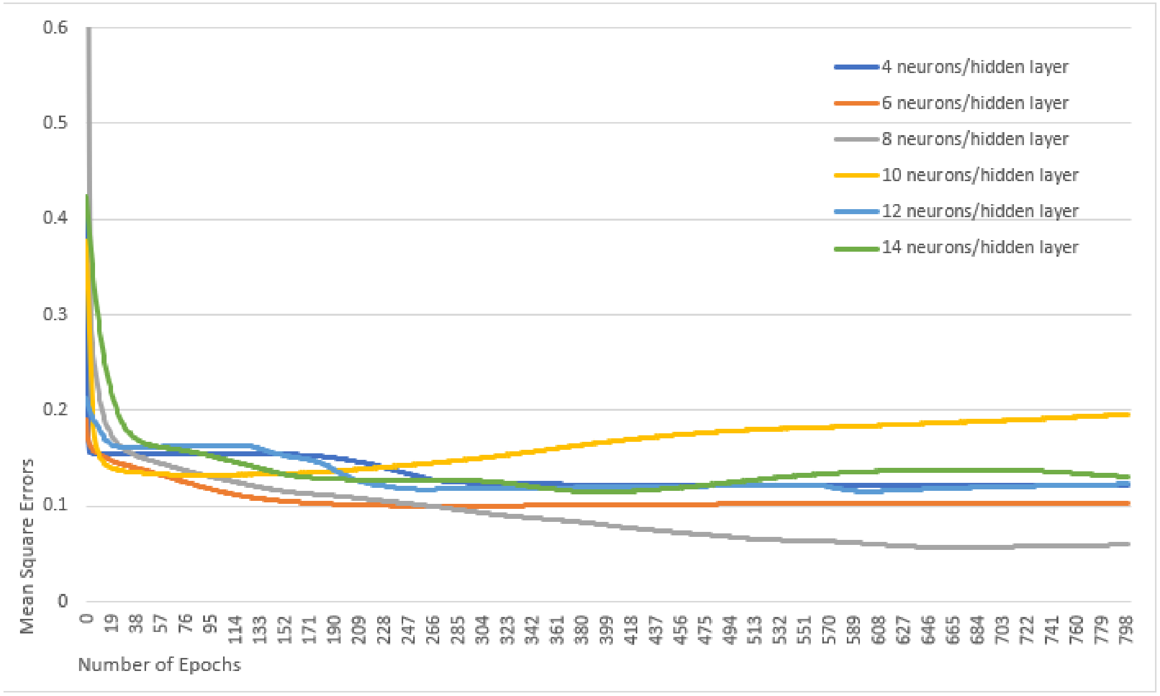

Figure 4.7. Example Validation loss curves of FCNN with different widths of hidden layers. Same learning rates ($\alpha = 0.00003$), same width of hidden layers (8) and same number of training epochs (800) are applied.

observed that generally FCNN with fewer parameters: fewer hidden layers or fewer neurons per hidden layer, descend slower. It is also worth mentioning that validation loss tends to ascend after reach the bottom, if hidden layers are wider. But validation loss seems to be more stable in deeper networks.

Testing results using a different mesh are reported in Fig. 4.8. Rendered images indicate that trained FCNN acts as an alternative of properly as GGX model, not just a mapping between training inputs and outputs.

4.1.3 Network using Different Widths in Different Layers

It can be inferred that wide hidden layers cause color shifts because excessive degrees of freedom (DoF) are offered to reproduce the color. Narrow hidden layers, on the contrast, fail to preserve all the information provided in inputs and result in

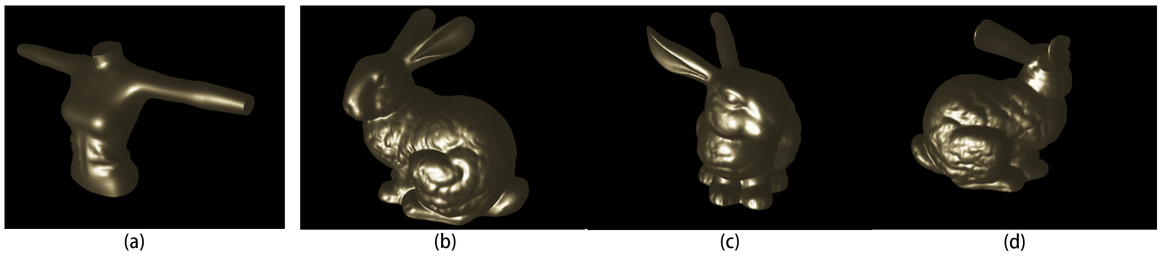

Figure 4.8. Example images rendered using FCNN with 8 hidden layers, width of 12. FCNN fits GGX model using samples from mesh in (a), and (b),(c),(d) show testing results using a different mesh in different view angles.

dull shades. Therefore, it is worth trying network structures with different numbers of neurons in different hidden layers. For instance, FCNN that has wider hidden layer on the input side to keep more information from inputs, and has narrower hidden layer on the output side to limit the DoF of outputs.

Experiments are described in this subsection to explore the effects of networks applying different width in hidden layers. BRDF model with GGX distribution is still used in this subsection. Parts of the experimental settings are similar to Subsection 4.1.2: inputs of 6 dimension and outputs of 3 dimension with RGB values are adopted in experiments. 8 hidden layers are applied in FCNN. As independent variables, fixed widths and variable widths are adopted in the experiments. 4 groups of experiments are conducted, using FCNN with 6 neurons in each layer, 13 neurons in each layer, 6 neurons on the input side and 13 neurons on the output side, 13 neurons on the input side and 6 neurons on the output side, separately. The structures are as follows in Fig. 4.9. Results are reported in Table 4.3.

It can be observed that FCNN with wider hidden layers (13 neurons) performs better than FCNN with narrower hidden layers, which also matches the results reported in 4.1.2. According to MSE reported in Table 4.3, FCNN with variable widths performs better than the one with widths of 6. FCNN with 6

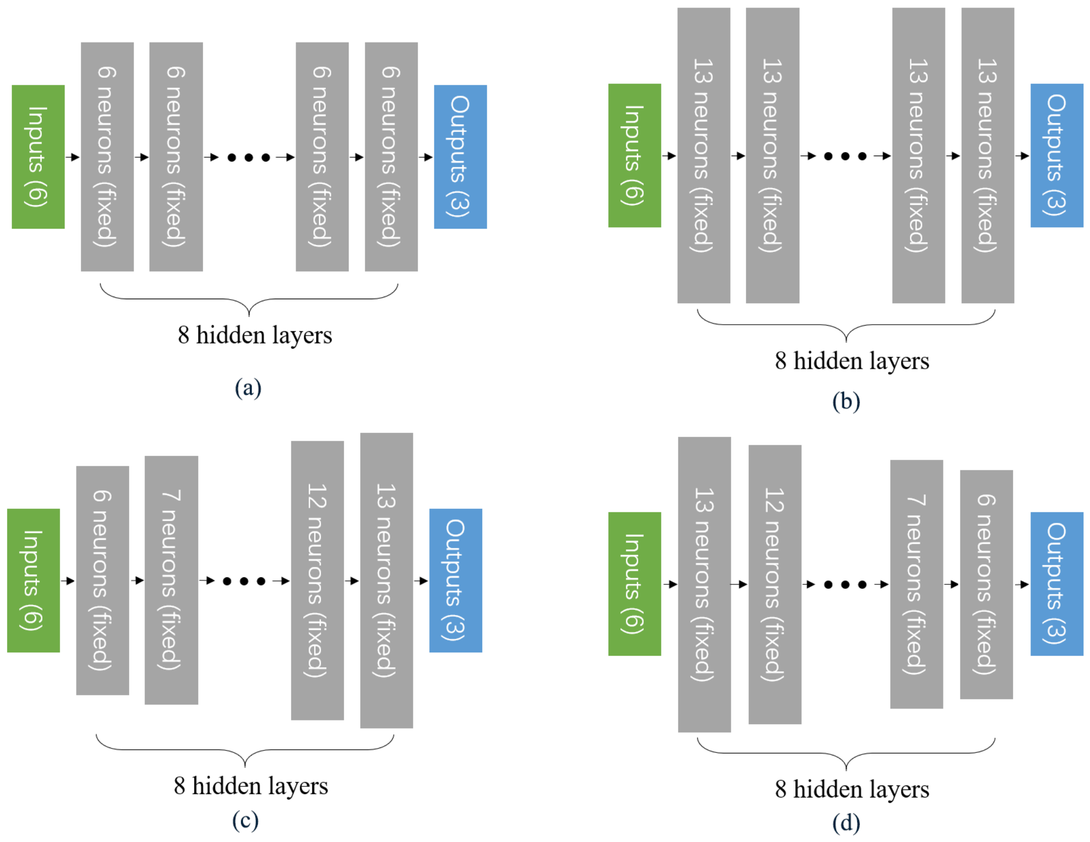

Figure 4.9. The network structures used in Subsection 4.1.3.

Table 4.3.

Validation loss of FCNN fitting GGX model, using different numbers of neurons in hidden layers. 8 hidden layers are applied in all experiments. Same learning rate ($\alpha = 0.00025, \beta_1 = 0.8, \beta_2 = 0.9$)*, same batch size of 8 and same number of training epochs (20) are set for all experiments. Medians of 10-run-groups are reported as results for each experimental settings.*

Number of Hidden Layers	**6(fixed)**	**13(fixed)**	**6 to 13**	**13 to 6**
Loss	0.000623	0.000134	0.000218	0.000106

neurons on the output side and 13 neurons on the input side may even have better performance than FCNN with 13 neurons in all hidden layers. To better indicate

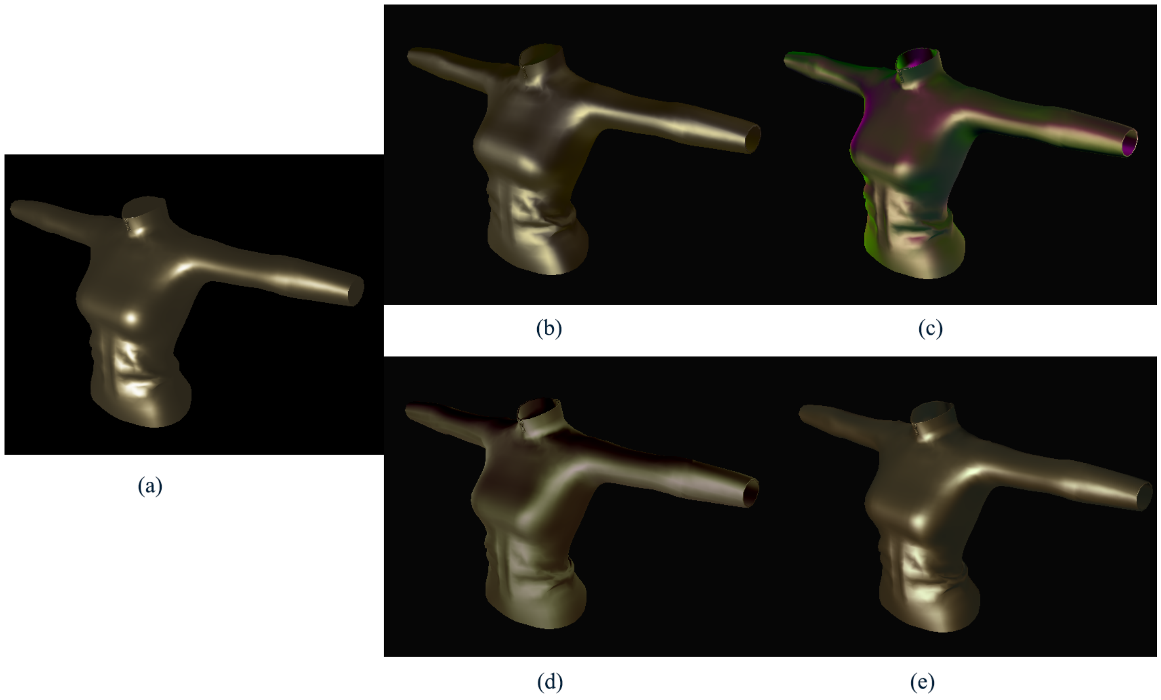

Figure 4.10. Example images rendered using FCNN with fixed widths and variable widths. (a) shows the ground truth. (b), (c), (d), (e) show the image rendered using FCNN of Fig. 4.9 (a), (b), (c), (d) separately

the result, some rendered images are also reported in Fig. 4.10 as examples. It can be observed that FCNN with fixed widths in each hidden layers, as shown in Fig. 4.10 (b) and Fig. 4.10 (c), may still indicate corresponding problems of incorrect specular reflections and color shifts separately. In Fig. 4.10 (d), rendered image still may show the problem of incorrect specular reflection, which can be seen in resulting images of FCNN with narrow hidden layers. In Fig. 4.10 (e), an image rendered using FCNN with narrower image on the output side. Differences are hard to be visually detected between Fig. 4.10 (e) and the ground truth, which also matches the data reported in Table 4.3.

4.2 Selection of Activation Function

Activation functions are applied in feedforward neural networks to introduce nonlinearity (Leshno, Lin, Pinkus, & Schocken, 1993). They have also been developed as neural networks are introduced for a variety of tasks. In addition to giving nonlinearity to the outputs of the previous layer, activation function also determines the distribution of inputs of the next layer. Therefore, it plays an important part in the network structure and training efficiency.

In this subsection, experiments and several example curves are reported to explain the selection of activation functions. BRDF model with GGX distribution and FCNN with 8 hidden layers and 8 widths is used for the experiments. hyperbolic tangent function, sigmoid function, ReLU (Nair & Hinton, 2010) and leaky ReLU (Maas, Hannun, & Ng, 2013) with $a = 0.05, 0.1, 0.2$ are used as independent variables for this subsection. Note that all these functions theoretically can act as activation functions properly in FCNN. But experimental results indicate that some activation functions may fail fitting lighting models in some cases. To better indicate the effects of each activation function to approximate lighting models, examples results are reported as curves in Fig. 4.11. Max training epochs of 800 and learning rate of 0.00003 are adopted for the curves.

Fig. 4.11 shows that hyperbolic tangent function achieves the best performance. MSE stop dropping at the point 0.28633 when using sigmoid function or ReLU as activation functions. 0.28633 is the error that one can get when all outputs are set to 0, using samples in this section. Activation values from the sigmoid function are always greater than or equal to 0, and not zero-centered. Therefore, it can be inferred that the sum of excessive neurons and hidden layers results in saturate activation values and causes blind drop of gradient. On the contrary, the activation values from ReLU are always 0 and stop parameters from updating when outputs from the previous layer are negative. This will “kill” neurons unless inputs with opposite signs are used in following training process.

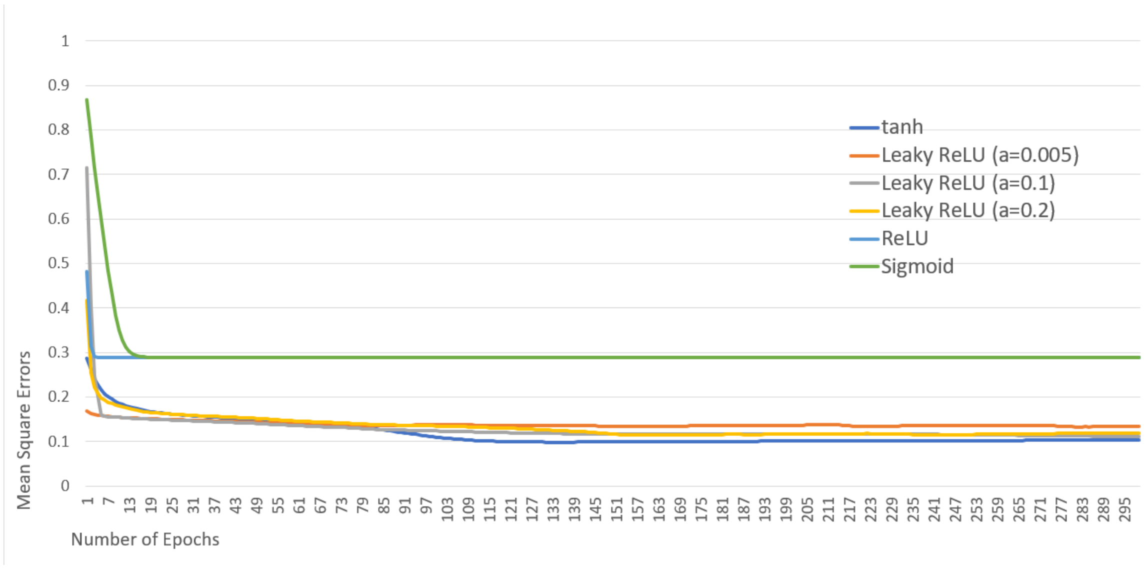

Figure 4.11. Example validation loss curve of FCNN with different activation functions.

Detailed observations of the training process proves that ReLU does kill all neurons after several epochs of parameter updates in these cases. ReLU may performs better on recognition or classification tasks. But it may not be suitable for regression tasks and approximating functions in this research. Since the problem of ReLU is caused by 0 activation values, leaky ReLU can be alternative in this research. According to Fig. 4.11, although not as good as hyperbolic tangent function, FCNN with leaky ReLU can also out perform the ones using ReLU and sigmoid function. The performance is better with relatively larger factor a.

4.3 Output Settings: Effects of Different Color Spaces

As the ground truth of training process, the space and distribution of outputs can also have effects on the training process. To explore the effects and possibilities of outputs in a shading-related research, color space, as one of the most important settings on outputs of rendering, can be a proper option for variable.

In this subsection, experimental results using of RGB, HSV or CIE as color space are reported and compared. BRDF model with GGX distribution, inputs of 6 dimension, outputs of 3 dimension and 8 hidden layers are adopted for FCNN in the experiments. Since the widths of hidden layers are related to the color of outputs, hidden layer with 4, 8, 16 neurons are applied to explore the effects of different color space on implemented FCNN. Results are reported in Table 4.4. Medians of 10-run-groups are reported as results for each experimental settings.

Table 4.4.

Validation loss (MSE) of FCNN using different color space.

Width of hidden layers	**4**	**8**	**16**
RGB color space	0.003399	0.001483	0.000335
HSV color space	0.001760	0.000563	0.000171
CIE color space	0.001312	0.000474	0.000161

It can be observed that FCNN with wider hidden layers perform better in all three color spaces. FCNN in HSV color space performs slightly better with more parameters, while FCNN in CIE color space performs better with less DoF. Sample images rendered in three color spaces are reported in Fig. 4.12. It can be observed that in Fig.4.12 (a)$\sim$(c), images indicate blurry structure in all three color spaces. On the contrary, with excessive dimension of parameters, images in different color space indicate different kinds of color shifts. In Fig.4.12 (e), rendered image in HSV color space indicates some shifts in saturation and value (intensity) channel. And in Fig.4.12 (f), image indicates color shifts in different color with the ones in RGB color space.

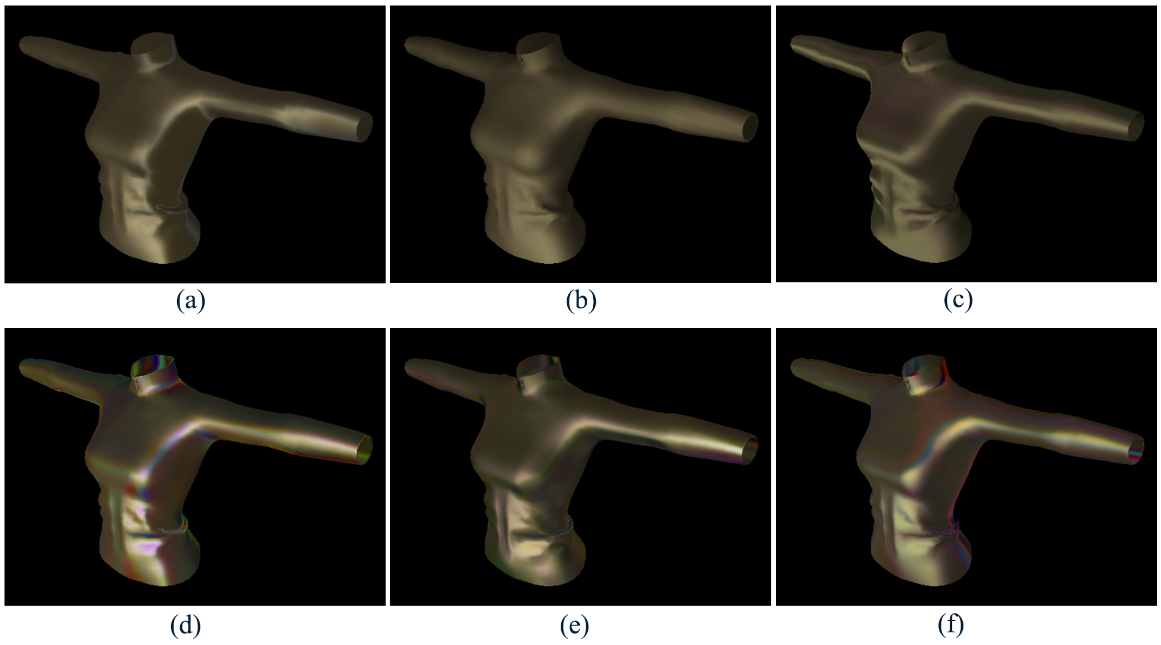

Figure 4.12. Example images rendered using FCNN in different color spaces, with different widths of hidden layers. (a)∼(c) indicate the images rendered using color space of RGB, HSV, CIE, separately with hidden layer width of 4, (d)∼(f) indicate the images rendered using color space of RGB, HSV, CIE, separately with hidden layer width of 16.

4.4 Manipulation of Parameters

As a attempts to provide alternative for lighting models, shader-based FCNN in this research should be able to allow interactions with the lighting results. To explore the possibilities of interactions, experimental results of manually edited parameters are reported in this section.

To avoid meaningless random parameters and deviation from the lighting models, FCNN is trained before edited in this section. A simple cartoon style

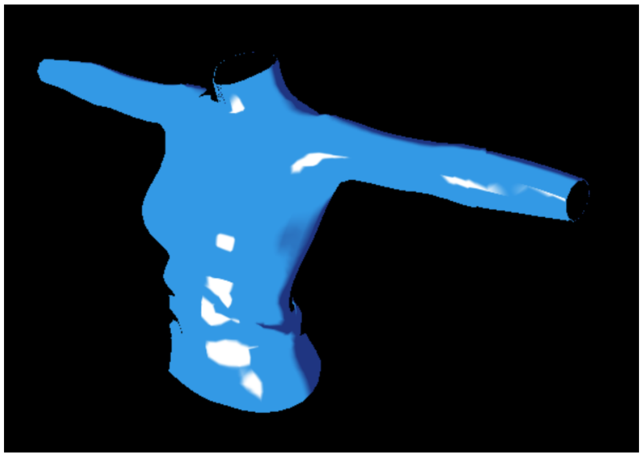

Figure 4.13. Image rendered using a simple three color shading. Ground truth of Section 4.4.

3-color shading is adopted as ground truth, for results with stronger style. On the basis of Phong lighting, it can be described as below:

$$C_o = \begin{cases} C_a, & threshold_1 > S_p \\ (1-a) * C_a + a * C_d, & threshold_1 > S_p > threshold_2 \\ C_d, & threshold_3 > S_p > threshold_2 \\ (1-a) * C_d + a * C_s, & threshold_4 > S_p > threshold_3 \\ C_s, & S_p > threshold_4 \end{cases}$$

where C_o is the output color, C_a, C_d and C_s are the colors for ambient light, diffuse reflection and specular reflection separately, set manually. S_p is the reflectance calculated using Phong lighting model, compared to the 4 manually set thresholds. Rendered image is reported in Fig. 4.13. Inputs of 6 dimension, with world-space normals and view directions, and output of 3 dimension with RGB values are used in experiments. 8 hidden layers with 12 neurons in each are adopted in the FCNN, learning rate of 0.00025 and batch size of 8 are adopted to fit the shading results.

After the training process, parameters, including weights and biases are edited manually to explore the effects of these parameters. Similar to the results of McGraw and Garcia (2019), the effects of parameters in shallower hidden layers

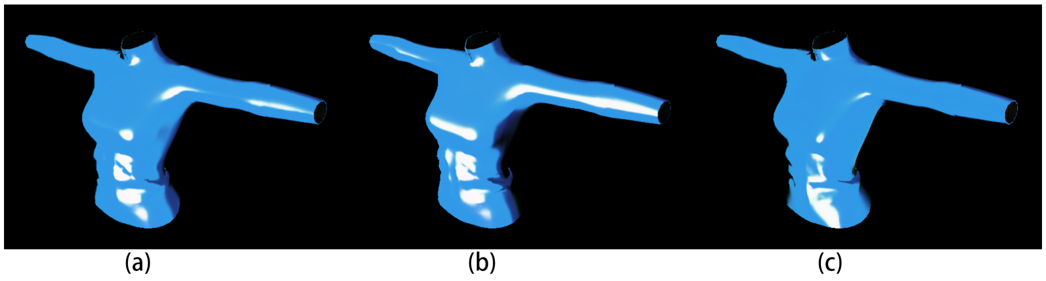

Figure 4.14. FCNN rendered images. (a) shows the result fitting Fig. 4.13, (b),(c) show the results editing the parameters in the first hidden layer.

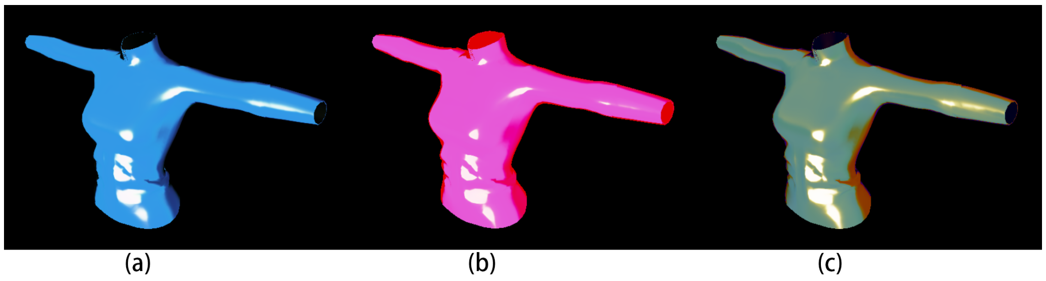

Figure 4.15. FCNN rendered images. (a) shows the result fitting Fig. 4.13, (b),(c) show the results editing the parameters in the eighth (last) hidden layer.

relates more to the input attributes, in this section, normals and view directions. As shown in Fig. 4.14, manipulating these parameters leads to change in the shapes and positions of specular reflection, which are similar to the results manipulating light source in mathematical lighting models. In Fig. 4.14 (b) image indicates a result similar to moving the light source to the left of the scene, Fig. 4.14 (c) indicate result similar to moving the light source down. On the contrary, manipulating parameters in deeper hidden layers has more effects on colors, which are the outputs of networks, as images shown in Fig. 4.15.

In Fig. 4.16, two examples are shown to report the results of editing parameters in the middle layers. It can be observed that in Fig, 4.16 (b), only the color in part of the shading changes, in Fig. 4.16 (c), only the color and shape of specular reflection changes but the rest of model doesn't. It is also worth mentioned

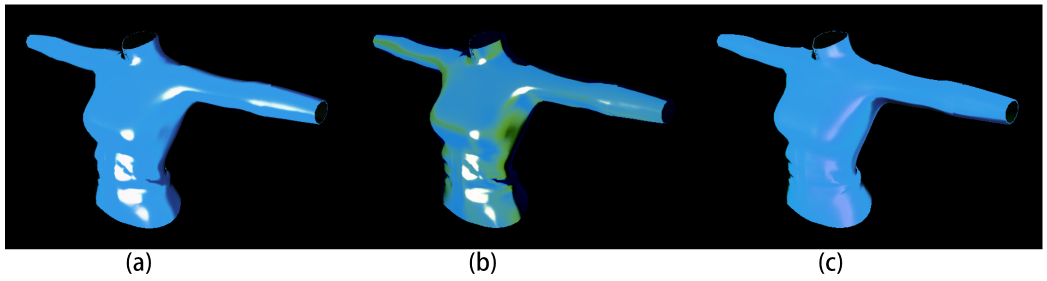

Figure 4.16. FCNN rendered images. (a) shows the result fitting Fig. 4.13, (b),(c) shows the results editing the parameters in the sixth and the third hidden layer, separately.

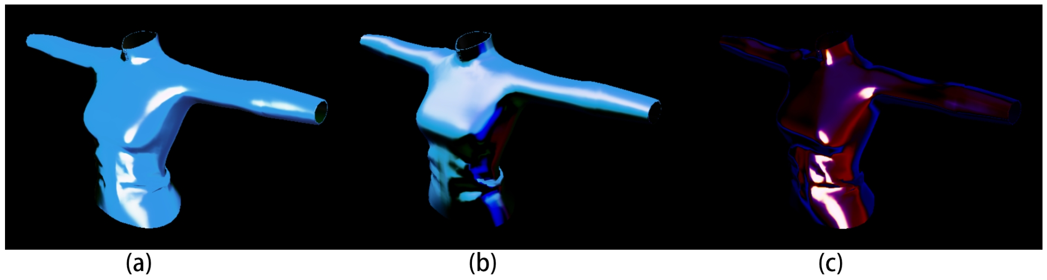

Figure 4.17. FCNN rendered images with manually edited parameters after fitting.

that in Fig. 4.14, color doesn't change while in Fig. 4.15, shape and position of specular reflection don't change.

In Fig. 4.17 images show some examples of manually editing parameters, with some stylized effects. In Fig. 4.17 (a) image shows a water color style shade. In Fig. 4.17 (b) different degrees of shades are represent with different color. In Fig. 4.17 (c) a stroke-like effect can be observed.

Testing results are also reported in Fig. 4.18 for the model in this subsection. Rendered images indicate that trained FCNN acts properly on a different mesh. Fig. 4.19 shows results of FCNN with edited parameters after fitting. Same specular reflection of green and shades of red can be observed in Fig. 4.19 (b)$\sim$(d) as well as in (a). Manually edited FCNN can also perform as an alternative of lighting model properly.

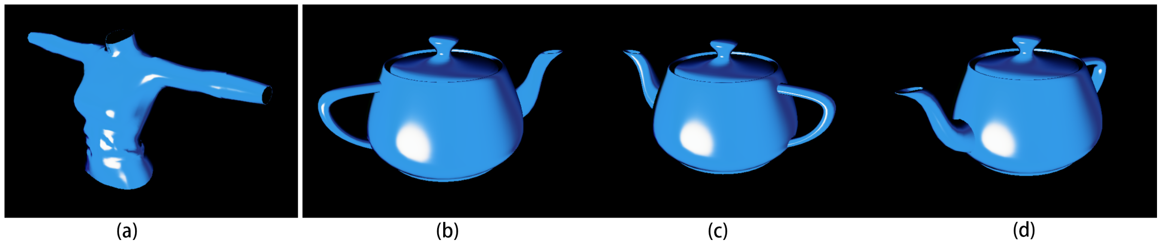

Figure 4.18. Example images rendered using FCNN with 6 hidden layers, width of 8. FCNN is trained using samples from mesh in (a), and (b),(c),(d) show testing results using a different mesh in different view angles.

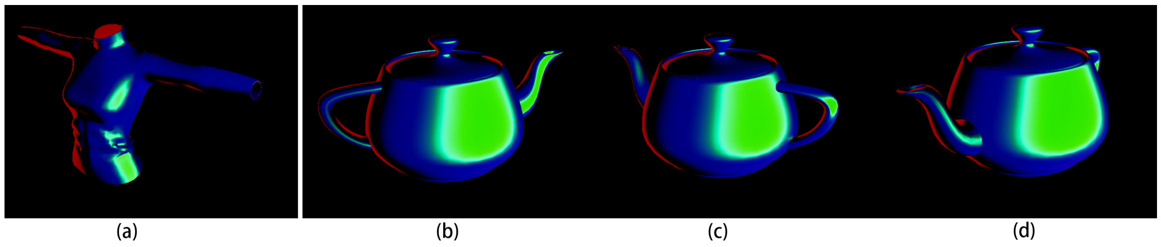

Figure 4.19. Example images rendered using FCNN with 6 hidden layers, width of 8. FCNN is manually edited after fitting ground truth in Fig. 4.13.(a) shows the editing result, (b),(c),(d) show a different rendered mesh in different view angles using edited FCNN.

4.5 Limitation of Fully Connected Neural Networks

Theoretically FCNN should be able to fit mathematical models to any degree of desired accuracy. But in practice, FCNN has its limitations as a basic and simple form of feedforward neural networks. Moreover, rendering results making use of textures don't have definite mathematical descriptions. FCNN may be insufficient to fit these results. In this sections, examples of such failed experiments are reported.

Fig. 4.20 shows the experimental results attempting to map normals and view directions, which are basic attributes related to lighting, to outputs of Phong lighting results with textures. It can be observed that the overall lighting can be well approximated using FCNN but some details are missing, such as the zipper in the middle and the dark parts on the cuff. It can be expected since such information

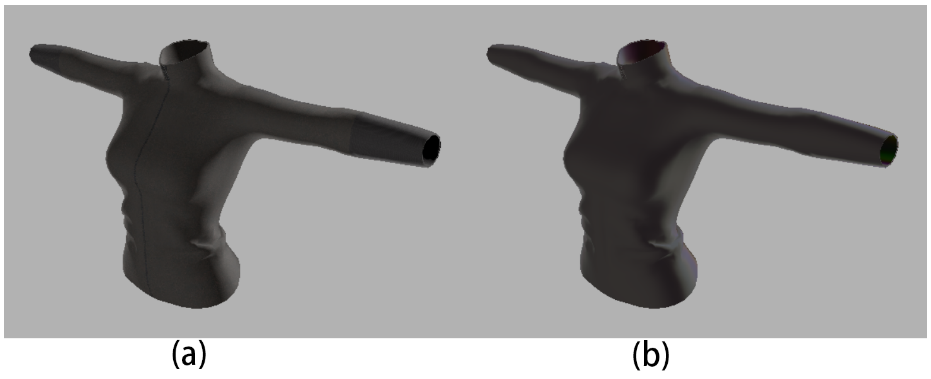

Figure 4.20. FCNN fitting result of Phong lighting model with texture. (a) shows the ground truth, (b) shows the result of FCNN using normals and view directions as inputs.

has nothing related to geometrical attributes. But different kinds of textures are important part of modern shading process. Therefore relevant attributes, texture coordinates and object coordinates are adopted in following experiments.

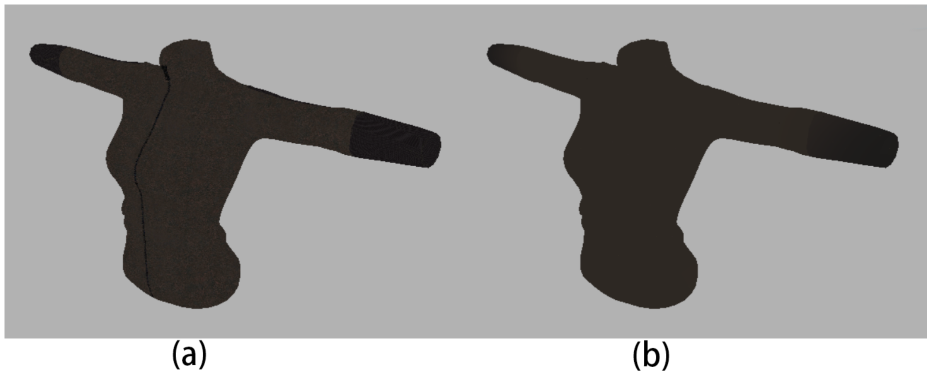

Figure 4.21. FCNN fitting texture using object coordinates and texture coordinates. (a) shows the ground truth, (b) shows the result of FCNN.

Fig. 4.21 shows the experimental results attempting to map an input of 5 dimension, with texture coordinates (2 dimension) and object coordinates (3 dimension), to a 3-channel texture. Basic color and the dark part on the left cuff are kept but more details, like the zipper and textures on the clothe, are missing. The result rendered by FCNN is overall flat and blurry . The noise-like texture is filtered to an average single color. It is worth mentioning that the validation loss (MSE) of result in Fig. 4.20 (b) is 0.000169, and MSE of Fig. 4.21 (b) is 0.000504. Both validation loss are satisfying while rendered images are not. Possible reasons and improvements will be discussed in next chapter.

4.6 Performance Analysis

In the last section, experimental results are reported to indicate the performance of implemented shader-based FCNN on training and rendering. As stated in Section3, experiments are conducted on a computer with Intel Core i7-7700HQ CPU, NVIDIA GeForce GTX1060 GPU, 16GB main memory and 6 GB video memory.

Table 4.5.

Average training time of 1000 batches with different batch sizes. Experiments are conducted on both CPU side and GPU side with different scale. Results are reported in milliseconds.

Batch size	**1**	**4**	**8**	**16**	**32**	**64**	**128**
4 hidden layer FCNN on CPU	0.032	0.059	0.112	0.186	0.461	0.649	1.166
4 hidden layer FCNN on GPU	0.381	0.434	0.437	0.450	0.444	0.446	0.449
8 hidden layer FCNN on CPU	0.059	0.130	0.238	0.312	0.604	1.156	3.377
8 hidden layer FCNN on GPU	0.476	0.475	0.461	0.464	0.471	0.472	0.484
16 hidden layer FCNN on CPU	0.075	0.188	0.318	0.647	1.129	2.681	4.660
16 hidden layer FCNN on GPU	0.888	0.896	0.900	0.903	0.922	0.920	0.930

Training times are reported in Table 4.5 in milliseconds. BRDF model with GGX distribution, inputs of 6 dimension, outputs of 3 dimension with RGB values and hidden layers of 8 dimension are adopted in experiments. 4, 8, 16 hidden layers are applied to explore the performance of implemented FCNN with different scale. 7 groups of experiments with different batch size are conducted. Average results of 1000 batches are reported. Training times on CPU side with C++ Implementations are reported as comparison to the training times in compute shader (GPU side) with OpenGL Shading Language. It can be observed that training times in compute shader are almost not affected by batch size. Implemented FCNN can be trained faster on CPU side with smaller batch size, but faster on GPU side when batch size is larger than or equal to 32.

Rendering performance is also reported in Table 4.6 and Table 4.7. Resulting images are rendered using the method of screen space rendering, which is an size-sensitive method. Implementation includes computation amount that proportional to the scale of the network. Therefore, performance are reported according to both the size of image and the scale of FCNN.

Table 4.6.

Average rendering time of 1000 frames, using FCNN with different scale and different image size.Results are reported in milliseconds.

Image size	640×480	1280×720	1920×1080
4×4 **FCNN**	0.879	1.697	2.177
8×8 **FCNN**	4.263	8.241	10.067
12×12 **FCNN**	12.754	24.002	29.653
16×16 **FCNN**	29.117	52.655	66.710

Table 4.7.

Average rendering time per-fragment, using FCNN with different scale. Results are reported in nanoseconds.

FCNN size	4×4	8×8	12×12	16×16
Time per fragment	15.8	79.3	237.3	538.9

Average rendering times of 1000 frames are reported in Table 4.6. Simple measures are adopted to reduce the rendering time of background pixels without data. Rendering times per-fragment are reported in Table 4.7 for different scale of

FCNN. It can be seen that rendering time grows rapidly as the network scale or image size increase. Implemented FCNN can run in real time (60 frames per second) within the scale of 8×8 under main.

4.7 Summary

This chapter provided the descriptions the experiments conducted in this research and reported the results. The next chapter provides analyses and discussions of this research.

CHAPTER 5. DISCUSSIONS

This chapter provides discussions on the experimental results and feasibility of implemented FCNN and method.

5.1 Feasibility of Implemented FCNN Fitting Models

According to the experimental results, implemented FCNN is able to approximate mathematical lighting models to about 97-99% accuracy with proper hyperparameters and settings. Failed experiments are also reported. Discussions on the feasibility of implemented FCNN fitting lighting models are provided in this chapter.

5.1.1 Performance of Implemented FCNN

As stated in Section 4.1, theoretically fully connected neural networks should be able to fit mathematical models to any accuracy with sufficient training time. Experimental results also indicate that with proper hyperparameters, implemented FCNN can fit Phong model and GGX model to satisfying accuracy. But with limited training time in practice, difference can still be observed between the effects of different network settings.

In Subsection 4.1.1 results are reported in Fig. 4.1 (b), (c) where shades are blurry, and in Fig. 4.2 (b), (c) where color shifts can be observed. It can be inferred that network parameters more than necessary may be more difficult to train, and lead to outputs which can not preserve enough information from the input. Instead, the network tends to reach the ground truth “in its own way” regardless of the inputs. It could be an average intensity that minimize MSE, when hidden layers are excessive, or complementary color shifts on samples with symmetrical input, when neurons in hidden layers are excessive.

In Subsection 4.1.2 results are reported in Fig. 4.4 (b), (c) where specular reflections are dim and flat. It can be inferred that neural networks with insufficient depth cannot provide sufficient nonlinearity to reproduce the “sharp” reflections of metallic materials. Because these reflections can be taken as sudden change of functions along the model surfaces. In Fig. 4.5 (b) specular reflections are in incorrect shape and position. A possible explanation is that fewer neurons are not able to keep enough information from the input, or not enough to reproduce the nonlinearity of GGX model. The resulting image can still indicate some information from the ground truth, such as the basic color and general direction of the lighting source, but fail to indicate the specular reflections. In Fig. 4.5 (f) and (g) color shifts can be observed. Nalbach et al. (2017) report similar color shifts in their research, and suppress the color shifts by increasing the dimension of input. It can be inferred that color shifts artifacts caused by excessive dimension of parameters. These parameters provide more degrees of freedom, which cannot be anchored properly by the ground truth and cause overfitting in the training process.

A possible reason is stated and tested in Subsection 4.1.3 for color shifts and incorrect specular reflections. Narrow hidden layers on the input side may mix and lost the information from different inputs, while wide hidden layers on the output side may provide excessive DoF. Experimental results are reported and prove that FCNN with designed variable widths can achieve the same or better performance with reduced parameters. Another possible reason for the problems shown in the figures could be the total amounts of network parameters. Although number and width of hidden layer can be easily taken as similar model hyperparameters, they contribute differently to the total amount of FCNN parameters. Because of fully connectivity, the amount of network parameters increase faster when the widths of hidden layers increase. Relatively minor errors (dimmer specular reflections) can be observed when the number of hidden layers are insufficient, while relatively serious ones (incorrect shape and position of specular reflections) can be observed when the width of hidden layers are insufficient. That could also explain that color shifts can be observed in Fig. 4.2 and Fig. 4.5, while the number of hidden layers has less effects on the color.

Implemented FCNN is feasible according to the accuracy fitting lighting models. Training time performance can also be promising with compute shader implementation, training 128 size batch within 1 milliseconds. Although rendering time performance depends on the image size and network scale, implemented FCNN in fragment shader can be run in real-time with necessary scale and image size, according to the results in Section 4.1 and 4.6. Optimizations are still needed for further tasks, but implemented FCNN is usable fitting mathematical lighting models.

5.1.2 Possible Explanations and Improvement for Failure Cases

Failure cases are reported in Section 4.5, where implemented FCNN is used fitting lighting and shading results with textures. Blurry images are generated with details lost. One possible reason for these failures may be that FCNN can process data of only limited dimension, and input information is not sufficient for learning textures. FCNN can be taken as a repeated classification using activation functions. Inputs are assigned to different partitions according to different values. Therefore it is difficult to learn detailed textures using limited number of fully connected layers. To generate images with textures or repeating patterns, information in surrounding vertices or fragment may be necessary to distinguish samples with similar geometrical attributes. Inputs of higher dimension may be needed. But such information is difficult for FCNN to handle, because of the huge amount of connections. Another possible reason may be the simple topology of FCNN, as the name suggests. In fully connected neural networks, all neurons from the previous layer contributing equally to all the neurons in the next layer. Every input channel of FCNN affect every output channel, which makes it more difficult to describe difference between similar samples.

Based on the analyses of reasons, a possible improvement for failures cases is to adopt new topology and even new class of ANN. Using larger scale FCNN may also solve the problem in an overfitting fashion. But sparse connectivity and structures like convolutional layer can be more efficient for the tasks of introducing more information and distinguishing similar samples.

5.1.3 Possible Applications of FCNN Approximated Lighting Models

Implemented FCNN can fit mathematical lighting models to satisfying accuracy. As an alternative to traditional lighting models, possible applications of FCNN approximated lighting models are promised by the DoF offered by network parameters and learning ability of neural networks. Fully connected layers offer only simple topology but relatively large number of parameters and degrees of freedom. As shown in Fig. 4.5 and Fig. 4.12, excessive DoF may cause color shifts fitting mathematical models, but also promises diversity of colors and images. Therefore it is feasible for FCNN to learn complex shading effects or stylized models. Furthermore, FCNN maps related data through learning. It is possible for implemented FCNN to fit ambiguous or undefined materials and effects.

5.2 Discussions on Network Parameter Manipulation

Section 4.4 reports results of manually edited parameters after fitting. As one of the motivations of this research, discussions on the experimental results and possible applications of network parameter manipulation is provided in this section.

5.2.1 Feasibility of Parameter Manipulation

It is feasible to achieve novel effects by editing network parameters after fitting certain lighting models. Feasibility of parameter manipulation is based on several properties of FCNN parameters reported in Section 4.4.

First of all, parameters in FCNN with multiple hidden layers affect only part of the attributes. As shown in Fig. 4.14 and Fig. 4.15, editing parameters in shallower or deeper hidden layers only changes the geometric properties or colors of reflections, but not both. The change caused by a single parameter is also limited in a certain range. This certainty of parameter manipulation promises it won't change the results much from the fitted model randomly, and meet the premise of reasonable manipulation result.

Results in Fig. 4.16 indicate the feasibility to achieve novel effects through parameter manipulation. Because of the high DoF provided by FCNN, editing network parameters can produce novel effects that variable parameter in traditional lighting models cannot offer. Results in Fig. 4.19 further indicate that manually edited parameters can also be applied on different meshes. FCNN provides simple mapping and relationship between inputs and outputs. It is stable to be generalized for different cases in lighting and shading, as long as same attributes are used for input.

5.2.2 Limitations and Improvements on Parameter Manipulation

Taking network weights and biases as general model parameters, FCNN provides high DoF which mathematical models cannot achieve. But a major problem of network parameter manipulation is the meanings and purposes of these parameters. Since neural networks are used for a wide range of purposes, and network parameters are mathematical equivalent, as “weighs” and “biases”, it is difficult to edit network parameters with certain purposes. While the parameters do have their own effects being manipulated, the effects vary from time to time after fitting. Another limitation is determined by the existence of activation function. Some parameters can be only edited within certain thresholds. Because manipulation crossing the decision boundary may cause unexpected effects in the results. It is difficult to determine the thresholds because they also vary in different fittings process. It can be easily saturated and generate unexpected effects.

A possible solution is to train and design a generalized model. Since DoF offered by FCNN is far more than traditional models, it is not necessary to fit a model every time before parameter manipulation. Designed model can be trained fitting similar type of material to be presented, then meaning and thresholds can be applied manually through research. Taking this designed model as basis, it is possible to generate considerable amounts of novel surface effects. If the aspects or attributes to be changed, or the effect to be achieved is determined, another possible solution is to introduce the attributes into inputs. FCNN maps related inputs and outputs through learning. Therefore it is suitable for ambiguous or complex relationships. Introducing attributes, for instances, “saturation of ambient color” or “brightness of high light”, into inputs, FCNN can be trained using labeled ground truth. After fitting different pairs of data, it is feasible to manipulate certain input to achieve desired effects.

CHAPTER 6. CONCLUSIONS

As stated in the Chapter 3, the goal of this research is to explore the possibility of shader-based FCNN learning lighting models and to find the suitable network structures and settings. In the research, a shader-based fully-connected neural network is implemented and tested using different lighting models and different hyperparameters. The experimental results, including the loss function and the comparison between rendered images using FCNN and ground truth images, prove that FCNN can approximate Phong lighting model and BRDF lighting model with GGX distribution well enough with proper network structure settings. Through groups of experiments, proper network structures and settings are found for the two lighting models. The efficiency of shader-based FCNN is also tested.

6.1 Summary of experiments

Experiments are conducted using Phong lighting model and BRDF model with GGX distribution. Experimental results and corresponding network structures and settings differ for different lighting models.

Two groups of experiments with control variable are conducted to determine the proper network structure for FCNN learning Phong model. According to the results, Phong model can be well approximated using FCNN with only 4 hidden layers and 6 neurons in each hidden layer. The number of hidden layers, or the depth of the network has relatively less effects on the performance of the network. Two groups of experiments with control variable are conducted to determine the proper network structure for FCNN learning BRDF model with GGX distribution. According to the results, GGX model can be well approximated using FCNN with 8 hidden layers and 10 neurons in each layer. Suitable hyperparameters are different for Phong model and GGX model. Results can be even better with sufficient training time, if FCNN of larger scale is applied.

Hypothesis is proposed that narrower hidden layer leads to incorrect specular reflections because of the loss of input information, and wider hidden layer leads to color shifts because of excessive DoF on the output side, no matter in which color space. Another group of experiments are conduct to verify the hypothesis. The results confirm the hypothesis, showing that FCNN with wider hidden layer on the input side, and narrower hidden layer on the output side performs better.

A group of experiments are conducted to determine the proper activation function for FCNN learning lighting models. According to the results, hyperbolic tangent function performs better learning lighting models. But ReLU and sigmoid function may cause failure in the tasks in this research. Leaky ReLU can improve the performance of FCNN by avoiding killing neurons during the training process.

Experiments are also conducted to explore the possibilities of shader-based FCNN editing parameters. Results indicate that implemented FCNN is able to fit cel-shading style model, to strengthen and to edit the model for rendering different stylized images. Therefore a possible application of implemented FCNN could be fitting and generating stylized shading models.

With proper network structures and settings, shader-based FCNN can approximate tested lighting models within acceptable training time and rendering time. FCNN approximations and manually edited parameters can be used for different meshes. Therefore it is possible to introduce neural networks into computer graphics production without external supporting libraries other than basic frameworks like OpenGL. Although both graphics and machine learning are still rapidly growing areas, it is worth trying to communicate and bring these two areas together.

6.2 Future works

Further works can still be conducted on the basis of these experiments. Hypotheses are tested in this research by proving that lighting models can be simply approximated by FCNN. But the direct lighting models are not as complex and expensive as many other heavy tasks like simulations or recognition. According to experimental results and of previous researches (Cybenko, 1989; Hornik, 1991; Leshno et al., 1993; Nalbach et al., 2017), shader-based neural networks should be able to approximate arbitrary shading-related models.

Future works for this research can include:

- Learning measured material data and textures;
- Implementing more complex network structures to introduce more information, such as surrounding pixels, into shader-based neural networks;
- Learning hand-painted and other stylized shading effects using shader-based neural networks;
- Applying shader-base neural networks on other graphics-related tasks.

LIST OF REFERENCES

LIST OF REFERENCES

Blinn, J. F. (1977). Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th annual conference on computer graphics and interactive techniques* (pp. 192–198). ACM.

Born, M., & Wolf, E. (2013). *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light.* Elsevier.

Bricman, P. A., & Tudor Ionescu, R. (2018, May). CocoNet: A deep neural network for mapping pixel coordinates to color values. *arXiv e-prints*.

Broad, T., & Grierson, M. (2017). Autoencoding blade runner: Reconstructing films with artificial neural networks. *Leonardo*, *50*(4), 376-383. doi: 10.1162/LEON_a_01455

Burley, B. (2012). Physically-based shading at disney. In *ACM SIGGRAPH* (Vol. 2012, pp. 1–7).

Cook, R. L., & Torrance, K. E. (1982). A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG)*, *1*(1), 7–24.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Dahl, G. E., Jaitly, N., & Salakhutdinov, R. (2014). Multi-task neural networks for qsar predictions. *arXiv preprint arXiv:1406.1231*.

Foster, N., & Metaxas, D. (1996). Realistic animation of liquids. *Graph. Models Image Process.*, *58*(5), 471–483.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural networks*, *2*(3), 183–192.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* The MIT Press.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., ... others (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, *29*.

Holden, D., Komura, T., & Saito, J. (2017). Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, *36*(4), 42.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257. Retrieved from `http://www.sciencedirect.com/science/article/pii/089360809190009T` doi: 10.1016/0893-6080(91)90009-T

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Curran Associates, Inc.

Leshno, M., Lin, V. Y., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, *6*(6), 861–867.

Liu, A., & Schisterman, E. F. (2004). Principal component analysis. *Encyclopedia of Biopharmaceutical Statistics. New York: Marcel Dekker*.

Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, *82*, 1013–1024.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (Vol. 30, p. 3).

Matusik, W., Pfister, H., Brand, M., & McMillan, L. (2003, 7). A data-driven reflectance model. *ACM Transactions on Graphics*, *22*(3), 759-769.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

McGraw, T., & Garcia, E. (2019). *Interactive neural networks for image processing and pattern generation.* (submitted)

Mikolov, T. (2012). Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, *80*.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814).

Nalbach, O., Arabadzhiyska, E., Mehta, D., Seidel, H.-P., & Ritschel, T. (2017). Deep shading: Convolutional neural networks for screen-space shading. , *36*(4).

Nicodemus, F. E., Richmond, J., Hsia, J., & Limperis, T. (1977). *Geometrical considerations and nomenclature for reflectance* (Vol. 160). US Department of Commerce, National Bureau of Standards.

Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of the ACM*, *18*(6), 311–317.

Ren, P., Dong, Y., Lin, S., Tong, X., & Guo, B. (2015). Image based relighting using neural networks. *ACM Trans. Graph.*, *34*(4), 1–12.

Ren, P., Wang, J., Gong, M., Lin, S., Tong, X., & Guo, B. (2013). Global illumination with radiance regression functions. *ACM Transactions on Graphics (TOG)*, *32*(4), 130.

Sillion, F. X., & Puech, C. (1994). *Radiosity and global illumination.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Spizzichino, A., & Beckmann, P. (1963). *The scattering of electromagnetic waves from rough surfaces.* New York, Paris.

Thermophysical properties of matter database tpmd. (2003). West Lafayette, Ind.]: CINDAS LLC.

Tompson, J., Schlachter, K., Sprechmann, P., & Perlin, K. (2017). Accelerating eulerian fluid simulation with convolutional networks. In *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 3424–3433).

Torrance, K. E., & Sparrow, E. M. (1967). Theory for off-specular reflection from roughened surfaces*. *J. Opt. Soc. Am.*, *57*(9), 1105–1114. Retrieved from `http://www.osapublishing.org/abstract.cfm?URI=josa-57-9-1105` doi: 10.1364/JOSA.57.001105

Walter, B. (2005). *Notes on the ward BRDF.*

Walter, B., Marschner, S. R., Li, H., & Torrance, K. E. (2007). Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th eurographics conference on rendering techniques* (pp. 195–206). Eurographics Association.

Wang, F., Kang, L., & Li, Y. (2015, June). Sketch-based 3d shape retrieval using convolutional neural networks. In *The ieee conference on computer vision and pattern recognition (cvpr).*

Ward, G. J. (1992). Measuring and modeling anisotropic reflection. In *ACM SIGGRAPH computer graphics* (Vol. 26, pp. 265–272). ACM.

Xu, Z., Sunkavalli, K., Hadap, S., & Ramamoorthi, R. (2018). Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (TOG)*, *37*(4), 126.

Yan, L.-Q., Hašan, M., Jakob, W., Lawrence, J., Marschner, S., & Ramamoorthi, R. (2014). Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Transactions on Graphics (TOG)*, *33*(4), 116.

Zhang, H., Starke, S., Komura, T., & Saito, J. (2018). Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, *37*(4), 145.