TU WIEN Informatics

Deep Learning for Lighting Evaluation

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Michael Steinkellner
Matrikelnummer 1705362

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Pierre Ecormier-Nocca, PhD

Wien, 25. März 2024

[illegible]

Michael Steinkellner

Michael Wimmer

TU WIEN Informatics

Deep Learning for Lighting Evaluation

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Michael Steinkellner
Registration Number 1705362

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Pierre Ecormier-Nocca, PhD

Vienna, March 25, 2024

[illegible]

Michael Steinkellner

Michael Wimmer

Erklärung zur Verfassung der Arbeit

Michael Steinkellner

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 25. März 2024

[illegible]

Michael Steinkellner

Danksagung

An dieser Stelle möchte ich mich bei meinem Betreuer Pierre Ecormier-Nocca für seine Unterstützung, insbesondere in den Momenten der Blockade, sowie für seine Mitwirkung an einigen Teilen der Arbeit bedanken. Ich möchte mich auch bei meinen Freunde und Freundinnen bedanken, die mich während dieses Prozesses immer wieder ermutigt haben. Zuletzt möchte ich mich bei meiner Familie bedanken, die mich nicht nur während meines Studiums, sondern mein ganzes Leben lang unterstützt hat.

Acknowledgements

I would like to take this opportunity to thank my supervisor Pierre Ecormier-Nocca for his support, especially in moments of blockage, and for his involvement in some parts of the thesis. I would also like to thank my friends who have encouraged me throughout this process. Finally, I would like to thank my family, who has supported me not only during my studies, but throughout my life.

Kurzfassung

Die Beleuchtung hat einen großen Einfluss auf die Atmosphäre eines Raumes. Sie spielt eine wichtige Rolle bei der Gestaltung von Innenräumen. Die Art und Weise, wie ein Raum beleuchtet wird, wirkt sich auf die Menschen in diesem Raum aus. Während ein Mensch dieses Konzept versteht und ausdrücken kann, wie eine Szene auf ihn wirkt, ist ein Computer derzeit nicht in der Lage, die gleiche Art von Analyse durchzuführen. Um eine Lösung für dieses Problem zu finden, wird in dieser Arbeit versucht, ein Convolutional Neural Network zu implementieren, welches Deep Learning verwendet, um Szenen anhand ihrer Lichtstimmung zu klassifizieren. Zu diesem Zweck habe ich einen Datensatz erstellt, der aus Bildern besteht, die verschiedene Lichtstimmungen repräsentieren. Die von mir verwendete Deep-Learning-Technik ist das Transfer-Learning, bei dem ein bereits vortrainiertes Netz, in diesem Fall VGG16, verwendet wird. Zuerst habe ich einen Multi-Klassen-Ansatz ausprobiert, bei dem alle Klassen von einem Klassifikator bewertet werden. Später ging ich jedoch zu einer binären Klassifikatorstruktur über, bei der der Klassifikator nur lernt und vorhersagt, ob ein Bild zu einer bestimmten Klasse gehört oder nicht. Dies führte zu vielversprechenden Ergebnissen und bietet eine Grundlage für zukünftige Verbesserungen.

Abstract

Lighting has a major impact on the mood of an indoor scene. This plays an important role in interior design. The way a room is lit affects the people in it. While a human can understand this concept and express how a scene feels to them, a computer cannot provide the same type of analysis at the moment. To provide a solution for this, this thesis seeks to implement a convolutional neural network to employ deep learning and classify scenes by their lighting mood. To this extent I created a data set consisting of images representing different lighting moods. The deep learning technique I used is transfer learning in which an already pre-trained network, in this case VGG16, is used. At first I tried a multi-class approach in which all classes are evaluated by one classifier, but later I moved to a binary classifier structure in which the classifier only learns and predicts if an image belongs to one given class or not. This lead to promising results and provides ground to improve upon in future work.

Contents

CHAPTER 1

Introduction

Lighting plays an important role in how humans perceive a scene, as the mood is greatly influenced by the lighting. A dimly lit room provokes different emotions than a room brightly lit by natural light. While this comes naturally to humans, there is currently no way for computers to perform this type of analysis. Being able to do this has many use cases, such as aiding in interior design by judging the lighting of a room, helping to create certain moods in scenes for photography as well as filming, or assist to create environments with the appropriate lighting with respect to specific tasks.

The goal of this thesis is to find a way, using Deep Learning, to automatically classify the mood of an image based on its lighting. This comes with many different challenges, such as obtaining a valid data set for such a task or how different lighting moods are generally described and classified by humans. To this extent, this thesis will present three different avenues for dataset generation: image retrieval from specifically crafted internet queries, AI image generation from text prompts, and manual labeling of images using crowd sourcing.

The other main challenge is the Deep Learning algorithm itself. Image classification is one of the classic and most used machine learning applications, hence, there are extensive resources on this topic and the actual technical implementation of the classifier is well defined if an appropriate data set is provided. The other challenge is finding the right class structure for the classifier. In the process of working on the thesis I tried both a general classifier, which classifies an image into one of the categories present in the data set, and binary classifiers, which decide whether an image belongs to a certain class or not.

This thesis will firstly provide a general overview of machine learning including image classification, deep learning, and convolutional neural networks, to provide the theoretical background for the technical part. Then I will describe the creation of the data set as well as the actual implementation of the algorithm and finally discuss the results.

I implemented a classifier using the pre-trained convolutional neural network VGG16 to determine the lighting mood of indoor scenes. These classifiers are binary and determine if an image belongs to one specific lighting mood or not. Additional to the implementation of the classifiers, I also collected the images and labeled them for the data set which is used for the classifiers. This paper shows that convolutional neural networks are capable of learning features to differentiate these scenes and achieve good results.

CHAPTER 2

Machine Learning Basics

John McCarthy first coined the term "Artificial Intelligence" in 1956 as a new domain in Computer Science that focuses on machines that mimic human cognitive functions. Three years later, in 1959, the term "Machine Learning" was introduced by Arthur Samuel, who described it as "a field of study that gives computers the ability to learn without being explicitly programmed". The phrase "Deep Learning" was established by Geoffrey Hinton in 2006 and describes a sub field of machine learning, which uses neural networks to imitate human-like decision making. [SAS19]

Sodhi et al. [SAS19] describe machine learning as the process of automating the automation, instead of just automating in the case of conventional programming. While conventional algorithms take input data and a program to produce some output, a machine learning algorithm intakes Data and Output, such as labels in the case of a classification problem, and then produces a program. This enables problem solving for cases in which a mathematical principle is hard to pin down by developing its own algorithm whenever data changes. This is done by repeatedly recognizing patterns and evolving the algorithm.

According to Sodhi et al. [SAS19], the types of problems being solved by machine learning can be divided in 5 categories: Classification, regression, similarity/anomaly, ranking, and sequence prediction. On top of that, there are 3 different kinds of machine learning algorithms: Supervised learning, unsupervised learning, and reinforcement learning. [SAS19] The problem this paper tries to solve is a classification problem, as I want to classify images on basis of their lighting and I did this through a supervised learning algorithm, which is discussed in Section 2.2.

Machine learning algorithms are often written in Python, which is what i used for this thesis as well. Due to its popularity for machine learning tasks, Python provides an abundance of libraries to help with this and has extensive resources available on machine learning. Sodhi et al. [SAS19] see the reason for Python's popularity in its simple and readable high level architecture, which enables it to be easily learned by non programmers. Though it still has a huge professional user base that provides ample documentation and libraries, also in part due to it being open source. While other languages like Java or C also have all the necessary matrix mathematics libraries, coding the same operations takes significantly more knowledge and effort compared to Python.

2.1 Deep Learning

According to LeCun et al. [LBH15], deep learning is an evolution from normal machine learning that can learn much more complex functions through its layered design. While conventional machine learning struggled to process natural data in its raw form, needing heavy engineering to get data in the right shape for it to be processed, deep learning takes the raw data and transforms it over multiple layers.

This is best explained by going through an example for image recognition. Images are represented as arrays of pixel values and in the initial layer the learned features represent edges by their presence, absence, orientation, and position. The second layer identifies patterns formed by these edges, even if they're slightly shifted. The third layer combines these patterns to recognize parts of known objects and subsequent layers detect complete objects by combining these parts. The crucial point in deep learning is that these feature layers are not handcrafted by humans, but instead acquired from data through a general-purpose learning procedure. [LBH15]

Deep learning has surpassed other machine learning techniques not only in image recognition, which is the focus of this paper, but also in speech recognition, in predicting the activity of potential drug molecules, in reconstructing brain circuits, in analysing particle accelerator data, in natural language understanding, in particularly topic classification, in sentiment analysis, in question answering, in language translation, and in predicting the effects of mutations in non-coding DNA on gene expression and disease. This is only part of its potential that is heightened because of the little manual engineering it requires. [LBH15]

2.2 Supervised Learning

Supervised learning is, as mentioned in section 2, one type of machine learning algorithm, the others being unsupervised learning and reinforcement learning. The type of algorithm is independent of the method used being either conventional machine learning or deep learning. According to Sodhi et al. [SAS19], a reinforcement learning algorithm takes a trial and error approach, by learning from each try and applying what it learned in future tries. Self-driving cars are an example of reinforcement algorithms in practice. Unsupervised learning does not take any labels with the input data. Instead it tries to find patterns without previously being trained on them. This is used on clustering problems for example.

This paper deals with a classification problem for which supervised learning is the right choice, as I have a data set with a labeled ground truth from which the algorithm should learn. Supervised learning receives the labeled data and trains on it until it reaches a desired level of accuracy. [SAS19] For this, a function that calculates the error is needed. According to this, error parameters then need to be adjusted to try and achieve a better result on the next run through. These parameters are most often called "weights". They define the input–output function of the algorithm. To find the right weights most procedures use stochastic gradient descent (SDG) [B+91], as do I in this paper by using the Adam optimization algorithm. This will be discussed further in section 2.5. This process involves presenting input vectors for a few examples, computing their outputs and errors, averaging the gradients, and adjusting the weights. This process is repeated with small example sets from the training data until the average objective function stops decreasing. It is called "stochastic" because these small sets provide noisy estimates of the overall gradient. This method often finds effective weights quickly, outperforming more complex optimization techniques. [LBH15]

When training is finished, the best weights that were found are used to classify another set of images. This is the test, which is used to test how the machine actually performs. It is important that the machine has not seen these test data while training as it allows estimating its ability to generalize arbitrary data. [LBH15] This test data accuracy will be used later to discuss the effectiveness of the classifiers trained for this paper in chapter 6.

LeCun et al. [LBH15] write that linear or any "shallow" classifiers have a problem extracting information out of raw pixel data and need a good feature extractor, which, as discussed before, requires a lot of additional engineering work. This problem is avoided by using deep learning algorithms. A deep learning architecture is comprised of multiple layers, the majority of which undergo learning. Many of them perform non-linear transformations from input to output. Each module's purpose in the stack is to enhance the representation's selectivity and invariance. With several non-linear layers, typically ranging from 5 to 20, such a system can execute highly complex functions on its inputs. These functions are capable of discerning fine details while disregarding substantial irrelevant variations like background, pose, lighting, and surrounding objects.

2.3 Convolutional Neural Networks

Deep learning works with neural networks, which, as the name suggests, try to imitate a biological brain in some way. One such kind of neural network that is often used for deep learning are artificial neural networks (ANNs). Using advanced learning algorithms, they make changes as they receive new input. Interestingly, they send information only forward into neurons from deeper layers. The output of an ANN is calibrated during training by using the information of an output error. An ANN learns how to reduce the chances of error, increasing efficiency over time. However, such networks are not suited for image classification because they require explicitly devised set data points. Moreover, converting 2D images to 1D Vectors is very demanding both in storage space as well as in processing power. [MA21]

Mascarenhas and Agarwal [MA21] write that this problem is solved by convolutional neural networks (CNNs). While an ANN deploys neurons or weights, a CNN uses filters to create feature maps the image, therefore producing a vector output. CNNs detect images by identifying and comparing patterns. Contrary to an ANN, which pushes the data forward, as described, in a CNN the same data is run multiple times through filters. This is done to create a feature map. It can extract features from images by itself which makes it much more suitable for image classification.

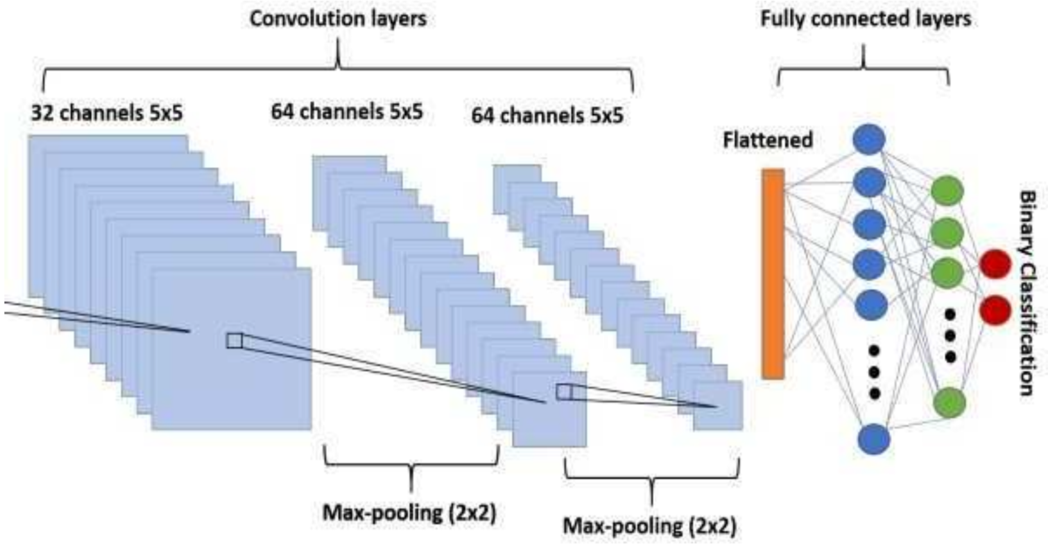


Figure 2.1: Visualization of a convolutional neural network. [MA21]

Going into more detail, the initial convolutional layer extracts image features by moving filters over the image. This is what generates feature maps of positive and negative pixel values. After this, the rectified linear activation function (ReLU) layer enhances the network's non-linear learning by receiving the output of the previous layer and outputting the positive input values, while outputting zero for negative values. Following this, the pooling layer reduces feature map sizes, therefore reducing computational load and accelerating the process. This is achieved through operations like max pooling and average pooling. [MA21]

Subsequently, the process continues with a sequence of convolution, ReLU, and pooling layers, further reducing input size. The final layer is the fully connected layer. It appears towards the end and uses activation functions like softmax or logistic to assign probabilities to various features detected in the image. The higher the probability, the likelier is the presence of the required feature. So this final stage represents the end of the extraction and learning process of the network. [MA21]

2.4 VGG16

There are many different CNNs to use for image classification. The first step was deciding to use a pre-trained network, a CNN that is pre-built, with the lower layers are already trained on an image set so that only the top layers need to be trained with your data set. This is called transfer learning. These networks are especially useful when designing and training a network from scratch is out of a project's scope or when the data set is too small for a full training. [KBKT17] Since my data set was created specifically for this task and, as a result is relatively small, further discussed in chapter 4, it was an obvious choice to use a pre-trained CNN.

The network I decided on is VGG16. A pre-trained convolutional neural network built on the VGGNet architecture, which was first proposed by Simonyan and Zisserman. [SZ15] Its main achievement is increasing the depth of the network but using very small convolutional layers which lead to great success in localisation as well as classification tasks.

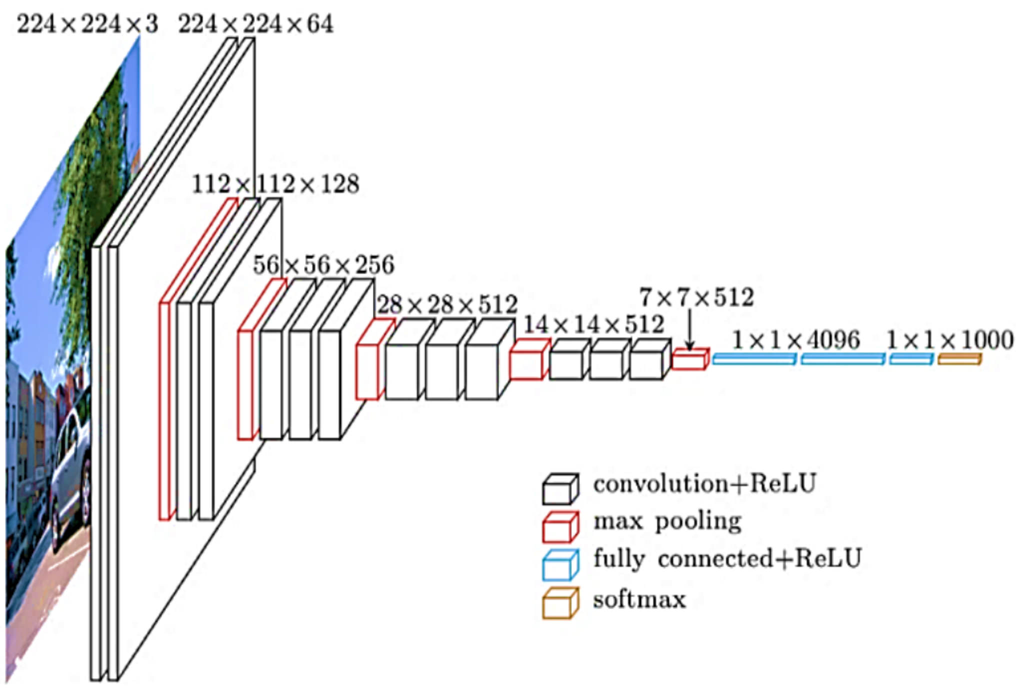


Figure 2.2: Visualization of the VGG16 Architecture. [KPB18]

The increase in layers is due to the amount of convolutional layers. The input RGB image, which has a fixed size of 224 x 224, is first passed through a stack of convolutional layers with a 3 x 3 filter, which is the smallest filter to capture the concept of left/right, up/down, center. Some, but not all of these layers are followed by Max-Pooling layers which use a 2 x 2 window. Another stack of convolutional layers is followed by the

Fully Connected layers. There are three such layers, with the first two each having 4096 channels, while the final one, which performs a 1000-way ILSVRC classification, contains one channel for each class. Therefore 1000 channels in total. This is the soft-max layer. The hidden intermediate layers are all equipped with ReLU. [SZ15] This architecture for VGG16 specifically is visualized in Figure 2.2.

The VGGNet architecture was created as a submission to the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) proposed by Russakovsky et al [RDS+15]. It was created to test different large-scale image classification systems and compare them. In 2014 Simonyan and Zisserman reached first and second places in localisation and classification tracks respectively. [SZ15] I will use the weights from the ImageNet data set for my transfer learning approach later in my implementation.

2.5 Optimizers

Optimizers are algorithms used to find an optimal solution for a problem. In the case of machine learning, they adjust the attributes of a neural network, typically its weights and biases, during the training process. Bennett et al. [BPH06] write that most machine learning problems can be reduced to optimization problems. The primary goal of optimization algorithms is to minimize the error or loss function so that the network can learn and improve its performance. They work by iteratively updating the parameters of the CNN based on the computed gradients of the loss function. These gradients indicate the direction and magnitude of changes needed to reduce the error. The optimizer adjusts the weights and biases by following these gradients to progressively minimize the loss.

There are plenty of different optimization algorithms used in deep learning, but I chose to use Adam for this thesis. The Adam optimizer was introduced in 2015 by Diederik P. Kingma and Jimmy Ba. [KB17] It combines the advantages of two other popular optimization methods, RMSprop [Tie12] and AdaGrad [DHS11], to address challenges like noisy gradients and sparse data in deep neural network training. Adam computes adaptive learning rates for each parameter by maintaining per-parameter adaptive learning rates and momentum-like terms. It efficiently scales the learning rates during training, providing faster convergence and better handling of non-stationary objectives compared to traditional stochastic gradient descent methods.

CHAPTER 3

Related Work

In this section I will look at some previous work that is of interest regarding this topic. I will provide a short overview for each paper and expand on how it relates to my work.

3.1 Illumination

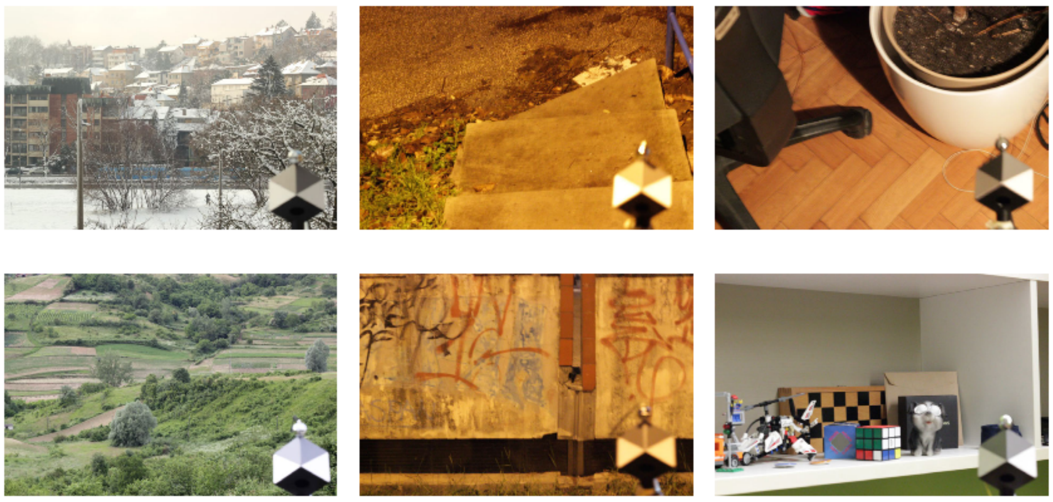


Figure 3.1: Example Images: Natural outdoor light source (left column), artificial outdoor light source (middle column), and artificial indoor light source (right column). [KSL20]

In 2020, Koščević et al. [KSL20] used deep learning to both classify the type of lighting in an image and estimate its illumination vectors, which are three-component vectors with one value for each color channel, that are then used to calculate the illumination estimation. To do this, they first categorize each image in one of three categories: Outdoor scene with natural lighting, outdoor scene with artificial lighting or indoor scene with artificial lighting, as seen in Figure 3.1. They used the convolutional neural network VGG16 for this purpose. On each of these categories they again use the pre-trained VGG16, but this time as a feature extractor, for the illumination estimation. They come to the conclusion that the classification into different light sources and running individual lighting estimation networks on each proved to be an improvement for the lighting estimation compared to only using one lighting estimation network. The relevant part for me is their demonstration that image classification using VGG16 in regards to lighting differences is possible. Though, contrary to my approach, they are focusing on clearly defined lighting scenarios such as natural lighting and not on mood, which is a more human way of perceiving lighting, and therefore not as clearly defined.

In their paper "Lighting (In)Consistency of Paint by Text" Farid [Far22] discusses the ability of DALL · E-2 to create images with consistent lighting. This is interesting since AI images do not base themselves on geometric or physical models. That means, the AI cannot calculate were the shadow should be according to the geometric and physical properties of the scene. They tested lighting consistency compared to real life scenes as well as lighting consistency within a scene.

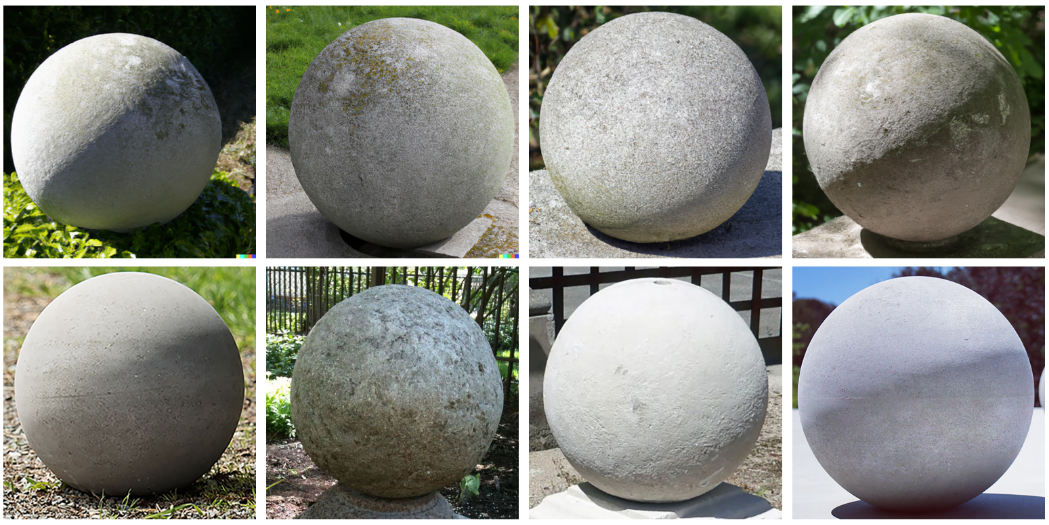


Figure 3.2: Top row shows synthesized images of garden spheres while the bottom row shows photographed real life counterparts used to evaluate lighting consistency. [Far22]

To compare the lighting consistency to real life they used images as shown in Figure 3.2. They show that in general the synthesized and photographed images correlate well, thus showing remarkable lighting consistency. When comparing lighting consistency within a scene, they show that it is lower than between the synthesized and the photographed image. This lead Farid [Far22] to the conclusion that these models achieve a highly realistic lighting, especially globally compared to photographed images, but less so within the image itself. This was especially interesting to me as I had considered creating my data set using such tools as I will discuss later in chapter 4.

3.2 Image Classification

2019 Swasono et. al. [STF19] used transfer learning with VGG16 to classify tobacco leaf pests. Some of these pests do not show striking differences in coloration; so, feature extraction can be difficult. Here CNNs have a big advantage. They tried both a SGD optimizer and the Adam optimizer, which is discussed in section 2.5. They show that VGG16 achieves great results on this task and that Adam outperforms the SGD optimizer.

Another interesting example of work in the field of image classification was done by Othman et. al. [OR19], who used CNNs to classify images by the type of room. Their goal was to have this classification help robots recognize a room in real time. I will focus on the image classification part. Just like my work, they classify indoor scenes. However, unlike in my case, the lighting does not play a role in the classification instead the focus is on in the content of room. For this, they tried VGG16, VGG19, and Inception V3 [SVI+16]. They tried two different approaches, a multiclass classifier and a binary classifier. They found that the binary classifiers performed better than the multiclass one. I came to the same conclusion in this paper, which will be discussed later on.

These two papers and Koščević et al. [KSL20] show the capabilities of VGG16 in image classification using transfer learning. Furthermore, they all demonstrate how these networks can handle some of the challenges of my approach. Othman et al. [OR19] show classification of images of indoor scenes, which my data set will consist of. Koščević et al. [KSL20] implement a classifier that focuses on illumination of images, which will be the task of my classifier, and Swasono et. al. [STF19] demonstrate the capabilities of CNNs when it comes to extracting features that are not very striking. Concerning lighting mood this can be important as differences between some moods might be difficult to define, as I will show later.

CHAPTER 4

The Data Set

A data set to train our network on is necessary. Unfortunately, as there is no previous work done with a mood-based lighting data set, such a data set was not available anywhere, which is why I created one myself. In this process, I went through multiple different iterations and approaches not only on how to create this data set but also on how exactly it should look like. The following sections will show the steps I used until settling on my final data set.

4.1 AI generated lighting images

As mentioned in Chapter 3, I thought about creating the data set with AI paint-by-text tools, specifically StableDiffusion, since at the time, it was gaining a lot of traction and it was fairly easy to run locally on private PCs. Unfortunately, at the time this AI did not provide the results I expected for depicting lighting mood in images. An example of this is shown in Figure 4.1, which is supposed to be a gloomily lit room.

This might be because the prompts used were not right. Maybe with enough trial and error it would have been possible to find a working prompt, but this would be an arduous process for each type of lighting mood decided upon and perhaps one prompt would not provide enough variety, increasing the effort even further. Add this to the long time even generating one image takes, at roughly 20 seconds, and the whole process would have taken an unreasonable amount of time for the scope of this thesis, even if the results would have been good. So the decision was made to use images collected from the internet to create the data set.

4.2 Web-based image retrieval

The decision to use images from the internet using a web-scraper was made for multiple reasons. Firstly, as just discussed, AI image generation was a rather slow process while

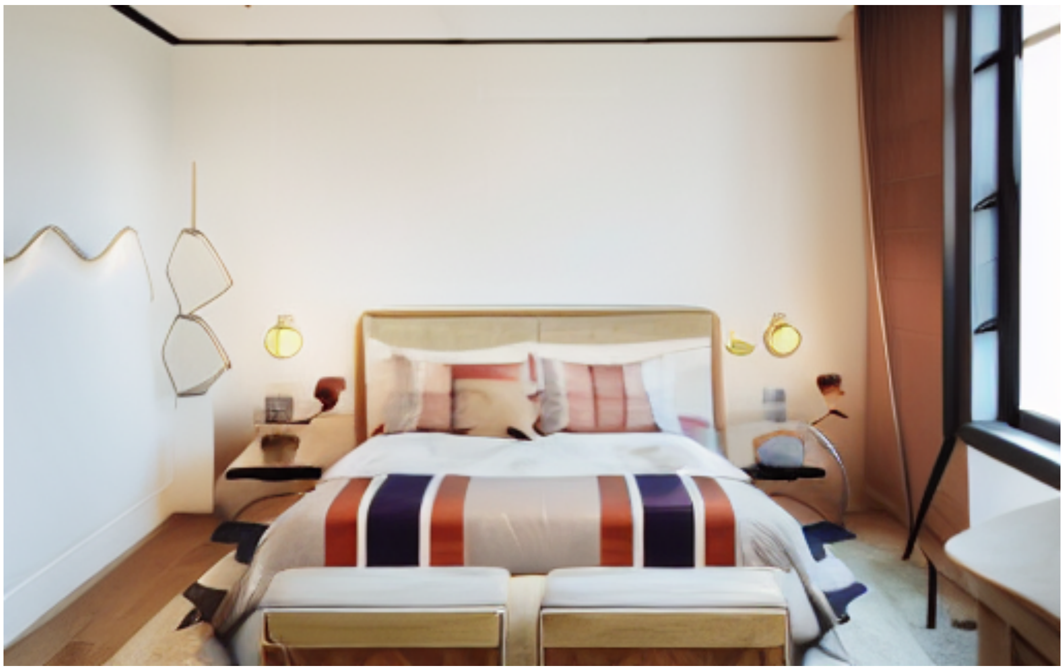


Figure 4.1: AI image with the prompt: "photo of a room with gloomy lighting"

scraping the internet for images should be quick. Secondly, when checking if search engines provided the images one would expect when asking for certain lighting moods the results were encouraging unlike with the AI image generation. Unfortunately, Google, which provided the best results, implements counter-measures to web scraping. Bing provided results, which did not meet expactions, but DuckDuckGo's results, while worse than Google's, were acceptable and, unlike Google, DuckDuckGo does not have any anti web-scraping measures.

Having found a way to get the images using a web-scraper for DuckDuckGo, I needed to decide on the different lighting moods for the categories. The first iteration was: "Gloomy", "Vibrant", "Cozy" and "Futuristic". These four categories were chosen because they seemed relatively easy to define at first and are able to be significantly differentiated from each other. This first set of lighting mood categories was meant to be adapted while working on the creation of this data set.

I used different search terms like "Gloomy lighting" or "Gloomy room lighting" to get a good amount of results. I did this because, while each different search term for each category gave some overlap in results, it also gave some unique results that were helpful in broadening the data set. The results then had to be manually searched through to remove any images that did not fit. This provided a pretty small data set that I then doubled in size by mirroring each image. This yielded 150 images for the smallest class and 176 images for the largest. This is called data set augmentation. [PW17]

Later on, I also searched manually through images on Google, and created a small test data set with 20 images for each class. After testing this data set with my implementation of the VGG16 network, which I will discuss in greater detail in chapter 5, it was clear that this data set was not big enough. The best prediction accuracy the network achieved on the test data set, after learning from the training data set, was just below 70%. To get a bigger data set, the decision was made to get a more general array of images and have a survey to categorize them.

4.3 The survey

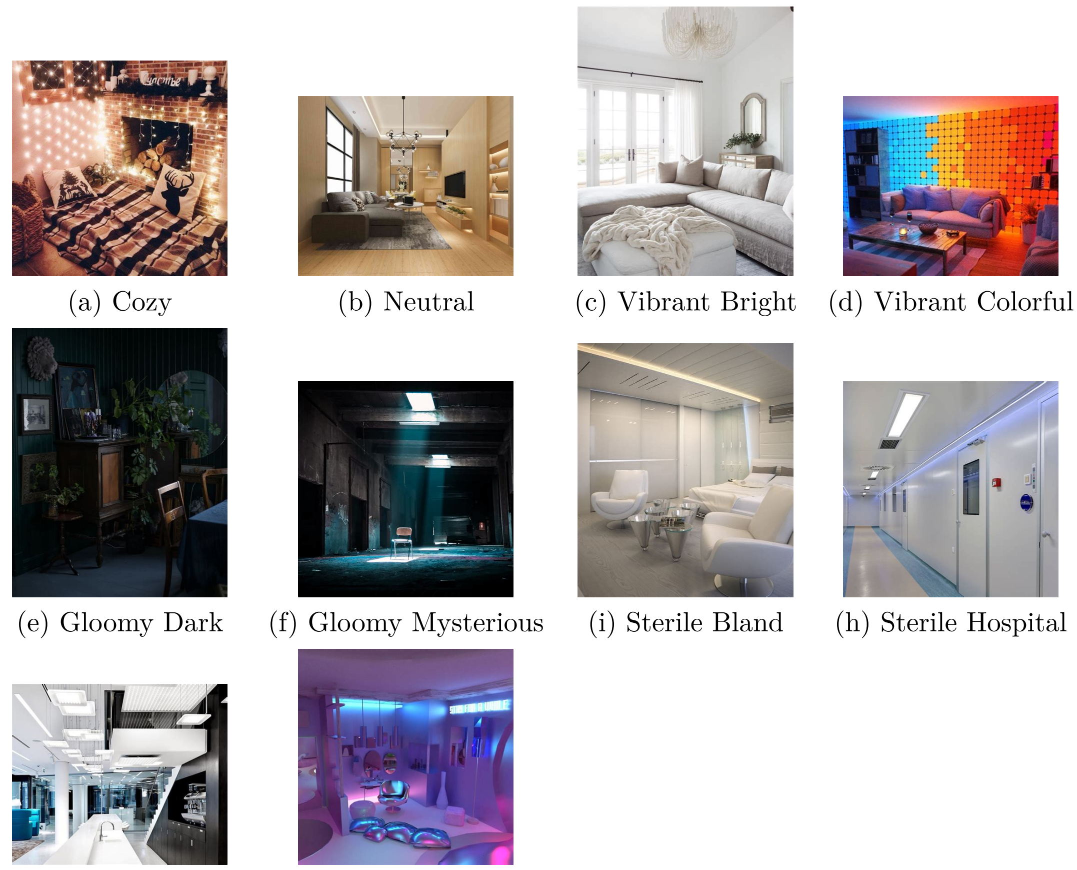
To get a bigger data set, I collected a greater amount of images, again through web-scraping, and then, instead of manually going through these images myself, crowd-sourced this process by conducting a survey. This survey provided participants with the images and asked them to pick a mood category for each.

Again, the first step was deciding on the categories. More categories were able to be added since I did not have to go through all the images alone. The additional categories were "Sterile" and "Neutral". "Neutral" was added because when going through the first iteration of the data set I noticed a lot of images that I could not decide on a category for. "Sterile" was added as some of the results for futuristic scenes reminded me of a "Sterile Hospital" room.

For the survey Amazon Mechanical Turk was used. This service crowd sources a distributed workforce which can be employed through tasks defined on the Amazon Mechanical Turk website. A set of tasks , which in our case was to classify images, is created by the user. Furthermore an amount paid per task has to be set. Amazon Mechanical Turk workers can then view this task and participate in it if they are interested. It is often used for similar data set collection tasks such as ours. [AmT]

The defined task gave each worker a set amount of images to categorize. No guidance or examples was provided, since I did not want to influence the worker's perception of different lighting configurations. However, in doing so, I might have created a task where answers were too open to interpretation and confused the users.

The results soon proofed to be unusable. Almost no images had a clear consensus on the lighting mood and there was no consistency or pattern visible on how images were categorized in the survey. Only around 34% of images had a majority consensus and only around 0.5% were unanimous. Therefore, it seemed almost random. There can be a lot of reasons for this: One could be that there were no clear rules or any guidance on what each category should encapsulate, as just mentioned. Another could be the possible variety of backgrounds of Amazon Mechanical Turk workers. This could have an effect on how they perceive these images. It could also be that the task was just not done faithfully. Whatever the reason, this was not the solution for the data set problem.



(a) Cozy (b) Neutral (c) Vibrant Bright (d) Vibrant Colorful

(e) Gloomy Dark (f) Gloomy Mysterious (i) Sterile Bland (h) Sterile Hospital

(i) Futuristic Other (j) Futuristic Neon

Figure 4.2: Examples for each image category

4.4 Final categories and retrieval process

It was finally decided to categorize all the images gathered so far manually by more strictly defined categories. The categories were: "Cozy", "Vibrant Colorful", "Vibrant Bright", "Sterile Hospital", "Sterile Bland", "Gloomy Mysterious", "Gloomy Dark", "Futuristic Neon", "Futuristic Other", and "Neutral". The decision to split them up this way was made to make them more clearly definable while still keeping the option to merge them if some categories did not end up with enough images. I will now describe each category by the traits decided on.

"Cozy" is defined by having usually dim lighting and small light sources like fairy lights, candle lights or similar. "Gloomy Dark" is a dark room with limited lighting, while "Gloomy Mysterious" differs by having even less light, mostly only sparse natural light falling in and having rooms that appear desolate. "Vibrant Bright" are very brightly lit spaces with a lot of natural light, while "Vibrant Colorful" is defined by having "Vibrant Colorful" light sources and reflections present. "Futuristic Neon" is close to this but separates itself by having neon colors from unusual light sources. "Futuristic Other" drops the need for neon colors and focuses on unusual light sources and shapes. "Sterile Hospital" are images where the lights are mostly unnatural and white with rooms that are very sterile akin to hospital floors. "Sterile Bland" is also defined by the unnatural and sterile white lights but the room itself are normal lived in rooms in a house. So both have a general lack of colors. "Neutral" is the category for images that do not fit any of the descriptions and evoke no real mood. An example for each category is shown in Figure 4.2.

After I completed the categorization and did some tests. It was quite obvious that some of these categories were just too small and not different enough from each other to justify having them separate. So the decision was made to merge "Sterile Bland" with hospital to one class and also "Vibrant Colorful" with future neon since they shared a lot of characteristics and it was often difficult to decide to which of these categories an image should belong to. Finally both "Gloomy" categories were merged since again they shared a lot of similarities and keeping them separate would have interfered with the results.

So the final categories that were used for the final results were: "Cozy", "Sterile", "Futuristic", "Vibrant Colorful", "Vibrant Bright", "Gloomy", and "Neutral". All classes were then doubled in size by employing the same data-set augmentation mentioned in section 4.2.

CHAPTER 5

Implementation

In this chapter I will talk about my implementation of the VGG16 network , including details about the training phase. This includes different iterations of the settings and how I decided on the final settings.

As discussed in chapter 2, the problem that this paper attempts to solve is a classification problem. This means that the deep learning algorithm must learn different classes in the training step and then classify the test set according to these classes in the testing step. In this work, the classes are the lighting moods discussed earlier in chapter 4.

This classification problem is approached by a supervised learning method, where the algorithm learns from labeled data. This means that the data in the training set is already labeled and the algorithm tries to recognize patterns in these labels and learn them. For this reason, I implemented a version of the VGG16 network that takes the images labeled with their respective mood category and tries to learn from them. Since VGG16 is a pre-trained network, most of the layers are already trained on a large dataset, and only the top layers are trained in my implementation. I chose this approach because of the small dataset I am using. This method is called transfer learning.

The implementation is just a single Python script. All the necessary imports for the network and the training process are from keras and sklearn. Most notable of course are VGG16 itself, the optimizer, and StratifiedKFold. The latter two will be discussed later. This script contains all the data preparation and network configuration that needs to be done.

Algorithm 5.1: Pseudo code for model training and evaluation

```
Input  : Input Shape, Number of Classes, Optimizer, Fine-tune
Output : Compiled Model
Function create_model(input_shape, n_classes, optimizer='rmsprop', fine_tune=0):
    Load VGG16 model without top layers
    Freeze layers based on fine-tune parameter
    Add custom fully connected layers on top of VGG16 model
    Compile model with specified optimizer and loss function
    return Compiled Model
top_level_folder ← 'path to data set;
ids, labels ← [], []
count ← 1
for folder_name in os.listdir(top_level_folder) do
    for file_name in os.listdir(folder_path) do
        Extract id and label from file_path
        Append id and label to respective lists
    end
end
Split the data into train and test using StratifiedKFold with 5 splits
for each train-test split do
    Prepare ImageDataGenerators for training and testing images
    Configure generators for training and validation data
    Define input shape, optimizer, number of classes, and other parameters
    Train VGG16 model without fine-tuning using train_model()
    Fine-tune the VGG16 model using train_model()
end
Close all result files
```

5.1 Data Preparation

Each data set is stored in a folder which contains two subfolders: One for the class the binary classifier focuses on and one for the "Other" class consisting of images from all the other categories. The names of these subfolders are "Classname" and "Other" respectively. These names are then used as the labels for training and testing.

After the script opens one data set folder it converts it to a train and a test data set using KFold. A Kfold splits the data set in as many folds as given by a variable and selects a random selection of images as train and test data. This is done to get a more accurate assessment on the performance of the network by averaging multiple such folds together. It also removes the need to manually separate train and test data. For this model, I used StratifiedKfold, which is an improvement on the traditional kFold because in cases where one or some classes have more images than others, StratifiedKfold keeps the ratio the same between this classes. [Str]

The next steps are done for each fold. Firstly, the test and train data are saved in a dataframe and the ImageDataGenerator, which does preprocessing on the images to transform them in the correct format for the VGG16 network, is created. With this, the final datasets are then being created: One for training and one for validation, both generated out of the train and test dataset. In this step the labels are assigned.

5.2 Model Configuration

The next interesting part is the configuration of the optimizer. As discussed in Section 2.5, there is a wide selection of different optimizers to choose from and I decided on Adam. The learning rate, which determines how strong each optimization step is, changed a lot with experimentation. In the end I decided to try two different learning rates. At first 0.0001 and a lower one at 0.00001, since some tests worked better with each of them. At higher learning rates over fitting becomes a problem, especially with small data sets such as the one used here. Additionally, after 5 epochs without improvements in the validation loss, the learning rate is reduced by a factor of 0.2.

The model then gets created by calling the `create_model` function. As parameters the input shape, amount of classes, the optimizer, and the amount of fine tuning layers, which for the first run is zero, are given. Since I use transfer learning for this thesis, the pre-trained convolutional layers are loaded using the ImageNet weights as discussed in Section 2.4. Next the layers in the convolutional base are switched from trainable to non-trainable in order to freeze them. If any fine tuning layers are given, the given amount of layers will not be frozen. Then a new 'Top' of the model is created. This is 'Bootstrapping' a new top_model onto the pretrained layers. This layer is then grouped with the pretrained ones and compiled in order to create the final model.

5.3 Training

For the training maximum amount of epochs is set to 50. An early stop is also configured which stops the model if no improvements have been made in the last 15 epochs regarding the validation loss. Afterwards the best weights up to that point are being restored. The best weights are defined by having the lowest validation loss.

Finally, the training is started by calling the fit method on the just created model. This function takes all the configurations done earlier as parameters. After either the 50th epoch is over or it stopped early, the best weights are loaded into the model and the predict function is called with the test data as parameter. This then gives the final results that get printed out and written into the plain text file opened in the beginning of the script.

Then for the same fold all the same steps are taken again but this time when creating the model two layers are unfrozen. The learning rate is also lower by one decimal place. This is done in order to have as many different results with various settings to get the best results possible.

CHAPTER 6

Results

6.1 Deciding on the Final Approach

As discussed in section 4.4, the final categories are "Cozy", "Neutral", "Colorful", "Sterile", "Gloomy", "Futuristic", and "Bright". Running the algorithm with these categories led to very disappointing results. The accuracy was only around 50% but even that is only because the algorithm predicted almost every image as "Neutral" and this was the biggest category so, the accuracy percentage was skewed.

Therefore, in the second run, I reduced the number of the images in "Neutral" to around 400 to reduce the imbalance in the data set and bring it to a similar size to "Cozy", the second biggest class. This provided even worse results with 22% accuracy, due to the fact that the algorithm now predicted almost all test images to be "Cozy". This is probably due to "Cozy" now being the more unique one of the two biggest classes, which is easier for the classifier to learn.

After these disappointing results, I tested a few run-throughs with binary classification. For example, "Cozy" and "Sterile" which reached above 85% accuracy or "Bright" and "Gloomy" which even reached above 90%. Seeing that these binary classifiers worked so well, the decision was made to go with binary classifiers. In these binary classifiers, one category will be one class and the other class will be images from all the other categories combined. This also helps with the imbalance in the size of the data set since it is easy to adjust the size of the "Other" class in the classifier.

6.2 Classifier Results

As just discussed, the final models were binary classifiers each focusing one of our categories comparing them to a subset of all the other categories combined. Each of these classifiers was run with 4 different settings. As discussed above, this was done with two different learning rates and for each of these then once with and once without fine tuning the last 2 layers. For this discussion I will be talking about the best settings for each of these classifiers as discussed in the implementation subsection.

What is also to note is that for each of these binary classifiers the "Other" category that includes images from all the other classes is trimmed to roughly match the size of the given category in order to not have unbalanced class sizes which have a significant negative impact on the results.

The best settings for each of them are as follows:

"Cozy"	lower learning rate, without fine tuning
"Colorful"	standard learning rate, with fine tuning
"Gloomy"	lower learning rate, without fine tuning
"Sterile"	standard learning rate, without fine tuning
"Bright"	standard learning rate, with fine tuning
"Futuristic"	lower learning rate, without fine tuning

Table 6.1: Best settings

In Table 6.2 the results are shown to generally lie around the 80 mark. An outlier is the future class. I will come back to this later. First it is important to establish what each of these numbers mean. Each classifier has been run with multiple folds, as explained in chapter 5, and as final result the average of all folds is taken. I also show a breakdown of the results for each category individually. Most of the time the "Other" category's prediction scores are higher.

Most of these classifier land around 80% in accuracy with 2 notable exceptions, "Bright" and "Futuristic". I will now go into detail on both these categories in order to find possible reasons for those exceptions by running some additional tests on them. Furthermore, I will also take a look into the "Sterile" class to have a reference point of a well-performing class.

I also had one such binary classifier running for "Neutral". The results are shown in Table 6.3. They are unfortunately not very good, though this was to be expected in my binary classifier approach. Since the "Neutral" class is functioning as the category for all images that did not fit anything else and did not express any tangible mood it lies between all classes. Therefore I think comparing it to the "Other" class which is a collection of all the other categories is bound to lead to a bad result as neither class has much connecting cohesion internally and therefore the algorithm cannot separate them from each other. So I decided to see "Neutral" not as a category to classify like the others but just as the main part of the "Other" class in each of the binary classifiers.

Class	**Cozy**			**Colorful**		
Key	Cozy	Other	Overall	Colorful	Other	Overall
Fold 1	0.79	0.81	80.00	0.80	0.82	81.30
Fold 2	0.76	0.80	78.26	0.80	0.84	82.26
Fold 3	0.76	0.74	75.00	0.71	0.84	79.84
Fold 4	0.77	0.82	79.38	0.84	0.88	86.29
Fold 5	0.81	0.85	83.02			
Average	0.78	0.80	**79.13**	0.79	0.85	**82.42**

Class	**Gloomy**			**Sterile**		
Key	Gloomy	Other	Overall	Sterile	Other	Overall
Fold 1	0.80	0.80	79.81	0.67	0.80	75.00
Fold 2	0.83	0.82	82.69	0.82	0.85	83.64
Fold 3	0.65	0.67	66.02	0.89	0.90	89.29
Fold 4	0.84	0.80	82.52	0.78	0.85	82.14
Fold 5	0.85	0.83	84.16	0.73	0.82	78.18
Average	0.79	0.78	**79.04**	0.78	0.84	**81.65**

Class	**Bright**			**Futuristic**		
Key	Bright	Other	Overall	Futuristic	Other	Overall
Fold 1	0.77	0.80	78.95	0.75	0.83	80.00
Fold 2	0.70	0.73	71.88	0.56	0.67	61.90
Fold 3	0.80	0.76	77.89	0.67	0.67	66.67
Fold 4	0.76	0.71	73.40	0.71	0.78	75.00
Fold 5				0.50	0.79	70.00
Average	0.76	0.75	**75.53**	0.64	0.75	**70.71**

Table 6.2: Results of the binary classifiers

Class	**Neutral**		
Key	**Neutral**	**Other**	**Overall**
Fold 1	0.56	0.73	66.51
Fold 2	0.64	0.75	70.09
Fold 3	0.60	0.72	67.29
Fold 4	0.31	0.72	60.28
Fold 5	0.49	0.72	63.85
Average	0.52	0.73	**65.60**

Table 6.3: Results for "Neutral" binary classifier

6.3 Comparing Classes

In this section I will take a look at some specific classes and why the results came to be as they are. I will be mainly focusing on the categories that did not work that well and comparing them to one that did work well. For this purpose I will let some additional classifiers run to see what caused the issues and how it could be improved.

As with the major classifiers, if the category in question is smaller than the "Other" category, the latter will be trimmed to be the same size. So for example in the section focusing on "Futuristic" the class "Bright" will be trimmed to fit the small sample size of "Futuristic". In the opposite case nothing will be trimmed. This will lead to differing results.
In the following chapters (called "deep dives"), I will only be using the best settings of the major classifier in order to keep the results compact.

6.3.1 Bright deep dive

For this first deep dive on the "Bright" class, my suspicion was that the images in "Bright" and in "Neutral" are not different enough and a lot of times are overlapping. When categorizing I often had a hard time deciding between these two categories on a lot of images and even when going through the data set later on, I was often questioning my decisions.

In order to test this theory I ran binary classifier, with the best settings for "Bright", with binary classifiers between the "Bright" class and every other class in turn, to see if there is one particular class that "Bright" is not working well with. The results are shown in Table 6.4.

These tests showed that the classifier can distinguish between "Bright" and all other classes pretty well with one exception. As expected, the results for the "Bright" and "Neutral" classifier were significantly worse at only 60.6% accuracy. To check if this is the main issue influencing the all-categories-classifier, I created another "Bright" versus all other categories but this time without images from "Neutral". And as demonstrated in the results, in Table 6.5, that lead to a significant improvement to 80.74% accuracy.

This significant improvement now made me question if this is specifically because "Neutral" was removed or if removing any class would also improve the results. If this would be the case, I could conclude that increased diversity in the "Other" category leads to worse results. So I ran this one time with all categories but "Cozy" and once with all categories but "Colorful" and "Gloomy".

And, as seen in Table 6.6, that the results are significantly worse this way. This implies that reducing the variety in the "Other" category alone is not enough to prompt improved results. Removing the worst class, in this case "Neutral", improves results, as now the average image in "Other" is more different than before. To see if this holds up I will took a similar look into the "Futuristic" class.

Class	**Neutral**			**Colorful**		
Key	Bright	Neutral	Overall	Bright	Colorful	Overall
Fold 1	0.62	0.57	60.00	0.91	0.92	91.25
Fold 2	0.49	0.57	53.33	0.85	0.88	86.42
Fold 3	0.58	0.68	64.00	0.94	0.94	93.83
Fold 4	0.71	0.62	67.57	0.88	0.90	88.75
Fold 5	0.60	0.56	58.11	0.76	0.77	76.25
Average	0.60	0.60	**60.60**	0.87	0.88	**87.30**

Class	**Gloomy**			**Cozy**		
Key	Bright	Gloomy	Overall	Bright	Cozy	Overall
Fold 1	0.90	0.91	90.54	0.79	0.82	81.08
Fold 2	0.87	0.89	88.00	0.82	0.83	82.67
Fold 3	0.93	0.94	93.33	0.85	0.86	85.33
Fold 4	0.89	0.89	89.33	0.86	0.88	86.67
Fold 5	0.85	0.86	85.33	0.86	0.84	85.14
Average	0.89	0.90	**89.31**	0.84	0.85	**84.18**

Class	**Sterile**			**Future**		
Key	Bright	Sterile	Overall	Bright	Future	Overall
Fold 1	0.67	0.26	54.10	0.89	0.00	80.85
Fold 2	0.84	0.71	79.03	0.89	0.00	80.43
Fold 3	0.86	0.76	82.26	0.94	0.62	89.13
Fold 4	0.86	0.76	82.26	0.91	0.22	84.44
Fold 5	0.84	0.80	81.97	0.89	0.18	80.43
Average	0.81	0.66	**75.92**	0.90	0.20	**83.06**

Table 6.4: Results for "Bright" individual classifiers

Class	**Other without Neutral**		
Key	**Bright**	**Other**	**Overall**
Fold 1	0.85	0.84	84.42
Fold 2	0.88	0.87	87.18
Fold 3	0.84	0.82	83.12
Fold 4	0.70	0.69	69.74
Fold 5	0.79	0.79	79.22
Average	0.81	0.80	**80.74**

Table 6.5: Results for "Bright" when "Neutral" removed from the "Other" class

Class	Other wo. Gloomy & Colorful			Other without Cozy		
Key	Bright	Other	Overall	Bright	Other	Overall
Fold 1	0.67	0.70	68.42	0.70	0.54	64.00
Fold 2	0.72	0.68	69.74	0.65	0.67	65.79
Fold 3	0.63	0.68	65.33	0.63	0.68	65.79
Fold 4	0.68	0.74	71.05	0.66	0.68	67.11
Fold 5	0.53	0.65	60.00	0.76	0.76	76.00
Average	0.65	0.69	**66.91**	0.68	0.67	**67.74**

Table 6.6: Results for "Bright" when other classes were removed from the "Other" class

6.3.2 Futuristic deep dive

Class	Neutral			Colorful		
Key	Future	Neutral	Overall	Future	Colorful	Overall
Fold 1	0.71	0.80	76.19	0.63	0.75	70.00
Fold 2	0.38	0.62	52.38	0.50	0.80	71.43
Fold 3	0.78	0.82	80.00	0.67	0.81	76.19
Fold 4	0.50	0.79	70.00	0.74	0.76	75.00
Fold 5	0.62	0.81	75.00	0.50	0.67	60.00
Average	0.60	0.77	**70.71**	0.61	0.76	**70.52**

Class	Gloomy			Cozy		
Key	Future	Gloomy	Overall	Future	Cozy	Overall
Fold 1	0.53	0.74	66.67	0.80	0.89	85.71
Fold 2	0.42	0.52	47.62	0.80	0.89	85.71
Fold 3	0.80	0.88	85.00	0.88	0.92	90.00
Fold 4	0.46	0.74	65.00	0.86	0.92	90.00
Fold 5	0.67	0.80	75.00	0.80	0.80	80.00
Average	0.58	0.74	**67.86**	0.83	0.88	**86.28**

Class	Sterile			Bright		
Key	Future	Sterile	Overall	Future	Bright	Overall
Fold 1	0.63	0.77	71.43	0.82	0.78	80.00
Fold 2	0.75	0.85	80.95	0.75	0.85	80.95
Fold 3	0.35	0.52	45.00	0.75	0.85	80.95
Fold 4	0.50	0.67	60.00	0.62	0.75	70.00
Fold 5	0.67	0.86	80.00	0.80	0.88	85.00
Average	0.58	0.73	**67.48**	0.75	0.82	**79.38**

Table 6.7: Results for "Futuristic" individual classifiers

In this section I will go into detail on the "Futuristic" category as I did with the "Bright" category, again running the same one-on-one binary classifier with all classes as I did with the "Bright" class. The results are shown in Table 6.3.2.

As these results show, the outcome was in general significantly worse than "Bright" with almost all categories leading to a bad result. The two exception were "Cozy" and "Bright". This generally worse performance is probably due to its relatively small sample size, constituting the smallest class with only 47 images. Another indicator for that is the inconsistency between folds. This could suggest that the classifier is guessing a lot or that on some folds the images are worse for training or testing which is obviously more of an issue with a smaller data set. However, the good results with "Cozy" and "Bright" shows that the classifier can still work with a distinct enough category.

As with "Bright" I also did some tests with a combined category but removed the worst performing classes. I did one with removing only "Sterile" as the worst performing class and one with "Sterile" and "Gloomy" removed, as "Gloomy" is only slightly better than "Sterile". The results were as follows in Table 6.3.2.

Class	**Other without Sterile**			**Other wo. Sterile & Gloomy**		
Key	Future	Other	Overall	Future	Other	Overall
Fold 1	0.63	0.70	66.67	0.20	0.75	61.90
Fold 2	0.56	0.67	61.90	0.64	0.56	60.00
Fold 3	0.00	0.65	47.62	0.42	0.52	47.62
Fold 4	0.50	0.77	68.42	0.67	0.80	75.00
Fold 5	0.44	0.55	50.00	0.67	0.80	75.00
Average	0.43	0.67	**58.92**	0.52	0.69	**63.90**

Table 6.8: Results for "Futuristic" when other classes were removed from the "Other" class

This shows no improvement, further solidifying the theory that the small sample size is the biggest issue with future since removing the worst performing classes in the "Bright" deep dive lead to significant improvements.

6.3.3 Sterile deep dive

Finally I will also do a deep dive on the "Sterile" class. This is done to draw a comparison with one of the best performing classes and see the individual results there. This hopefully gives more insight into how individual results impact the overall performance.

Firstly, as with the other two deep dives, I will set up the one-on-one binary classifiers with all other classes. If this test is consistent with previous observations, the average results of these should be better than with the other two deep dives and therefore lead to a better overall result. Table 6.9 shows the results.

As Table 6.9 demonstrates, the results are pretty similar to "Bright", and even get a bit worse. Though, as talked about before, "Futuristic" being this bad is a direct consequence of the small data set size, which just puts a ceiling on the success rate with this class. It is also by far the smallest subset of the "Other" class so it will not have much impact

Class	**Neutral**			**Colorful**		
Key	Sterile	Neutral	Overall	Sterile	Colorful	Overall
Fold 1	0.83	0.89	86.27	0.94	0.94	94.00
Fold 2	0.71	0.76	74.00	0.91	0.89	89.80
Fold 3	0.47	0.72	63.27	1.00	1.00	100.00
Fold 4	0.77	0.79	78.00	0.79	0.65	73.47
Fold 5	0.65	0.79	74.00	0.86	0.85	85.71
Average	0.69	0.79	**75.11**	0.90	0.87	**88.60**

Class	**Gloomy**			**Cozy**		
Key	Sterile	Gloomy	Overall	Sterile	Cozy	Overall
Fold 1	0.92	0.91	91.84	0.88	0.88	88.00
Fold 2	0.89	0.90	89.80	0.98	0.98	97.92
Fold 3	0.98	0.98	97.96	0.68	0.15	53.06
Fold 4	0.88	0.88	87.76	0.88	0.91	89.80
Fold 5	0.88	0.87	87.76	0.96	0.96	95.92
Average	0.91	0.91	**91.02**	0.88	0.78	**84.94**

Class	**Bright**			**Future**		
Key	Sterile	Bright	Overall	Sterile	Future	Overall
Fold 1	0.80	0.79	79.59	0.82	0.47	72.73
Fold 2	0.74	0.84	80.00	0.76	0.48	66.67
Fold 3	0.81	0.82	81.63	0.84	0.67	78.12
Fold 4	0.40	0.67	57.14	0.22	0.43	34.38
Fold 5	0.76	0.79	77.55	0.57	0.48	53.12
Average	0.70	0.78	**75.18**	0.64	0.51	**61.00**

Table 6.9: Results for "Sterile" individual classifiers

on the end result. If "Futuristic" is removed from the equation, the results seem to be slightly better than the results for "Bright". It is still interesting to see that "Neutral" which represents the biggest part of the "Other" category is second worst. Just as with the other deep dives, I did some more tests by removing "Neutral" from the "Other" category to see if this improves the accuracy. The results are shown in Table 6.10.

Here it shows that it did not actually improve the accuracy but actually decreased it. A potential reason for that could be the increased impact of "Futuristic" and "Bright" which also both have subpar results with "Sterile". While the results with "Bright" are not that bad, it is visually and by our own definition the most similar class to "Sterile". To test this, I also made one classifier removing "Bright" and "Futuristic" additionally to "Neutral" from the "Other" class. The results for this are shown in Table 6.11

As this demonstrates, removing "Futuristic" had basically no impact on the results. As said before, this is likely due to the minuscule impact these images have since they only represent a small subsection of the data set. Removing "Bright" improves the results

Class	**Other without Neutral**		
Key	**Sterile**	**Other**	**Overall**
Fold 1	0.68	0.75	72.00
Fold 2	0.70	0.83	78.00
Fold 3	0.00	0.68	52.00
Fold 4	0.80	0.80	80.00
Fold 5	0.78	0.80	79.17
Average	0.59	0.77	**72.23**

Table 6.10: Results for "Sterile" when "Neutral" removed from the "Other" class

Class	**Other wo. Neutral & Future**			**Other wo. Neutral & Bright**		
Key	Sterile	Other	Overall	Sterile	Other	Overall
Fold 1	0.71	0.75	73.47	0.48	0.75	66.00
Fold 2	0.72	0.81	77.55	0.84	0.87	86.00
Fold 3	0.53	0.73	65.96	0.78	0.81	80.00
Fold 4	0.00	0.67	50.00	0.88	0.91	89.80
Fold 5	0.83	0.83	83.33	0.57	0.76	69.39
Average	0.56	0.76	**70.06**	0.71	0.82	**78.24**

Table 6.11: Results for "Sterile" when more classes were removed from the "Other" class

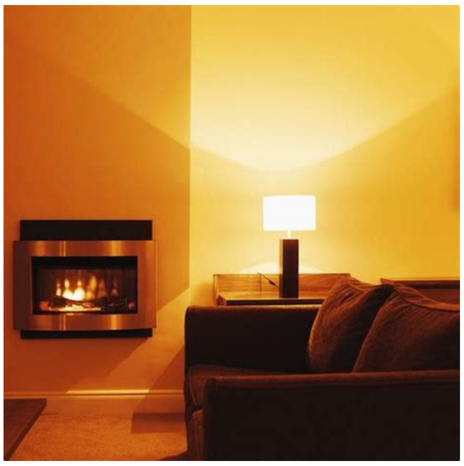
compared to only removing "Neutral" but it is still worse than the complete binary classifier and also more inconsistent. So while the one-on-one classifier with "Sterile" and "Bright" works decently well, when "Bright" is part of the complete binary classifier it seems to have a negative effect on the results. On the other hand "Neutral", which leads individually to the worst results, seems to help the major classifier.

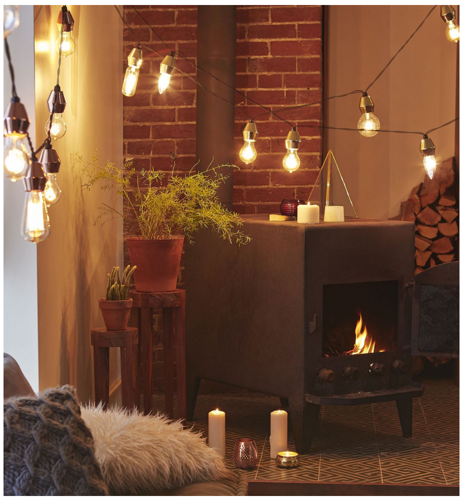
6.4 Interesting images

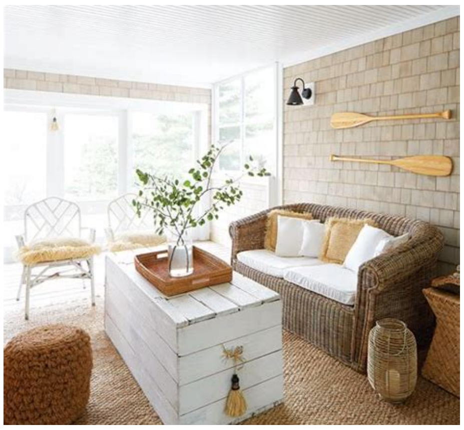
In this section I will present some images that represent the abilities of the classifiers. I will look at images that were correctly predicted, wrongly predicted and some that the classifiers were unsure about to see what could be the reason for that.

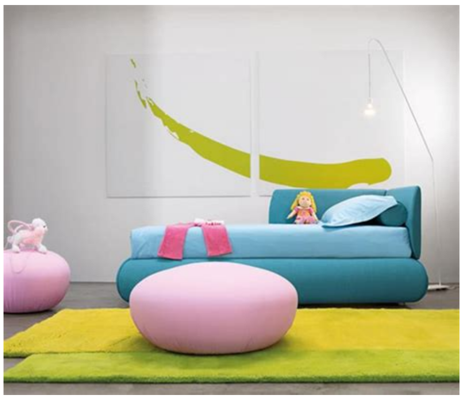
The image captions will each show the true class and the predicted class together with the confidence of the prediction. The prediction score will be rounded to a percentage with one decimal place. So in the case of The true class being "Other" and the prediction being correctly "Other" with a prediction score of 0.9923 the caption will look like this: *Truth: Other | Pred: Other [99.2%]*

The first images discussed here are taken from the "Colorful" classifier, which has the highest accuracy rating. I will give an example for each very confident prediction: One for correctly predicting "Colorful", one for correctly predicting "Other" and the corresponding incorrect predictions. Lastly, I will also look at one image that has a roughly 50/50 confidence.


(a) Truth: Colorful | Pred: Colorful [100%]


(b) Truth: Other | Pred: Colorful [100%]


(c) Truth: Other | Pred: Other [100%]


(d) Truth: Colorful | Pred: Other [100%]

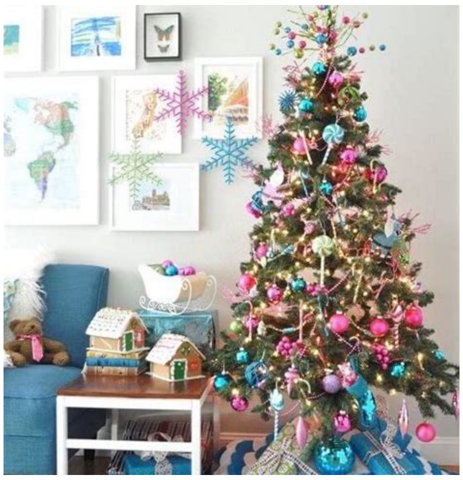

(e) Truth: Colorful | Pred: Colorful [54.4%]

Figure 6.1: Interesting images of the "Colorful" binary classifier

In Figure 6.1 (a) an image, that the algorithm correctly predicted to be "Colorful" with almost 100% confidence, is shown. The warm orange color is very noticeable. It could also almost fit in "Cozy" but it does not adhere to some of the ground rules set in Section 4.4 for that category. However, as the next image will demonstrate, this can lead to blurred lines between these classes.

Here in Figure 6.1 (b) an example for exactly that is given. With again almost 100% confidence, the classifier wrongly predicts this image, which belongs to the "Cozy" class and therefore is part of the "Other" category, as "Colorful". And it is easy to see why. Again there is a lot of warm light that is somewhat reminiscent of the image shown in Figure 6.1 (a). However, the one in Figure 6.1 (b) meets the criterion of the smaller, more abundant light sources, as described in Section 4.4. Both images also feature a fireplace.

The image in Figure 6.1 (c) was correctly predicted as "Other". This image originally belongs to the "Bright" class, being defined by the "Bright" natural light falling in. It is very clearly distinguishing itself from the "Colorful" class, as it is missing any strong color and is mainly white and brown, so it was easy to predict correctly for the algorithm.

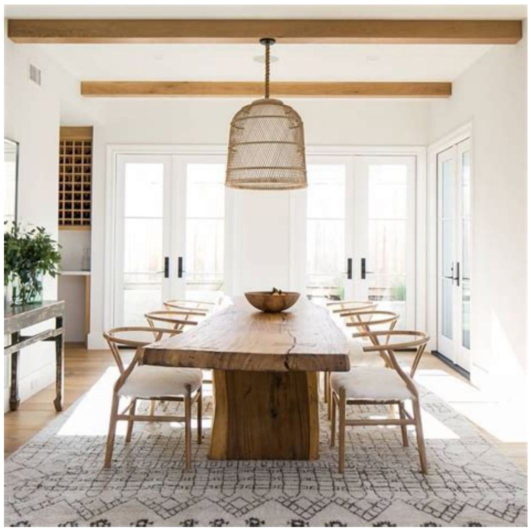
Now, Figure 6.1 (d) represents an image that I classified as "Colorful" but the algorithm predicted as "Other". The reason why I classified it as "Colorful" were the loud colors of the furniture in the bottom half of the image, as so much color did not fit with any other class. The algorithm though seems to see it as not enough color in the overall mood of the image.

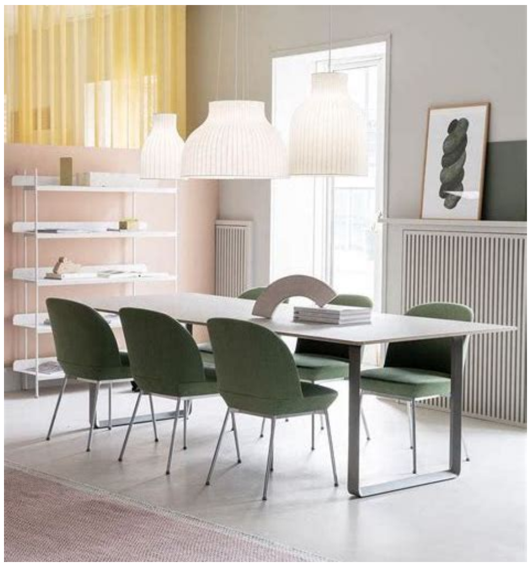
In Figure 6.1 (e) a similar effect can be seen. As before, a lot of the color is coming from objects in the scene rather than from the lighting itself. However, this time it seems the classifier decided it is "Colorful" even though it was very close. This might show that adding these type of images to "Colorful" could have been a mistake and it also could lead to the conclusion that the classifier actually classifies based on the lighting more than on the other content in the image.

I will now also look at a few images for the "Bright" classifier since it was one of the worst performing classifiers but has a decent data set size compared to the other classes. For that reason, it was more interesting to analyze than "Futuristic". As before the first image was correctly predicted, demonstrating what worked.

Figure 6.2 (a) is an image that was very confidently predicted correctly as "Bright". It hits all the markers I have set for the class "Bright" Natural lighting in Section 4.4. Now I will discuss the images that failed, to see if they have any significant difference or if anything else can be extracted from it.

The image in Figure 6.2 (b) is another one that would fit all the criteria for "Bright" as defined in Section 4.4. It is a bit darker than the first one and also has more color to it. These factors might make it harder for the algorithm to distinguish it from the "Other" category, since, as seen in Section 6.3.1, "Bright" is very close to "Neutral" and images like this are even closer.


(a) Truth: Bright | Pred: Bright [100%]


(b) Truth: Bright | Pred: Other [99.5%]

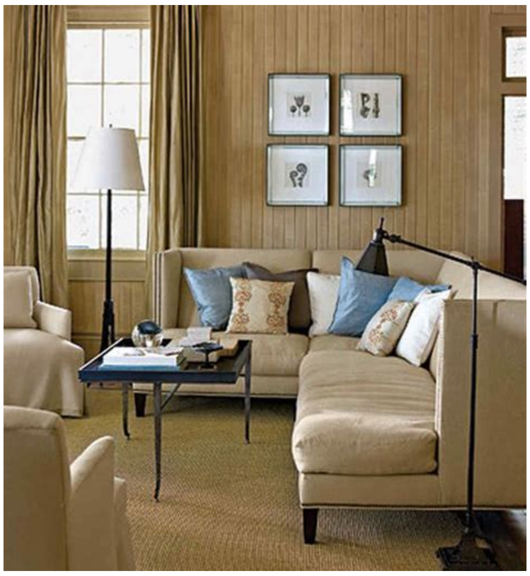
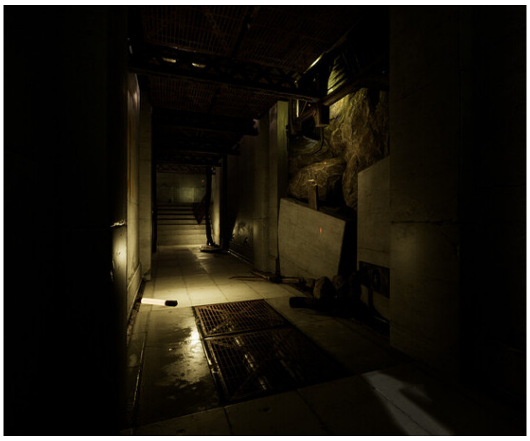

(c) Truth: Other | Pred: Bright [99.2%]

Figure 6.2: Interesting images of the "Bright" binary classifier

Now, looking at the other cases, Figure 6.2 (c) is an image that is part of "Other" and originally belongs to the "Neutral" class. It has been predicted as "Bright" and it does share some of the features with "Bright". It has natural light falling in, though one can see it is significantly darker than the two previous images that should be "Bright". This again shows how "Neutral" and "Bright" are just very difficult to distinguish.


(a) "Gloomy" image compared over all classifiers.

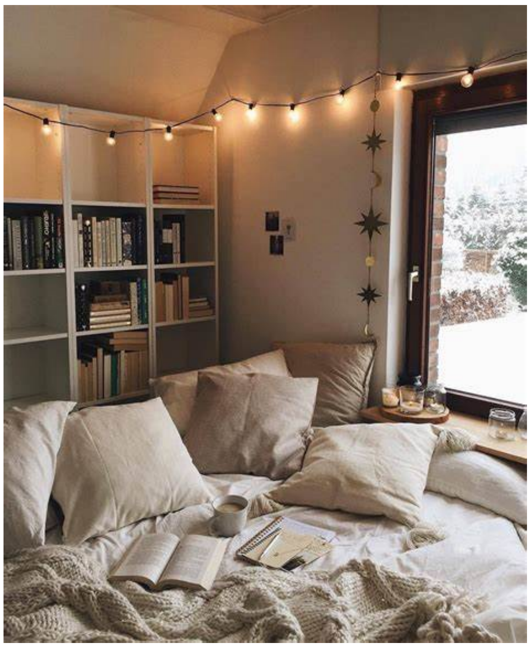

(b) "Cozy" image compared over all classifiers.

Figure 6.3: Interesting images being compared over different classifiers

Here I will take a look at one image and how it gets classified over multiple classifiers. An image of the "Gloomy" category is shown in Figure 6.3 (a). It has dark sparse lighting and fits perfectly in the "Gloomy" category. And when looking at the "Gloomy" classifier when the image was selected by the kFold algorithm for the test set (this is something to keep in mind as an image has to be randomly selected by the kFold algorithm to be part of the test set) performed very well. Once with a confidence of 94.7%, and another time the mirrored version of the image achieved 99,8% confidence. So both times the image achieved a very high confidence. Looking at the other classifier where this image is part of the "Other" class and in the test set, it mostly predicted correctly, with one exception. In the "Futuristic" classifier, the algorithm was confidently incorrect reaching a confidence of 99.8%. As discussed in previous sections, "Futuristic", due to its small data set, is very unreliable. It is likely that in the bad fold the random selection for the "Futuristic" train set included more images which were quite dark and the train set for "Other" contained more images that appeared brighter, leading to darker images being classified as "Futuristic". So putting this already established difficult classifier aside, this is as expected from an image that works very well with its own class but also differs enough from all other classes to work well as part of the "Other" category.

After this positive example, the following image did not work out so well. This "Cozy" image seen in Figure 6.3 (b) got incorrectly classified by the "Cozy" classifier. It meets the requirements of "Cozy" by being more dimly lit by multiple small light sources. The confidence in the "Cozy" classifiers was 93% in favor of "Other". The same image in the "Gloomy" classifier was also incorrectly classified this time as "Gloomy" instead of "Other" with a confidence of 99.8%. As a consequence one can assume that this image, despite fitting the defined standards of "Cozy" better than "Gloomy", is in the eyes of the algorithm closer to the images of the "Gloomy" category. This could be because the fold for "Gloomy" happened to consist of generally less dark images or more images taken in dark rooms with natural light coming in, like in this cozy image. Also, since the classifier was not trained on "Cozy" and therefore not trained to recognize the fairy lights as cozy lighting, the general darker atmosphere of the room fit better with "Gloomy" than with "Other," which is also filled with a lot of brighter images.

CHAPTER 7

Conclusion

The final binary classifiers I created for this thesis showed really promising results. Achieving around 80% for most classes and even 70% for the worst, demonstrates that the algorithm has learned most of the features that make up the lighting in the scenes. This is especially encouraging as creating the data set turned out to be a significant challenge and only resulted in a quite small data set. However, this showed that, even though the data set is small, transfer learning can provide impressive results.

The amount of images in a class definitely proved to have an impact on the result as the worst performing class "Futuristic" is also the smallest data set. Though this alone is not a deciding factor as the "Bright" class which had a moderate amount of images, more than the second best performing class "Sterile", had the second worst performance at 75,5% accuracy. How much the class differentiates itself from all others seemed to be also a deciding factor.

This paper also shows how difficult it is to classify based on these lighting moods, not only for deep learning algorithms but almost even more so for humans. The survey carried out for this paper through Amazon Mechanical Turk provided no meaningful results. Surveying the Mechanical Turk workers on lighting mood showed great disagreement between them, with most images not reaching a consensus. But even deciding on detailed characteristics for classifying manually proved to be difficult and even after having set characteristics for each class they were still difficult to be applied to some of the images. Of course a lot of this stems from lighting mood being a very subjective human concept. This resulted in some classes just not showing enough uniqueness to be easily told apart, which also impacted the classifiers. This is seen in the deep dive sections and also when looking at the individual images. There it is demonstrated how similar some of the classes are and how these similarities led to worse results, as well as how removing the similar images from the "Other" class improved the classifier. Therefore, focusing on how to separate the classes and how to set up the classifiers and network to lessen the impact of these issues would be interesting future work on this subject.

Another interesting avenue for future work would be in would be to test different networks like ResNet50 or MobileNet V3 in order to compare their performance with this task. Additionally, getting a bigger data set, especially for the smaller classes like "Futuristic", would be a great way to improve accuracy. It could also be worth, setting up a new survey to do that, maybe with clearer instructions and examples to guide the participants more. Additionally, recent advances in AI text to paint models might pave the way to automatic creation of such a data set.In that regard, it would also be interesting to compare the interpretation of the AI to the human perception by having a survey. For example, this survey could be set up to ask participants what lighting mood a given image, which has been generated by the AI, has. Therefore, investigating if the AI generated images actually fit to the human perception of the mood given in the text prompt provided to the AI.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AmT] Amazon mechanical turk. https://www.mturk.com. Accessed: 2023-12-08.

[B$^+$91] Léon Bottou et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[BPH06] Kristin P Bennett and Emilio Parrado-Hernández. The interplay of optimization and machine learning research. *The Journal of Machine Learning Research*, 7:1265–1281, 2006.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[Far22] Hany Farid. Lighting (in)consistency of paint by text, 2022.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[KBKT17] Brady Kieffer, Morteza Babaie, Shivam Kalra, and H. R. Tizhoosh. Convolutional neural networks for histopathology image classification: Training vs. using pre-trained networks. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2017.

[KPB18] A. Kamilaris and F. X. Prenafeta-Boldú. A review of the use of convolutional neural networks in agriculture. *The Journal of Agricultural Science*, 156(3):312–322, 2018.

[KSL20] Karlo Koščević, Marko Subašić, and Sven Lončarić. Deep learning-based illumination estimation using light source classification. *IEEE Access*, 8:84239–84247, 2020.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[MA21] Sheldon Mascarenhas and Mukul Agarwal. A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification. In *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, volume 1, pages 96–99, 2021.

[OR19] Kamal M. Othman and Ahmad B. Rad. An indoor room classification system for social robots via integration of cnn and ecoc. *Applied Sciences*, 9(3), 2019.

[PW17] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[SAS19] Pinky Sodhi, Naman Awasthi, and Vishal Sharma. Introduction to machine learning and its basic application in python. *SSRN Electronic Journal*, 01 2019.

[STF19] Dwiretno Istiyadi Swasono, Handayani Tjandrasa, and Chastine Fathicah. Classification of tobacco leaf pests using vgg16 transfer learning. In *2019 12th International Conference on Information Communication Technology and System (ICTS)*, pages 176–181, 2019.

[Str] Stratifiedkfold. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html. Accessed: 2023-12-08.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[Tie12] Tijmen Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26, 2012.