Deep learning for characterizing full-color 3D printers: accuracy, robustness, and data-efficiency

vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

DISSERTATION

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
von

Danwu Chen

geboren in Zhanjiang, China

Referenten der Arbeit:	Prof. Dr. Arjan Kuijper Technische Universität Darmstadt Prof. Dr. techn. Dieter W. Fellner Technische Universität Darmstadt Prof. Dr. Philipp Urban Norwegian University of Science and Technology
Tag der Einreichung:	05/10/2023
Tag der mündlichen Prüfung:	21/11/2023

Darmstadt 2023 (year of the viva voce)
Darmstädter Dissertation
D 17

Danwu Chen: Deep learning for characterizing full-color 3D printers: accuracy, robustness, and data-efficiency
Darmstadt, Technische Universität Darmstadt,
Year thesis published in TUprints 2023
Date of the viva voce 21.11.23

Erklärung zur Dissertation

Hiermit versichere ich die vorliegende Dissertation selbständig nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 05/10/2023 Danwu Chen

Abstract

High-fidelity color and appearance reproduction via multi-material-jetting full-color 3D printing has seen increasing applications, including art and cultural artifacts preservation, product prototypes, game character figurines, stop-motion animated movie, and 3D-printed prostheses such as dental restorations or prosthetic eyes.

To achieve high-quality appearance reproduction via full-color 3D printing, a prerequisite is an accurate *optical printer model* that is a predicting function from an arrangement or ratio of printing materials to the optical/visual properties (e.g. spectral reflectance, color, and translucency) of the resulting print. For appearance 3D printing, the model needs to be inverted to determine the printing material arrangement that reproduces distinct optical/visual properties such as color. Therefore, the accuracy of optical printer models plays a crucial role for the final print quality. The process of fitting an optical printer model's parameters for a printing system is called *optical characterization*, which requires test prints and optical measurements. The objective of developing a printer model is to maximize prediction performance such as accuracy, while minimizing optical characterization efforts including printing, post-processing, and measuring.

In this thesis, I aim at leveraging deep learning to achieve *holistically*-performant optical printer models, in terms of three different performance aspects of optical printer models: 1) *accuracy*, 2) *robustness*, and 3) *data efficiency*.

First, for model accuracy, we propose two deep learning-based printer models that both achieve *high* accuracies with only a moderate number of required training samples. Experiments show that both models outperform the traditional cellular Neugebauer model by large margins: up to 6 times higher accuracy, or, up to 10 times less data for a similar accuracy. The high accuracy could enhance or even enable color- and translucency-critical applications of 3D printing such as dental restorations or prosthetic eyes.

Second, for model robustness, we propose a methodology to induce physically-plausible constraints and smoothness into deep learning-based optical printer models. Experiments show that the model not only almost always corrects implausible relationships between material arrangement and the resulting optical/visual properties, but also ensures significantly smoother predictions. The robustness and smoothness improvements are important to alleviate or avoid unacceptable banding artifacts on textures of the final printouts, particularly for applications where texture details must be preserved, such as for reproducing prosthetic eyes whose texture must match the companion (healthy) eye.

Finally, for data efficiency, we propose a learning framework that significantly improves printer models' data efficiency by employing *existing* characterization data from *other* printers. We also propose

a contrastive learning-based approach to learn dataset embeddings that are extra inputs required by the aforementioned learning framework. Experiments show that the learning framework can drastically reduce the number of required samples for achieving an application-specific prediction accuracy. For some printers, it requires only 10% of the samples to achieve a similar accuracy as the state-of-the-art model. The significant improvement in data efficiency makes it economically possible to frequently characterize 3D printers to achieve more consistent output across different printers over time, which is crucial for color- and translucency-critical individualized mass production.

With these proposed deep learning-based methodologies significantly improving the three performance aspects (i.e. accuracy, robustness, and data efficiency), a *holistically*-performant optical printer model can be achieved, which is particularly important for color- and translucency-critical applications such as dental restorations or prosthetic eyes.

Zusammenfassung

Die Wiedergabe von Farben und Erscheinungsbildern mit hoher Wiedergabetreue durch Multi-Material-Jetting-Vollfarb-3D-Druck findet zunehmend Anwendung, darunter die Konservierung von Kunst- und Kulturartefakten, Produktprototypen, Figuren von Spielfiguren, Stop-Motion-Animationsfilme und 3D-gedruckte Prothesen wie z Zahnrestaurationen oder Augenprothesen.

Um eine qualitativ hochwertige Reproduktion des Erscheinungsbilds durch vollfarbigen 3D-Druck zu erreichen, ist ein genaues *optisches Druckermodell* eine Voraussetzung, das eine Vorhersagefunktion aus einer Anordnung oder einem Verhältnis von Druckmaterialien zu den optischen/visuellen Eigenschaften (z. B. spektraler Reflexionsgrad) darstellt , Farbe und Lichtdurchlässigkeit) des resultierenden Drucks. Für den 3D-Druck mit Erscheinungsbild muss das Modell invertiert werden, um die Druckmaterialanordnung zu bestimmen, die bestimmte optische/visuelle Eigenschaften wie Farbe reproduziert. Daher spielt die Genauigkeit optischer Druckermodelle eine entscheidende Rolle Entscheidende Rolle für die endgültige Druckqualität. Der Prozess der Anpassung der Parameter eines optischen Druckermodells für ein Drucksystem wird als *optische Charakterisierung* bezeichnet und erfordert Testdrucke und optische Messungen. Das Ziel der Entwicklung eines Druckermodells besteht darin, die Vorhersageleistung zu maximieren B. Genauigkeit, bei gleichzeitiger Minimierung des Aufwands für die optische Charakterisierung, einschließlich Drucken, Nachbearbeitung und Messen.

In dieser Arbeit möchte ich Deep Learning nutzen, um *ganzheitlich* leistungsstarke optische Druckermodelle im Hinblick auf drei verschiedene Leistungsaspekte optischer Druckermodelle zu erstellen: 1) *Genauigkeit*, 2) *Robustheit* und 3) *Dateneffizienz*.

Erstens schlagen wir für die Modellgenauigkeit zwei auf Deep Learning basierende Druckermodelle vor, die beide *hohe* Genauigkeiten mit nur einer moderaten Anzahl erforderlicher Trainingsbeispiele erreichen. Experimente zeigen, dass beide Modelle das traditionelle zelluläre Neugebauer-Modell bei weitem übertreffen: bis zu sechsmal höhere Genauigkeit oder bis zu zehnmal weniger Daten für eine ähnliche Genauigkeit. Die hohe Genauigkeit könnte farb- und transluzenzkritische Anwendungen des 3D-Drucks wie Zahnrestaurationen oder Augenprothesen verbessern oder sogar ermöglichen.

Zweitens schlagen wir für die Modellrobustheit eine Methodik vor, um physikalisch plausible Einschränkungen und Glätte in Deep-Learning-basierte optische Druckermodelle zu induzieren. Experimente zeigen, dass das Modell nicht nur unplausible Zusammenhänge zwischen Materialanordnung und den resultierenden optischen/visuellen Eigenschaften fast immer korrigiert, sondern auch deutlich glattere Vorhersagen gewährleistet. Die Verbesserungen der Robustheit und Glätte sind wichtig, um inakzeptable Streifenartefakte auf Texturen der endgültigen Ausdrucke zu mildern oder zu vermeiden, insbesondere für Anwendungen, bei denen Texturdetails erhalten bleiben müssen, wie zum Beispiel für

die Reproduktion von Augenprothesen, deren Textur mit dem begleitenden (gesunden) Auge übereinstimmen muss.

Schließlich schlagen wir für die Dateneffizienz ein Lernframework vor, das die Dateneffizienz von Druckermodellen erheblich verbessert, indem es *vorhandene* Charakterisierungsdaten von *anderen* Druckern verwendet. Wir schlagen außerdem einen kontrastiven, lernbasierten Ansatz vor, um die Einbettung von Datensätzen zu erlernen sind zusätzliche Eingaben, die für den oben genannten Lernrahmen erforderlich sind. Experimente zeigen, dass das Lernrahmenwerk die Anzahl der erforderlichen Proben zum Erreichen einer anwendungsspezifischen Vorhersagegenauigkeit drastisch reduzieren kann. Bei einigen Druckern sind nur 10% der Proben erforderlich, um eine ähnliche Genauigkeit wie beim Stand der Technik zu erreichen Modell. Die deutliche Verbesserung der Dateneffizienz macht es wirtschaftlich möglich, 3D-Drucker häufig zu charakterisieren, um im Laufe der Zeit eine konsistentere Ausgabe über verschiedene Drucker hinweg zu erzielen, was für eine farb- und transluzenzkritische individualisierte Massenproduktion von entscheidender Bedeutung ist.

Mit diesen vorgeschlagenen Deep-Learning-basierten Methoden, die die drei Leistungsaspekte (d. h. Genauigkeit, Robustheit und Dateneffizienz) deutlich verbessern, kann ein *ganzheitlich* leistungsfähiges optisches Druckermodell erreicht werden, was besonders wichtig für farb- und transluzenzkritische Drucker ist Anwendungen wie Zahnrestaurationen oder Augenprothesen.

Acknowledgement

I would like to take this opportunity to thank all people, without whom my research journey wouldn't have accumulated momentum to arrive at this point.

First, I am grateful for the research funding of H2020 Marie Skłodowska-Curie Actions, which not only has allowed me to focus on research without being distracted by irrelevant stuff, but also has provided various academic communication/training opportunities.

I thank Fraunhofer IGD for providing not only a friendly working environment, but also a leading platform for applied research whose direct industrial/social applications motivate me often. I also thank the secretary team at IGD and the Interactive Graphics Systems Group (GRIS) for their support and help, in particular Sylvia Weber, Ursula Böck, Cornelia Kurkowski, and Georgia Agelopoulou.

I would like to thank my PhD supervisor Prof. Dr. Arjan Kuijper for his informative advices on paper/dissertation writing and publication venue selection, as well as his kind guidance regarding PhD process planing and advancing.

I would also like to thank Prof. Dr. techn. Dieter W. Fellner for serving as the second advisor of my PhD defense, as well as Prof. Stefan Roth, Ph.D., Prof. Dr. Anna Rohrbach and Prof. Dr. Justus Thies who agree to join the committee of my PhD defense.

My sincere Thank-You goes to Prof. Dr. Philipp Urban, who not only has provided kind supports and valuable coaching during my daily work and research, but also has substantially inspired my research with his professional expertise in color science as well as insights into the pain points of color management for 3D printing applications. Together we have had enjoyable interdisciplinary brainstorming and cooperation to bring together deep learning and full-color 3D printing to successfully accomplish the research goals mentioned in this thesis.

I thank all those who have been supportive at some points during my research for their advice or help, including Tejas Madan Tanksale, Alan Brunton, Johann Reinhard, and Mostafa Morsy Abdelkader Morsy.

Finally, my special thanks go to my wife Li, who has been always supportive and caring to me. Without her encouragement, I wouldn't even have been determined to start this research journey. Accompanying me to start this new journey, she had to move 10 thousand km away from hometown to enter a new environment where she had tough time to settle down in the beginning. She had to work so hard together with me to overcome difficulties. I am very happy to see that things have become much better now, and I am really grateful for her support, love and devotion. This thesis is dedicated to my beloved.

Contents

1. Introduction

Recent advances in multi-material-jetting have broadened applications of high-quality appearance reproduction via full-color 3D printing, including artwork replicas (e.g. for museum demonstrations), cultural artifacts preservation, product prototypes, game character figurines, stop-motion animated movie, and 3D-printed prostheses such as prosthetic eyes or dental restorations. For example, a groundbreaking application was reported [BBC,CNN,Fraa] that a British citizen became the first person in the world to be supplied with a 3D-printed prosthetic eye in November 2021, as shown in Fig. 1.1. According to the reports, this 3D printing-based strategy provides patients more realistic eye appearance via high-fidelity color and appearance reproduction of the healthy eye, and significantly shorter waiting time by replacing traditional handcraft process with a fully automatic process. Fig. 1.2 shows a more recent administered 3D-printed prosthetic eye, whose photo was demonstrated at Formnext, the world's leading trade fair for 3D printing, in November 2022. The methodologies proposed in this thesis had been used to significantly improve the appearance reproduction accuracy and reduce unacceptable artifacts for these 3D-printed eyes.

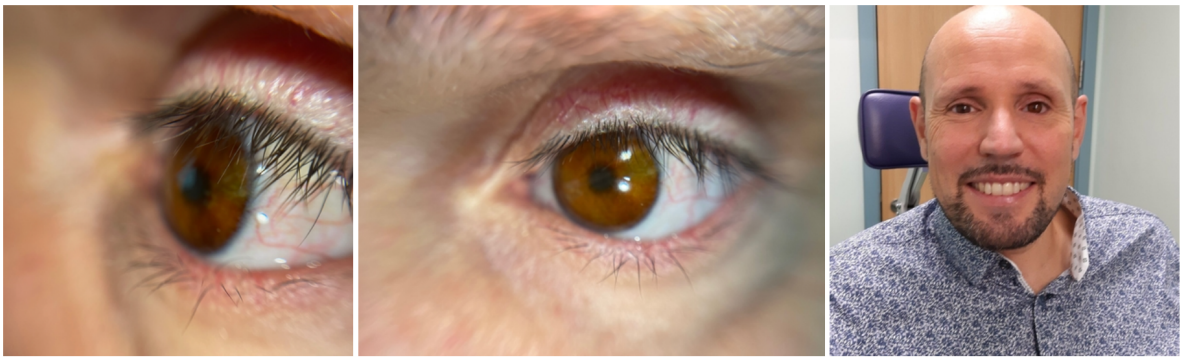


Figure 1.1.: The first patient in the world to be supplied with a 3D printed prosthetic eye (The patient's left eye) [BBC, CNN, Fraa]. Images by courtesy of Professor Mandeep Sagoo, consultant ophthalmologist at Moorfields Eye Hospital and professor of ophthalmology at the NIHR Biomedical Research Centre at Moorfields Eye Hospital and UCL Institute of Ophthalmology

Fig. 1.3 shows 3D printing's application in manufacturing several 100,000 colorful character figurines. The methodologies proposed in this thesis were used for color management for these 3D-

Figure 1.2.: The patient's right eye (left in the figure) is a 3D-printed prosthetic eye created by a fully-automatic digital end-to-end workflow. Input of the workflow are Optical Coherence Tomography (OCT) 3D images from the eye socket and the patient's left eye (healthy), as well as a color calibrated 2D image of the left eye. All images are taken by Tomey's Casia 2 Advanced OCT. Fraunhofer IGD's Cuttlefish:Eye data-driven design software is used to compute a digital model of the prosthetic eye that is then printed via the 3D printer driver Cuttlefish on a Stratasys J750 multi-material 3D printer. Image rights: Occupeye Ltd (stephen.bell@occupeye.com)

printed figurines. Fig. 1.4 shows 3D printing's application in a stop-motion animated movie, *Missing Link*, which won the Golden Globe Award for Best Animated Feature Film and received a nomination at the 92nd Academy Awards (also known as the Oscars) for Best Animated Feature. More 3D printed examples are shown in Appendix B.1.3 to show a few high-resolution multimaterial 3D prints that reproduce both color and translucency in addition to shape, illustrating application areas of colorful 3D printing. Note that 3D-printed character faces in the movie Missing Link (Fig. 1.4) and the examples shown in Appendix B.1.3 had been printed before our methodologies were proposed and they did not use our methodologies. Nevertheless, they are demonstrated here for illustrating application areas.

To achieve high-quality appearance reproduction via full-color 3D printing, a prerequisite is an accurate *optical printer model* that is a predicting function from an arrangement or ratio of printing materials to the optical or visual properties of the resulting print, e.g. spectral reflectance, color, and translucency. For appearance 3D printing, the model needs to be inverted to determine the printing material arrangement that reproduces distinct optical properties or visual properties such as color or translucency. Therefore, the accuracy of optical printer models is crucial for the final print quality.

Figure 1.3.: Several 100,000 color figurine prints via Cuttlefish (A 3D printer driver developped by Fraunhofer IGD) on Mimaki 3DUJ-553 fabricated. Image rights: Hero Forge [Herb] (the left image) and DeadAussieGamer [Hera] (the right image).

1.1. Printing phenomena, characterization data collection, and problem formulation

Optical characterization of a printing system is the process of fitting an optical printer model's parameters, which requires test prints and optical measurements.

1.1.1. Material Arrangements, Material Cross-Contamination and Dot Gain

Material arrangements in multi-material printing systems are specified by a voxel distribution at print resolution. A voxel is the smallest geometric structure (cuboid), that a 3D printing system can address. Most systems use anisotropic voxels with a much smaller extent orthogonal to the slice plane. Commonly each voxel can be filled with only one material. Some printing systems allow variable drop sizes and can fill a voxel with multiple materials.

The printer model has to consider two major effects:

1. Mechanical dot gain caused by material cross-contamination: Due to mechanical limitations, a printing system is generally not capable of filling a voxel with the intended material(s) perfectly. There is some cross-contamination of materials between neighboring voxels. In 3D printing, a mixing of materials across voxels is also necessary to ensure the structural integrity of the object.

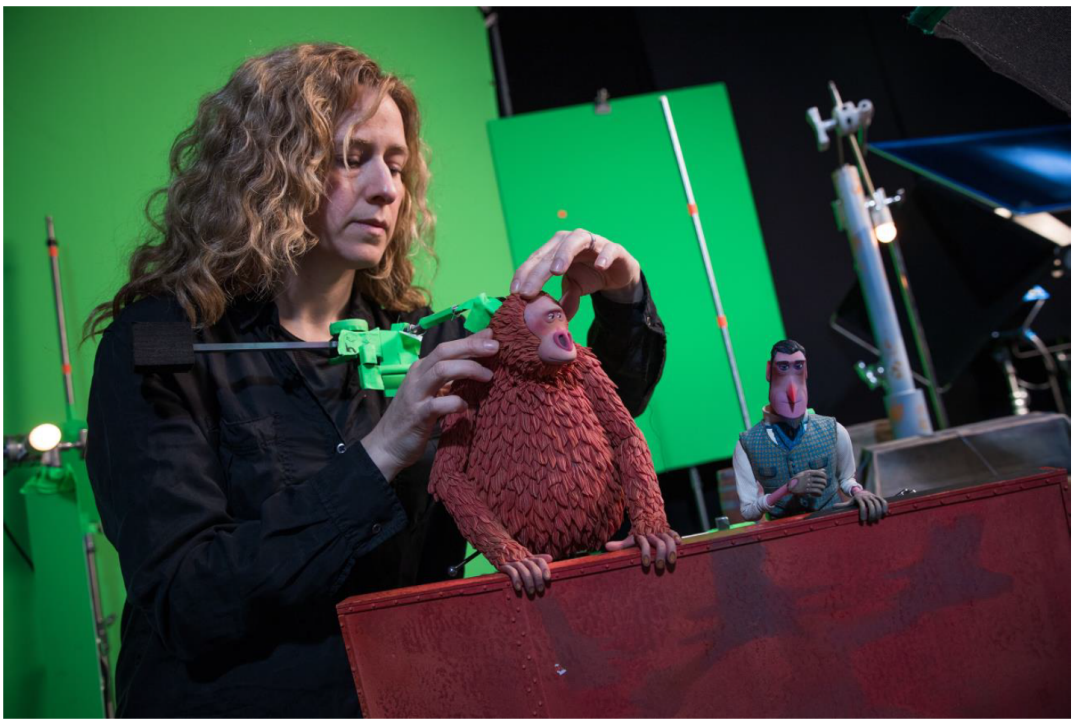


Figure 1.4.: To animate the characters' facial expressions in Missing Link, LAIKA used a number of Stratasys J750 3D printers, in conjunction with the Cuttlefish® 3D-printer driver, to create 106,000 highly detailed color 3D faces [frab, beh]. Special thanks to LAIKA Studios for the image.

2. Optical dot gain caused by light transport blurring: For translucent printing materials, incident light is transported through multiple voxels before it leaves the objects. Its spectral power distribution (SPD) is altered by all voxels in its path. Voxels filled with dark material surrounded by voxels filled with bright materials appear therefore bigger, which is called *optical dot gain*.

Light transport is described by the Radiative Transfer Equation (RTE) [Cha60] and requires the knowledge of the intrinsic optical properties of the printing materials, i.e. spectral absorption, scattering and phase function. Furthermore, refractive indexes of the printing materials are required since Fresnel reflection at the material-air interface and between the materials impact light transport as well. Using the RTE it is principally possible to simulate the light transport within heterogeneous media using a Monte-Carlo-based path tracer [ZHF*18]. This was done in the context of 3D printing by Sumin *et al.* [SRB*19], who employed the physical-based rendering engine Mitsuba [Jak10] for local path tracing. There are, however, various problems with Monte-Carlo-based approaches:

1. It is difficult to measure the intrinsic optical properties of the printing materials.
2. It is difficult to measure material cross-contamination (material mixing between voxels) but it must be considered for accurate simulations.

3. Due to the high resolution of modern 3D printing systems, Monte-Carlo approaches are slow. Even for computing the appearance of small objects (a few centimeters), billions of voxels must be considered and a simulation may take days on today's consumer hardware.

Because of the shortcomings of Monte-Carlo-based approaches, optical printer models use either phenomenological quantities such as measured reflectances of selected printing material arrangements or simplifications of the RTE.

Almost all models do not directly use voxel-based material arrangements as input but material ratios (or transformed ratios) instead. This assumes intrinsically that material distributions are dominated by high spatial frequencies minimizing material agglomerations. Modern 3D halftoning methods produce material arrangements possessing such *blue-noise characteristics* [BAU15]. This minimizes graininess and apparent banding artifacts, particularly in smooth color transitions, by exploiting the human visual system's low contrast sensitivity for high spatial frequencies. We refer to material ratios or transformed material ratios as *tonal values* and denote the tonal value space by $\mathcal{T} = [0,1]^M$ where M is the number of explicitly specified materials. We use $M = 5$ corresponding to the materials colored in Cyan (C), Magenta (M), Yellow (Y), Black (K), and a fully Transparent (T) material (also referred to as Clear material). A more detailed description of the tonal space will be presented in Section 1.1.3.

1.1.2. Characterization data collection

We aim to predict color and translucency of the arrangement of printing materials corresponding to each tonal value in the tonal space $\mathcal{T}$. To fit an optical printer model and to evaluate its accuracy, we need a thorough specification of how color and translucency of test prints are measured.

1.1.2.1. Measuring Optical Properties

For color, we measure reflectance spectra in an off-specular 45/0 measurement geometry [AST11] as shown in Fig. 1.5(a) using a white backing. For measuring the reflectances of highly translucent materials, edge loss [YKOI11] must be considered, which is a darkening effect caused by subsurface light transport away from the illuminated area. For this, the illuminated area D must be much bigger than the detection area A [ABTU15] (both is adjusted by circular apertures). To avoid measurement bias by edge loss for highly translucent printing materials we use $D \approx 90A$. The reflectance corresponding to a tonal value $v \in \mathcal{T}$ is determined by

$$r_{\lambda,D}(v) = r^w_\lambda \frac{\int_A E_{\lambda,D}(v,x)\,dx}{\int_A E^w_{\lambda,D}(x)\,dx} \tag{1.1}$$

where $\lambda \in [400\text{nm}, 700\text{nm}]$ is the wavelength, $E_{\lambda,D}(v,x)$ is the radiance emitted perpendicularly from the print's surface at point $x \in \mathbb{R}^2$ when illuminated at area D, $E^w_{\lambda,D}(x)$ is the radiance emitted perpendicularly from the surface of the opaque white reference (Lambertian) at point x using the same illumination and r^w_λ is the reference white's reflectance. For evaluating color prediction accuracy, CIELAB

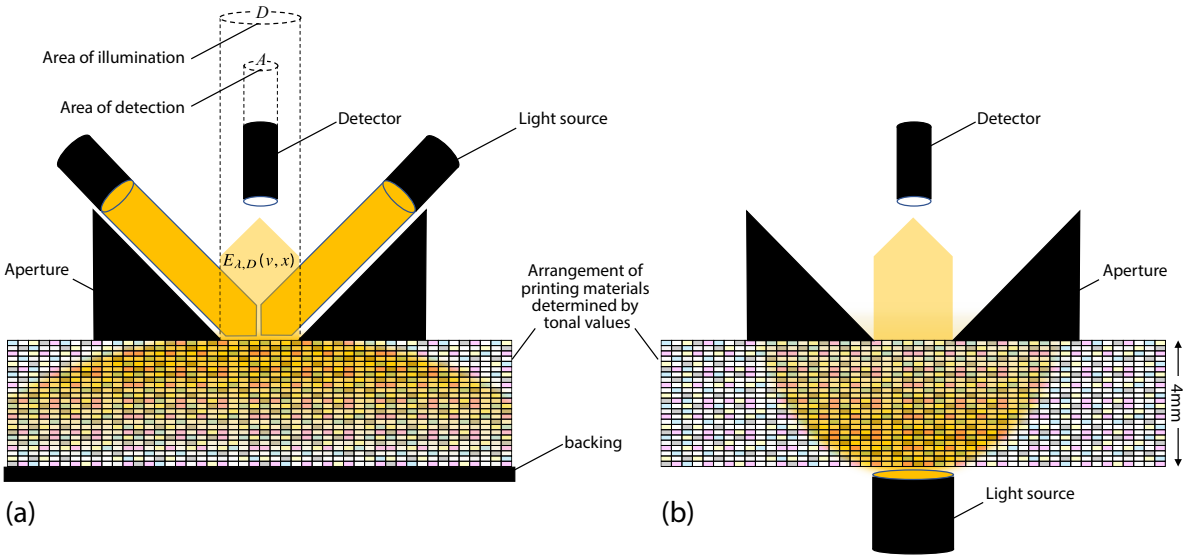


Figure 1.5.: Setup for reflectance (a) and transmittance (b) measurements for determining lateral and vertical light transport within printing material arrangements (shown here idealistically in a voxel grid) to compute the translucency quantity α [UTB*19]. Setup (a) with white backing and $D \approx 90A$ is used for spectral color measurements. This figure is from our published paper [CU21].

values are then computed (See Section 1.1.2.2) from reflectances assuming specified viewing conditions.

For translucency, optical properties can be light transport distances/profiles parallel to the surface normal [DWP*10a, HFM*10b]. In this paper, we use the one-dimensional quantity α proposed by Urban *et al.* [UTB*19]. It considers subsurface light transport parallel and orthogonal to the surface normal creating visual cues (shine through effects, reduction of high-spatial-frequency luminance contrasts) used by the human visual system to judge the degree of translucency [FB05, Mot10]. For measuring α, two spectrophotometric reflectance measurements using black backing (see Fig. 1.5(a)) and a spectrophotometric transmittance measurement (see Fig. 1.5(b)) are required in addition to the color measurement. The two reflectance measurements use different illumination areas, D_1 and D_2, adjusted by two apertures. D_1 is equal to the detection area, i.e. $D_1 = A$, causing severe measurement bias by edge loss for highly translucent materials and D_2 is much bigger than the detection area, $D_2 = 16A$, resulting in less edge loss. The resulting reflectance measurements, $r_{\lambda,D_1}(v)$ and $r_{\lambda,D_2}(v)$, are used to compute the edge-loss difference [YKOI11], a quantity that describes subsurface light transport distances orthogonal to the surface normal. The definition of the α-value for a printing material arrangement defined by the tonal value $v \in \mathcal{T}$ is based on vertical light transport determined by the measured transmittance $t(v)$, lateral light transport determined by the edge loss difference measured by $r_{\lambda,D_1}(v)$ and $r_{\lambda,D_2}(v)$, and the lightness of the color measurement. We refer for more details to Urban *et al.* [UTB*19].

1.1.2.2. Computing color from reflectance measurements

For the computation of CIELAB color from reflectances, we refer to Chapter 5 of book *Academic Press Library in Signal Processing* [USD14]. Nevertheless, to facilitate reading, in this section we also quote the computation details from that book. For more details, readers are referred to Chapter 5 of the aforementioned book.

CIELAB color space is derived from the CIEXYZ color space where CIEXYZ colors are device-independent and are obtained from reflectances. CIEXYZ colors are obtained by:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K \int_{\Lambda} r(\lambda) l(\lambda) \begin{bmatrix} \bar{x}(\lambda) \\ \bar{y}(\lambda) \\ \bar{z}(\lambda) \end{bmatrix} d\lambda \tag{1.2}$$

where λ is wavelength; $\Lambda = [380\text{nm}, 730\text{nm}]$ is the human visible wavelength range; $\bar{x}$, $\bar{y}$ and $\bar{z}$ are color matching functions (CMFs) describing a CIE standard observer that represent an average human's chromatic response within a certain arc (e.g. 2° or 10°) inside human eye fovea; r is the reflectance; l is the viewing illuminant; and K is a constant that normalizes Y to one for the perfect reflecting diffuser i.e. $r(\lambda) = 1$. In this thesis, we use the CIE 2° standard observer, the CIED50 illuminant, and equidistant sampling in wavelength range [400nm, 700nm] in 10nm steps. Methodologies in this thesis can be generalized to other illuminants and observers (For more details on definitions of illuminants and observers, readers are referred to [Ber00]), as well as different sampling methods in the human visible wavelength range.

The CIEXYZ color space is far from being perceptually uniform, i.e. uniform coordinate changes at different locations in the CIEXYZ color space does not cause uniform color changes perceived by human eyes. A color space with improved perceptual uniformity is the CIELAB color space, which derives from the CIEXYZ color space. The CIELAB color space has a lightness axis L^*, a red-green axis a^* (negative/positive values correspond to green/red), and a blue-yellow axis b^* (negative/positive values correspond to blue/yellow). It derives from the CIEXYZ color space by an invertible transformation from CIEXYZ coordinates (X, Y, Z) to CIELAB coordinates (L^*, a^*, b^*):

$$\begin{aligned} L^* &:= 116 f\left(\frac{Y}{Y_w}\right) - 16, \\ a^* &:= 500 \left[f\left(\frac{X}{X_w}\right) - f\left(\frac{Y}{Y_w}\right) \right], \\ b^* &:= 200 \left[f\left(\frac{Y}{Y_w}\right) - f\left(\frac{Z}{Z_w}\right) \right], \\ f(\xi) &:= \begin{cases} \xi^{\frac{1}{3}}, & \xi \geq 0.008856, \\ 7.787\xi + \frac{16}{116}, & \text{else}, \end{cases} \end{aligned} \tag{1.3}$$

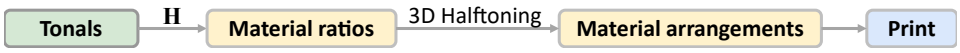


Figure 1.6.: Multi-material 3D printing pipeline. This figure is from our published paper [CU21].

where X_w, Y_w and Z_w are the white point's CIEXYZ values corresponding to the perfect reflecting diffuser i.e. $r(\lambda) = 1$.

1.1.3. Problem formulation

A 3D printing system with $M+1$ materials can be controlled by a tonal space of M dimensions, because material ratios must sum up to one in 3D printing (*unity condition*) and the fraction of the $(M+1)^{\text{th}}$ material in the material mixture is implicitly defined by the sum of the other material fractions. We use $M = 5$ corresponding to the materials colored in Cyan (C), Magenta (M), Yellow (Y), Black (K), and a fully Transparent (T) material (also referred to as Clear material). The implicitly defined material is White (W). For example, for a printer using CMYKWT materials, $\mathbf{t} = (0,0,0,0,0)$ corresponds to the pure white ink. This approach can be generalized to more and other materials.

We aim to allow all tonal values in the hypercube $[0,1]^M$ to be valid inputs. We denote $\mathcal{T} = [0,1]^M$ to be the tonal space. It does not directly represent material ratios but needs to be converted to material ratios ensuring the unity condition by a transform $\mathbf{H} : \mathcal{T} \mapsto [0,1]^M$, with $\|\mathbf{H}(\mathbf{t})\|_1 \leq 1, \forall \mathbf{t} \in \mathcal{T}$ from which the fraction of the white material can be computed as $1 - \|\mathbf{H}(\mathbf{t})\|_1$. $\mathbf{H}$ is part of the 3D printing pipeline before 3D halftoning (see Fig. 1.6) and is a function composition $\mathbf{H} = \mathbf{P} \circ \mathbf{Q}$, where $\mathbf{P}(\mathbf{t}) = \left(\mathbf{t}_1 / \max\{\|\mathbf{t}\|_1, 1\}, \ldots, \mathbf{t}_M / \max\{\|\mathbf{t}\|_1, 1\}\right)$ is a projection of the tonals to ensure the unity condition. $\mathbf{Q} : [0,1]^M \mapsto [0,1]^M$ can be an identity function but recommended to be an invertible transform similar to the transforms from nominal to effective tonals (e.g. 1D-per-tonal curves as described in [WB00]). This invertible transform aids the selection of printing patches corresponding to a more uniform distribution of optical quantities to fit/train the optical printer models.

Optical printer models predict optical properties of the resulting print from an arrangement or ratio of printing materials. For the sake of simplicity, we just use two optical quantities that state-of-the-art 3D printing systems can both reproduce within physical limits to describe the approach: Spectral reflectance to predict color and α for translucency [UTB*19]. It is worth mentioning that deep learning-based methodologies proposed by this thesis do not rely on any special properties of the two selected optical quantities except their boundedness. Therefore, our methodologies can be canonically extended to predict other optical quantities, given appropriate loss functions are selected.

Optical printer models investigated in this thesis is a function $\Phi : \mathcal{T} \mapsto \mathcal{S} \times \mathcal{A}$ that predicts spectral reflectances $\mathbf{r} \in \mathcal{S} = [0,1]^N$ and translucency $\alpha \in \mathcal{A} = [0,1]$ from material arrangement i.e. tonals $\mathbf{t} \in \mathcal{T} = [0,1]^M$. N is the number of considered wavelengths that is set to $N = 31$ with equidistant sampling in the range of [400nm, 700nm] (that stretches across the vast majority of human visible spectrum). The objective of developing a printer model is to maximize prediction performance such as accuracy, while minimizing optical characterization efforts including printing, post-processing, and measuring.

This section (Section 1.1) provided an overview about printing phenomena impacting visual appearance, characterization data collection, and a problem formulation. In the next section, we will present research questions that are tackled by this thesis.

1.2. Research questions

In this thesis, we aim at leveraging deep learning's capabilities of representation learning to achieve *holistically*-performant optical printer models, in terms of three different performance aspects: 1) *accuracy*, 2) *robustness*, and 3) *data efficiency*. To achieve this challenging goal, this thesis tackles three Research Questions (RQ) corresponding to the three performance aspects:

1.2.1. Accuracy

Among all three performance aspects, the most critical one is the model prediction accuracy (i.e. accurately predicting the optical properties from a printing materials arrangement or ratio). An accurate optical printer model is a prerequisite for high-quality appearance reproduction via full-color 3D printing and thus is crucial for color- and translucency-critical applications such as 3D printed prosthetic eyes or dental restorations. In this thesis, we are interested in leveraging deep learning to achieve *highly accurate* learning-based printer models, a research field that is still in its infancy. This leads to our first research question:

RQ1: *How to leverage deep learning to achieve* ***high accuracies*** *of optical printer models?*

To answer *RQ1*, in Chapter 3, we will present two deep learning-based printer models that both achieve high accuracies with a moderate number of required training samples.

1.2.2. Robustness

Since a deep learning-based optical printer model is, essentially or at least to a certain degree, a black-box without considering any physical grounding, it is sensitive to outliers or noise of the training data and tends to create physically-implausible tonal-to-optical relationships. The optical printer model is required to be not only accurate on-average with a low bias, but also to be reliable robust with low variance. The robustness is important to alleviate or avoid unacceptable banding artifacts on textures of the final printouts, particularly for applications where texture details must be preserved, such as for reproducing prosthetic eyes whose texture must match the companion (healthy) eye.

This leads to an *open* research question:

RQ2: *How to improve the* ***robustness and plausibility*** *of deep learning-based optical printer models?*

To answer *RQ2*, in Chapter 4, we will present a methodology to improve the robustness and plausibility of deep learning-based optical printer models by inducing physically-plausible constraints and smoothness into the models.

1.2.3. Data efficiency

In addition to prediction accuracy and robustness, another important performance aspects of optical printer models is resource efficiency, i.e. the number of required training prints and measurements to fit the model's parameters shall be as small as possible for a similar accuracy. This is particularly important because multi-material 3D printing is time-consuming and expensive, with regard to machine and material costs. Furthermore, optically characterizing a printing system is a recurring process because of inter- and intra-printer variability and the rapidly growing number of combinable printing materials. In practice, besides the data collected from the printer to be characterized, we might also have access to historical characterization data of *other* printers. Naturally, one might ask: Given the similarity between some printers, is the historical data from other printers helpful for the characterization of a printer of our interest? If so, how could we make use of these supporting datasets? This leads to an *open* research question:

RQ3: *How to exploit other printers' data to improve* ***data efficiency*** *of the characterization for a particular printer?*

To answer *RQ3*, in Chapter 5, we will present a learning framework that significantly improves printer models' data efficiency by employing supporting data from other printers.

1.3. This thesis

This chapter first presented the importance of highly-performant optical printer models; it then presented printing phenomena impacting visual appearance, characterization data collection, and a problem formulation, respectively; finally it defined three research questions that are to be tackled in order to achieve holistically-performant optical printer models. The rest of this thesis is organized as follows:

Chapter 2 presents relevant deep learning concepts and an overview of related works on optical printer models.

Chapter 3, answering *RQ1*, presents two deep learning-based printer models that both achieve high accuracies with a moderate number of required training samples.

Chapter 4, answering *RQ2*, presents a methodology to improve the robustness and plausibility of deep learning-based optical printer models by inducing physical heuristics into the models, enhancing robustness to erroneous data as well as plausible smoothness in model predictions.

Chapter 5, answering *RQ3*, presents a learning framework that significantly improves data efficiency for characterizing a targeted printing system, by exploiting characterization data from other printing systems. Since the learning framework is conditioned on dataset embeddings, we also propose a contrastive learning-based approach to learn dataset embeddings.

Chapter 6 concludes this thesis with an overview of the research questions and our corresponding contributions, as well as an outlook on future work.

1.3.1. Conclusion

To achieve high-quality appearance reproduction via multi-material 3D printing, a prerequisite is an optical printer model that accurately predicts from an arrangement or ratio of printing materials the optical properties of the resulting print.

This thesis aims at leveraging deep learning to achieve holistically-performant optical printer models, in terms of three performance aspects of optical printer models: 1) *accuracy*, 2) *robustness*, and 3) *data efficiency*. This thesis tackles this challenging goal by answering three corresponding Research Questions (RQ):

RQ1: *How to leverage deep learning to achieve* ***high accuracies*** *of optical printer models?*

RQ2: *How to improve the* ***robustness and plausibility*** *of deep learning-based optical printer models?*

RQ3: *How to exploit other printers' data to improve* ***data efficiency*** *of the characterization for a particular printer?*

To answer these three research questions, Chapter 3-5 will present three deep learning-based methodologies, respectively. As will be shown in these chapters, experiments on multiple state-of-the-art multi-material 3D printers demonstrate significant improvements in each of these three performance aspects, leading to accurate, robust, and data-efficient optical printer models that have benefited real-world applications of full-color 3D printing e.g. 3D-printed prosthetic eyes as mentioned in the beginning of this chapter.

In the next chapter, we will provide technical background knowledge of relevant deep learning techniques, and then present an overview of related works.

2. Relevant deep learning techniques and related work

The previous chapter presented an overview about printing phenomena impacting visual appearance, optical measurements and a problem formulation. It then defined three research questions that are to be tackled in order to achieve holistically-performant optical printer models.

Since this thesis aims at leveraging deep learning to achieve performant optical printer models, in the first section of this chapter we will first present technical background knowledge of relevant deep learning techniques. We then present an overview of related works in the second section.

2.1. Deep learning

In this section, we will first present the connection between machine learning concepts and printer characterization, then provide a brief overview of deep learning, and finally have more discussion on the most relevant deep learning architecture to this thesis. Note that here only the most relevant machine learning concepts are covered, and interested readers are referred to books [GBC16, RN20] for detailed discussions.

2.1.1. Characterizing by learning

There are three types of learning [GBC16, RN20]: 1) *Supervised learning* learns from example input-output pairs to reveal a target but unknown function that maps from the input to the output (also termed the label or annotation). 2) *Unsupervised learning* learns patterns in the input without the output provided. An example is clustering, which assigns samples into groups each consisting of similar samples. 3) *Reinforcement learning* learns the next best action from a series of rewards or punishments feedback. Optical printer models investigated in this thesis all fall into the type of supervised learning.

Suppose a general supervise learning problem: there exists an unknown target function $f : \mathcal{X} \mapsto \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces respectively. Given a dataset $\mathcal{D}$ of n independent input-output pairs (Each pair is referred to as an observation, a sample, or a data point), i.e. $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y} \mid \mathbf{y}_i = f(\mathbf{x}_i)\}_{i=1}^{n}$, one is asked to discover a function $h : \mathcal{X} \mapsto \mathcal{Y}$ that approximates the true but unknown function f. The function h is called a *hypothesis*. A certain learning algorithm uses data $\mathcal{D}$ to select h from a set of candidate hypotheses under consideration, which is referred to as the *Hypothesis Set* or *Hypothesis Space* $\mathcal{H}$. For example, for a regression task with input x and output y (For simplicity,

assume they are both scalars), one might choose polynomials of degree d to approximate the input-to-output relationship by predicting $\hat{y} = b + \sum_{j=1}^{d} w_j x^j$ where $b, w_1, ..., w_d \in \mathbb{R}$ are selectable/trainable parameters. In this case, $\mathcal{H}$ is the set of all polynomials $h_{b,w_1,...,w_d}(x) = b + \sum_{j=1}^{d} w_j x^j$ parametrized by $b, w_1, ..., w_d$, i.e. $\mathcal{H} = \{h_{b,w_1,...,w_d}\}_{b,w_1,...,w_d \in \mathbb{R}}$. The learning algorithm is to find the best hypothesis h^* from hypothesis space $\mathcal{H}$, given data $\mathcal{D}$.

To find the best hypothesis h^*, one could use *maximum a posteriori* (MAP) [GBC16,RN20] to choose the hypothesis h^* that is the most probable given the observed data, i.e. the hypothesis that maximizes the posterior probability:

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(h|\mathcal{D}) \tag{2.1}$$

With Bayes' rule, this maximization is equivalent to:

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} \frac{P(\mathcal{D}|h)P(h)}{P(\mathcal{D})} \tag{2.2}$$

Since $P(\mathcal{D})$ in the denominator does not depend on h, the above maximization is simplified to:

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(\mathcal{D}|h)P(h) \tag{2.3}$$

where $P(h)$ is the *hypothesis prior* probability representing how probable a hypothesis h is before data $\mathcal{D}$ is observed, and $P(\mathcal{D}|h)$ is the likelihood of data $\mathcal{D}$ under hypothesis h representing how likely $\mathcal{D}$ is observed given hypothesis h.

Before the data is observed, one might favor some hypotheses over the others, based on the one's prior knowledge or belief. For example, one could discourage very unusual-looking functions (e.g. polynomials of very high degrees) by lowering the functions' prior probability $P(h)$. Note that a hypothesis h with a low prior probability $P(h)$ still has the chance to be the optimal hypothesis, given that the data likelihood $P(\mathcal{D}|h)$ is sufficiently high such that $P(\mathcal{D}|h)P(h)$ is high. This means that the hypothesis, though appearing not very probable according to one's prior knowledge or belief, is well consistent with the observed data, thus could still be selected as the optimum. This could be the case when an unusually-complex task can only be well tackled by an unusually-complex hypothesis. On the other hand, one could also assume an *uniform* prior over the whole hypothesis space when there is no reason to favor any hypotheses over the others. This could be reasonable, when, for example, all hypotheses are equally complex.

When an uniform hypothesis prior $P(h)$ is assumed, $P(h)$ does not affect the maximization in Eq. (2.3) any more. Therefore, this maximization is further simplified to:

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(\mathcal{D}|h) \tag{2.4}$$

In this case, MAP reduces to selecting a hypothesis that maximizes only $P(\mathcal{D}|h)$, the likelihood of data $\mathcal{D}$ under hypothesis h. This is called *maximum likelihood estimation* (MLE), a special case of MAP.

As will be presented, our proposed optical printer models in Chapter 3 assume an uniform hypothesis prior over a selected hypothesis space. While the proposed optical printer models in Chapter 4 favors some hypotheses by inducing physically-plausible constraints identified *a priori*.

In practice, data $\mathcal{D}$ is usually not generated by a deterministic target function. Instead, it can be generated in a noisy way such that the output $\mathbf{y}$ is to some extent affected by but not fully determined by the input $\mathbf{x}$. For example, in characterization data collected from 3D printers, noise is inevitable during the printing and measuring processes due to factors such as machinery/environment variance. For a given input, the output might be stochastic thus have multiple possible values. In this case, the input-to-ouput process is not strictly a function. Therefore, the target we have to learn now is not a deterministic function $\mathbf{y} = f(\mathbf{x})$ any more, but rather a conditional probability distribution $P(\mathbf{y}|\mathbf{x})$ over all possible $\mathbf{y}$ values given $\mathbf{x}$. For a *classification* task where the output is to be predicted as one (or multiple) among a finite set of classes, one could define a model to predict the conditional probability distribution over classes. For a *regression* task where the output is a number, conceptually one predicts the conditional expectation or average of the output for the given input. As described in Section 1.1.3, an optical printer model investigated in this thesis is a function $\Phi : \mathcal{T} \mapsto \mathcal{S} \times \mathcal{A}$ that predicts spectral reflectances $\mathbf{r} \in \mathcal{S} = [0,1]^N$ and translucency $\alpha \in \mathcal{A} = [0,1]$ from material arrangement i.e. tonals $\mathbf{t} \in \mathcal{T} = [0,1]^M$. It falls into the type of regression tasks. That is, for a given tonal vector $\mathbf{t}$, the optical printer model predicts the conditional expectation of the resulting spectral reflectances $\mathbf{r}$ and translucency α.

For simplicity, we assume data points in dataset $\mathcal{D}$ are independent and identically distributed (i.i.d.). Let θ be the parameters (e.g. polynomial coefficients, or neural network weights) parametrizing a hypothesis h, i.e. the hypothesis space is $\mathcal{H} = \{h_\theta\}_\theta$. The maximization in Eq. (2.4) is equivalent to:

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmax}} P(\mathcal{D}|\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} P(\mathcal{D}^{(i)}|\theta) \end{aligned} \tag{2.5}$$

where $\mathcal{D}^{(i)}$ is the i^{th} data point in dataset $\mathcal{D}$ of cardinality n.

To avoid inconveniences and numerical underflow of the product, we could replace the likelihood with the logarithm of it, transforming the product into a summation. This leads to a result-equivalent maximization:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log P(\mathcal{D}^{(i)}|\theta) \tag{2.6}$$

One could harmlessly introduce a scaling factor $1/n$ to the summation, leading to an expectation w.r.t. an empirical data distribution $P_{\mathcal{D}}$:

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} \log P(\mathcal{D}^{(i)}|\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \mathbb{E}_{\mathbf{D} \sim P_{\mathcal{D}}} [\log P(\mathbf{D}|\theta)] \end{aligned} \tag{2.7}$$

With a negative sign added before the logarithm, the maximization is converted to a minimization:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathbb{E}_{\mathbf{D}\sim P_{\mathcal{D}}}[-\log P(\mathbf{D}|\theta)] \tag{2.8}$$

The expectation to be minimized above appears in the form of *cross entropy* $\mathbf{CE}(P,Q) = \mathbb{E}_P[-\log Q]$ between two probability distributions P and Q. Therefore, Eq. (2.8) is minimizing the cross entropy between the empirical data distribution $P_{\mathcal{D}}$ and the data likelihood distribution $P(\mathbf{D}|\theta)$. As shown below, this equivalently minimizes the dissimilarity between the two distributions.

By harmlessly adding another term that does not depend on θ, Eq. (2.8) becomes:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathbb{E}_{\mathbf{D}\sim P_{\mathcal{D}}}[-\log P(\mathbf{D}|\theta)] - \mathbb{E}_{\mathbf{D}\sim P_{\mathcal{D}}}[-\log P(\mathbf{D})] \tag{2.9}$$

where the first term is the same cross entropy as in Eq. (2.8), and the second term $\mathbb{E}_{\mathbf{D}\sim P_{\mathcal{D}}}[-\log P(\mathbf{D})]$ is the *entropy* of data distribution $P_{\mathcal{D}}$ (which is the same as the cross entropy of $P_{\mathcal{D}}$ with itself, i.e. $\mathbf{CE}(P_{\mathcal{D}}, P_{\mathcal{D}})$) and it depends on the data-generation process but not the hypothesis (thus not hypothesis parameters θ). Subtracting the entropy from the cross entropy yields the *relative entropy* or called *Kullback–Leibler divergence* (KL divergence), which is a measure of statistical distance, i.e. the distribution dissimilarity between two probability distributions. Therefore, the optimizations (Eq. (2.4) through Eq. (2.9)) in MLE (maximum likelihood estimation) can be interpreted as finding the best parameters $\theta = \theta^*$ that minimizes the KL divergence (a distribution dissimilarity) between the empirical data distribution $P_{\mathcal{D}}$ and the data likelihood distribution $P(\mathbf{D}|\theta)$ [GBC16, RN20].

The above optimizations in MAP and MLE are constrained to searching through a pre-selected hypothesis space. By selecting a class of predictive models (e.g. polynomials of a certain degree), one selects the hypothesis space. Models with insufficient capacity tend to fail to tackle complex tasks, while models with excessive capacity tend to overfit to the training data while performing poorly on unseen test data, so it is important to choose an appropriate class of models, or hypothesis space. For 3D printer characterization, we select deep learning, a class of machine learning techniques, because of its representational capacity that will be discussed in the next section. As will be presented in Chapter 3-5, we select hypothesis spaces by designing deep neural networks, and we inject prior preference of hypothesis via different regularization/constraints that prevent deep neural network-based optical printer models from overfitting or implausible predictions.

Before the data is observed, all hypotheses are simply as per the hypothesis prior distribution (uniform for MLE or arbitrary for MAP). As more and more data points are observed, "good" hypotheses start standing out, because data likelihood $\prod_{i=1}^{n} P(\mathcal{D}^{(i)}|\theta)$ (in Eq. (2.5)) for less "competent" hypotheses quickly diminishes with an increasing n (i.e. more samples are observed). This could also motivate the deep learning-based methodology in Chapter 5 that employs *existing* characterization data (for free) from *other* printers, without requiring more data from the target printer.

In the next section, we provide a brief overview of deep learning.

2.1.2. Deep learning

Deep learning is a branch of the broader family of machine learning. In recent years, deep learning has seen great successes in some historically difficult areas of machine learning, such as computer vision [DDS*09, KSH12, SZ15, HZRS16, KW22, GPAM*14, RBL*22], natural language processing [Kob18, HS97, VSP*17, DCLT19, Ope23], and robotics [MKS*15, SHM*16, BBC*23].

Deep learning is based on deep neural network (DNN) that is typically a network consisting multiple layers of artificial "neurons" or nodes, with each layer performing a transformation on features outputted by the previous layer(s) and then outputting the transformed features as the input of the next layer(s).

We view a typical deep neural network as a function $\Omega_\theta : \mathcal{X} \mapsto \mathcal{Y}$ mapping from an input space $\mathcal{X}$ to an output space $\mathcal{Y}$, where θ is the neural network's parameters. Let L be the number of layers performing transformations. The neural network's multilayer structure can be regarded as a composition of a series of transformation functions denoted as $(\Theta^{[i]}_{\theta^{[i]}})_{i=1}^{L}$ where $\Theta^{[i]}$ (The superscript $[i]$ represents the i^{th} layer.) is the transformation performed by the i^{th} layer parametrized by $\theta^{[i]}$ (Note that for neatness we sometimes do not explicitly show $\theta^{[i]}$). Then the neural network is a chain of transformations, i.e. $\Omega_\theta = \Theta^{[L]} \circ ... \circ \Theta^{[1]}$ and $\theta = (\theta^{[1]}, ..., \theta^{[L]})$.

2.1.2.1. Representation learning

With deep neural networks in core, deep learning performs *representation learning* that *automatically* learns to transform the input data to a meaningful representation (or features). Such transformations are parametrized by the neural network parameters (or called weights). An *objective function*, i.e. a loss to be minimized or a reward to be maximized (For simplicity, we only discuss minimizing a loss because maximizing a reward can be accomplished by minimizing its opposite), calculated on the final (and/or intermediate) outputs provides guidance signals to update neural network parameters to adjust the transformations so as to optimize the objective function, i.e. minimize a loss (e.g. a distance or dissimilarity between predictions and groundtruths). A learning algorithm is to find the best transformations $(\Theta^{[1]}, ..., \Theta^{[L]})^*$ that minimizes the loss function denoted as $J(\mathcal{D}; \Theta^{[1]}, ..., \Theta^{[L]})$ where $\mathcal{D}$ is a given dataset.

$$(\Theta^{[1]}, ..., \Theta^{[L]})^* = \underset{(\Theta^{[1]}, ..., \Theta^{[L]})}{\operatorname{argmin}} \; J(\mathcal{D}; \Theta^{[1]}, ..., \Theta^{[L]}) \tag{2.10}$$

Since the transformations $(\Theta^{[1]}, ..., \Theta^{[L]})$ are parametrized by $(\theta^{[1]}, ..., \theta^{[L]})$, intrinsically J depends on $(\theta^{[1]}, ..., \theta^{[L]})$, so the loss function can be re-denoted as $J(\mathcal{D}; \theta^{[1]}, ..., \theta^{[L]})$. Therefore, the minimization in Eq. (2.10) is intrinsically finding the best parameters $(\theta^{[1]}, ..., \theta^{[L]})^*$ that minimizes J:

$$(\theta^{[1]}, ..., \theta^{[L]})^* = \underset{(\theta^{[1]}, ..., \theta^{[L]})}{\operatorname{argmin}} \; J(\mathcal{D}; \theta^{[1]}, ..., \theta^{[L]}) \tag{2.11}$$

The optimization of neural network parameters is usually via the *steepest descent* (also referred to as *gradient descent*) optimization algorithm [NW06] or more commonly its variants e.g. mini-

batch stochastic gradient descent (SGD) [Bot99], Momentum [Pol64], Nesterov Accelerated Gradient [Nes83], or Adam [KB14]. These gradient-based optimization algorithms use the first-order partial derivatives of the loss function w.r.t. neural network parameters to find an update direction in which the loss function decreases fast. The algorithms then update parameters in the found direction with a step size controlled by a *learning rate*. Additionally, the variants of gradient descent leverage historical or future (estimated) derivatives to accelerate the training [Pol64], reduce oscillation [Nes83], or adaptively adjust learning rates [KB14]. Partial derivatives are calculated typically via a popular algorithm called *backpropagation* leveraging the chain rule of calculus [RHW86].

As shown by Eq. (2.10)-(2.11), given data $\mathcal{D}$, deep learning automatically learns representation transformations $(\Theta^{[1]}, ..., \Theta^{[L]})$ (via learning their parameters $(\theta^{[1]}, ..., \theta^{[L]})$), towards the minimization of a predefined loss function J.

This automatic representation learning is particularly contrast to the traditional handcrafted *feature engineering* that extracts features/representations based on domain knowledge. The manual feature engineering involves human intervention efforts (e.g. explicitly programming a feature extractor and evolving it by repeating brainstorming and testing) and is limited by domain knowledge that varies from task to task. For example, despite that image recognition tasks appear easy and intuitive to human, the underlying mechanisms of human vision and brain reasoning remain not fully revealed, which in turn limits the effectiveness of handcrafted feature extractors. In contrast, deep learning performs representation learning that automatically *learns* how to extract features or transform representation.

Furthermore, with a multilayer structure, deep learning is capable of representation learning in a hierarchical way [GBC16, Cho17, Gro17] where later layers could learn increasingly meaningful higher-level representations out of simpler lower-level representations from earlier layers. Intuitively, for example in an image recognition task, an earlier layer detects low-level features e.g. horizontal or vertical edges in an input image, and a later layer assembles these low-level features and further transforms them into high-level features e.g. object contours (e.g. of a plant or an animal) that are more meaningful for recognizing objects in the image. A multilayer structure can also be motivated by a general belief that the "true" function/process to be learnt involves composition of multiple simpler functions/steps each approximated by a hidden layer.

2.1.2.2. Different deep neural network architectures

There are a variety of deep neural network architectures including multilayer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN), and transformer. A distinct architecture might be favored over others, depending on the task of the learning. In the following paragraphs we provide a brief overview of a non-exhaustive list of deep neural network architectures:

Multilayer perceptron (MLP) is typically a multilayer fully-connected feedforward neural network [GBC16, Gro17] with one or more hidden layers of neurons, each is connected to all neurons of the previous layer. Each neuron is a weighted summation aggregation (parametrized by trainable weights

and bias parameters) of outputs of all neurons in the previous layer, followed by a transformation on the weighted summation via an activation function that is typically a non-linear function (e.g. rectified linear unit (ReLU) [MHN13]) so as to enable non-linear transformations. With a much more straightforward structure than other deep learning architectures, MLP is the quintessential deep learning model. In spite of its simple structure, MLP can be an effective model for a variety of tasks such as regression or classification tasks on tabular data. It is the de facto neural network structure for a new class of methods coined *neural fields* or *neural implicit representation* [MST*20, SMB*20, PFS*19, MESK22, LME*23], which has seen widespread success since 2019 in visual computing problems such as 3D scene reconstruction, novel view synthesis, and 3D scene editing.

Convolutional neural network (CNN) drastically reduces the number of parameters and thus the training difficulty compared to MLP because of 2 factors: 1) *sparse connection* (i.e. each neuron is connected to the previous layer's neurons within only a small local region termed *receptive field* e.g. a 5x5 pixel matrix, where the output neuron is calculated as a dot product between the local input neurons (within the output neuron's receptive field e.g. a 5x5 scalar matrix) and an array of trainable weights termed *filter* (e.g. a 5x5 scalar matrix)) followed by a non-linear transformation e.g. ReLU, and 2) *parameter sharing* (i.e. all output neurons share the same filter with which a convolutional operation is performed on the whole input (e.g. an image, or a transformed image (also called *feature map*) produced by the previous layer) in a sliding window manner). With parameter sharing, CNN is capable to recognize translationally-invariant structure or repeated patterns of images, which intuitively means an object (e.g. a plant or an animal) within the image can be recognized regardless of the object's spatial location within the image. With sparse connection, CNN can efficiently process visual imagery data. It is most commonly applied for computer vision such as image recognition [DDS*09, KSH12, SZ15, HZRS16] or image generation [KW22, GPAM*14, RBL*22].

recurrent neural network (RNN) repeatedly performs the same representation transformations shared through all (time) points (e.g. time points within a time series or words within a sentence) within a sequential data, via parameter sharing through all points within the sequence, thus is efficient to process long sequential data. It also passes intermediate transformed representation of each (time) point to the next and/or the previous point, allowing learning interactions/dependencies between points which is important for perceiving context. Classic RNN meets difficulties in learning long-range or long-term dependencies mainly due to the *vanishing gradient problem* [HS97]. Its variant *gated RNNs* including especially *long shot-term memory* (LSTM) [HS97] significantly ease the difficulties in learning long-term dependencies, by learning when to remember/forget and when to update information flowing across long-term context. RNN and its variants are suitable for processing sequential data such as natural language in the form of words or voice, and are mostly applied for natural language processing (NLP) such as machine translation and speech recognition.

Transformer, with a self-attention mechanism [VSP*17] as the core, is designed to efficiently process large datasets such as the Wikipedia Corpus, and is particularly used for natural language process-

ing such as training *large language models* (LLMs) [DCLT19, Ope23], the core of several recently-striking chatbots such as ChatGPT [Cha]. It also sees increasing applications in other fields such as computer vision [DBK*21].

Above we provided only a brief overview of a non-exhaustive list of deep neural network architectures, while MLP is the most relevant architecture to this thesis thus will be further discussed in the next section.

2.1.3. Multilayer perceptron neural network

Multilayer perceptron (MLP) neural network is a class of deep neural networks, consisting of 3 kinds of layers:

Input layer: An input layer is the first layer in an MLP neural network. It is essentially just an identity function $\mathbb{R}^{m_{in}} \mapsto \mathbb{R}^{m_{in}}$ taking as input the input features of a data point (or sample) $\mathbf{x} \in \mathbb{R}^{m_{in}}$ where m_{in} is the number of features in each input data point. The number of "neurons" or units in this layer is simply equal to the number of input features m_{in}. This layer's role is simply taking in the input data and feed it through the neural network.

Hidden layer: An MLP neural network contains L typically sequentially-stacked hidden layers between the input layer and the output layer. Each neuron in a hidden layer connects to all neurons in the previous layer. The i^{th} hidden layer ($1 \leq i \leq L$) with $m^{[i]}$ neurons (The superscript $[i]$ represents the i^{th} layer.) performs a representation (features) transformation $\Theta^{[i]} : \mathbb{R}^{m^{[i-1]}} \mapsto \mathbb{R}^{m^{[i]}}$ on its inputs i.e. the output features $\mathbf{h}^{[i-1]}$ of the $(i-1)^{\text{th}}$ layer (For simplicity we define the special case $\mathbf{h}^{[0]} = \mathbf{x}$, i.e. the features from the input layer, and $m^{[0]} = m_{in}$). $\Theta^{[i]}$ is a function composition $\Theta^{[i]} = \mathbf{g}^{[i]} \circ \mathbf{f}^{[i]}$. $\mathbf{f}^{[i]}$ is a function performing a linear transformation on $\mathbf{h}^{[i-1]}$:

$$\mathbf{z}^{[i]} = \mathbf{f}^{[i]}(\mathbf{h}^{[i-1]}) = \mathbf{W}^{[i]\top}\mathbf{h}^{[i-1]} + \mathbf{b}^{[i]}, \tag{2.12}$$

where $\mathbf{W}^{[i]} \in \mathbb{R}^{m^{[i-1]} \times m^{[i]}}$ is a *weights* matrix, $\mathbf{b}^{[i]} \in \mathbb{R}^{m^{[i]}}$ is a *bias* offset vector. $\mathbf{W}^{[i]}$ and $\mathbf{b}^{[i]}$ are trainable parameters, parametrizing the $\mathbf{f}^{[i]}$ transformation. $\mathbf{g}^{[i]}$ is an activation function that is typically a non-linear function (e.g. rectified linear unit (ReLU) $\mathbf{ReLU}(\mathbf{z})_j = \max(0, \mathbf{z}_j)$ for $j \in \{1, ..., \dim(\mathbf{z})\}$ for a vector $\mathbf{z}$) performing a non-linear transformation on representation $\mathbf{z}^{[i]}$. The output features of the i^{th} hidden layer is determined by

$$\mathbf{h}^{[i]} = \Theta^{[i]}(\mathbf{h}^{[i-1]}) = \mathbf{g}^{[i]}(\mathbf{f}^{[i]}(\mathbf{h}^{[i-1]})) = \mathbf{g}^{[i]}(\mathbf{W}^{[i]\top}\mathbf{h}^{[i-1]} + \mathbf{b}^{[i]}) \tag{2.13}$$

Output layer: An output layer is the final layer in an MLP neural network. Each of its neurons is connected to all neurons of the final hidden layer i.e. the L^{th} hidden layer. The number of neurons in this layer is usually set to be equal to the number of entries (denoted as m_{out}) in the groundtruth

$\mathbf{y} \in \mathbb{R}^{m_{out}}$ for each data point (Note that in some machine learning tasks, the number of neurons may be different from the number of entries in the groundtruth. For example, in a multiclass single-label classification task, the groundtruth in raw data is typically one scalar representing a class ID. This scalar groundtruth is typically one-hot encoded into a vector (1.0 (hot) for the "true" class and 0.0 (cold) for others) representing the groundtruth probability distribution, so that the training loss could penalize the distribution dissimilarity (e.g. indirectly measured as the cross entropy) between the one-hot encoded groundtruth (as a probability distribution) and the predicted probability distribution (a vector of probabilities over classes). In this case, the number of output neurons is equal to the number of classes. However, if we regard the one-hot encoded groundtruth as the "effective" groundtruth exposed to the neural network, then indeed the number of neurons equals the number of entries in the groundtruth. For simplicity, this equality is assumed.) The final layer performs the final transformation to make the final predictions $\hat{\mathbf{y}} \in \mathbb{R}^{m_{out}}$ as close as possible to the groundtruth $\mathbf{y}$. Similar to Eq. (2.13), the final prediction $\hat{\mathbf{y}}$ is determined by

$$\hat{\mathbf{y}} = \mathbf{g}_{out}(\mathbf{W}_{out}^{\top}\mathbf{h}^{[L]} + \mathbf{b}_{out}) \tag{2.14}$$

where $\mathbf{W}_{out} \in \mathbb{R}^{m^{[L]} \times m_{out}}$ is a weights matrix of the output layer, $\mathbf{b}_{out} \in \mathbb{R}^{m_{out}}$ is a *bias* offset vector, $\mathbf{g}_{out}$ is the activation function of the output layer and is usually tailored to meet the properties of the groundtruth $\mathbf{y}$ e.g. subjecting to a theoretical or empirically-estimated output range (e.g. a scaled sigmoid function where sigmoid is $\sigma(\mathbf{z})_j = \frac{1}{1+\exp(-\mathbf{z}_j)}$ to predict range-bounded values in a regression task, or a softmax function $\mathbf{softmax}(\mathbf{z})_j = \frac{\exp(\mathbf{z}_j)}{\sum_k \exp(\mathbf{z}_k)}$ to predict a probability distribution over classes in a multiclass single-label classification task).

2.2. Related Work

In this section, we will present an overview of related works.

Optical printer models can be classified into phenomenological models [Mur36, Neu05, RB93, YN51, YC51, Vig90, HECC05, HC05, HF13, CY55, CY53, HCE03, Rog00, HH15, AA98, BH16], models based on the Radiative Transfer Equation (RTE) or its simplifications [KM31, Sau42, VSAOS16a, VSAOS16b, SHHM16, ZHF*18], and neural-network-based models [Tom96, AM94, LDS02, SBK*19].

2.2.1. Phenomenological Models

Simple phenomenological optical printer models are adapted from 2D color printing. They usually operate on so-called *effective tonal* values computed from the input tonal values, called *nominal tonals*, to correct for mechanical dot gain. This correction is coined *linearization* and uses mostly 1D-functions per tonal [WB00] but also higher-dimensional functions have been proposed [RR10]. For duotone printers, e.g. using white and black materials, the simplest phenomenological model is the Murray-Davies model [Mur36], which is just a linear interpolation between reflectance spectra of the printing

materials. The canonical extension to N inks is called Neugebauer Model [Neu05], that uses multilinear interpolation of the so-called Neugebauer primary reflectance spectra corrected for mechanical dot gain. The Neugebauer primaries consists of the base-materials and any mixture with equal material ratios, i.e. an N material printer has 2^N Neugebauer primaries. The interpolation weights can be determined by the Demichel equations [Dem24a, Dem24b]. The prediction performance of the Neugebauer model is rather low already in 2D printing [RB93] because it neglects subsurface light transport. This can be addressed by an extension introduced by Yule and Nielsen [YN51, YC51, Vig90] who applied a simple power function to the Neugebauer primaries. Various enhancements of the Yule-Nielson modified spectral Neugebauer model were proposed [HECC05, HC05] yielding a prediction performance within the noise-level of 2D printing systems. For 3D printing systems, the Yule-Nielsen modified Neugebauer model, however, performs poorly [HF13]. Furthermore, it is conceptually not well suited to model 3D printing systems able to reproduce multiple levels of translucency, since using just one parameter (Yule-Nielson n-value) intrinsically relies on constant translucency of the substrate.

Another popular model is the Clapper-Yule model [CY55, CY53]. It considers multiple internal reflections at the material-air interface and can be adapted to various materials [HCE03, Rog00]. Also the Clapper-Yule model intrinsically assumes constant translucency of the substrate [HH15].

A way to arbitrarily improve the prediction performance of phenomenological models is to separate the tonal-value space into cells and evaluate the models within each cell. The most popular model is the cellular Yule-Nielsen spectral Neugebauer model and its modifications [RR10, AA98]. The improved model performance comes at the expense of more training prints and spectral measurements. For joint color and translucency reproduction with a 6-material 3D printer more than 3500 patches were necessary to achieve an average CIEDE2000 [CIE01] prediction error of 2.3 that is sufficient for most applications [BATU18]. Babaei and Hersch proposed a cellular Yule-Nielsen modified spectral Neugebauer Model using barycentric subdivision of the tonal value space [BH16]. This reduces the number of required prints and measurements particularly if many materials are used. A disadvantage is that the barycentric subdivision is less localized than the common cube-based subdivision and the interpolation yield larger maximum errors.

2.2.2. Models based on RTE-Simplifications

The advantage of the models based on RTE-simplifications is the limited number of necessary printed samples required for parameter fitting.

The simplest model is based on the Kubelka-Munk-Theory [KM31] that only considers two opposite collimated fluxes parallel to the surface normal. This Two-flux approximation of the RTE was used to model light propagation in layered arrangements of materials with different refractive indexes to fabricate spatially-varying translucencies [DWP*10a, HFM*10b]. The Kubelka-Munk-Theory assumes an infinite extent of the layers and their homogeneity. Since this does not hold for high-frequent 3D-halftones, the resulting color errors are rather high. Hensley and Ferwerda reported mean CIEDE2000 errors of 3.45 [HF13] which is unacceptable for most applications. To account for Fresnel reflection a Saunderson correction is often used [Sau42].

A four-flux approximation of the RTE has been used for modeling 2.5D (relief) prints. In addition to the fluxes considered by the Kubelka Munk theory it considers also diffuse up and down facing fluxes. Van Song *et al.* reported maximal CIE94 error rates of less than 2 [VSAOS16a, VSAOS16b]. Simonot *et al.* developed a four-flux model relying on a matrix formalism and extended it to predict the bidirectional scattering distribution function of a stack of layered materials [SHHM16].

2.2.3. Neural-Network-based Models

Already more than a quarter of a century ago neural networks were used to invert optical printer models (also called separation) in 2D printing, i.e. computing the tonal values necessary to reproduce a given color [Tom96] or to reproduce the same color as shown on a display [AM94]. Littlewood *et al.* [LDS02] used a one-hidden-layer feedforward network to model the relationship between tonal values and color.

Shi *et al.* [SBK*19] concatenated a fully-connected neural network of a backward spectral model (predicting ink layer layout from the spectral) and another fully-connected neural network of a forward spectral model (predicting spectral from ink layer layout), optimizing for both spectral RMSE and the CIE76 color difference formula for multiple illuminants, for a laboratory-scale 3D printer MultiFab. They reported mean CIEDE2000 errors of the forward spectral model ranging between 1.5 and 2.5 but relatively large maximum errors of approx. 12 for the considered illuminants.

2.3. Conclusion

This chapter presented background knowledge of relevant deep learning techniques and an overview of related works.

From the perspective of machine learning, this thesis regards developing an optical printer model as a regression task where we are to discover, given characterization data (i.e. tonal-optical pairs), a true but unknown function mapping from tonal to optical properties, or an unknown conditional data distribution over all possible optical properties given tonal. Specifically, for a given tonal vector $\mathbf{t}$, we are to develop an optical printer model predicting the conditional expectation of the resulting spectral reflectances $\mathbf{r}$ and translucency α. For this learning task, we select deep learning due to its representational capacity (i.e. ability to approximate complex functions) and its automatic representation learning (instead of handcrafted feature engineering). Multilayer perceptron (MLP) is the most relevant deep neural network architecture to this thesis.

Optical printer models in related works can be classified into phenomenological models, models based on the Radiative Transfer Equation (RTE) or its simplifications, and neural-network-based models. Achieving an optical printer model that has a sufficient accuracy for most applications remains challenging, and particularly difficult for color- and translucency-critical applications such as 3D printed prosthetic eyes or dental restorations. Neural network-based models show promising results but is still a research field in its infancy.

In this thesis, we propose deep learning-based methodologies to achieve holistically-performant optical printer models, in terms of not only *accuracy* (Chapter 3), but also *robustness* (Chapter 4), and *data efficiency* (Chapter 5).

In the next chapter, we will tackle *accuracy* via answering the first research question:

RQ1: *How to leverage deep learning to achieve* ***high accuracies*** *of optical printer models?*

3. Deep learning models for optically characterizing 3D printers

The previous chapter provided background knowledge of relevant deep learning techniques, and an overview of related works on optical printer models. An *accurate* optical printer model is challenging but crucial for achieving high-quality appearance reproduction via full-color 3D printing.

In this chapter, we will answer the first research question *RQ1* (i.e. *How to leverage deep learning to achieve **high accuracies** of optical printer models?*) by proposing two deep learning-based printer models that both achieve high accuracies with only a moderate number of required training samples.

This chapter is based on the published paper [CU21].

3.1. Introduction

Optical printer models are functions that predict a print's optical properties given the arrangement or ratio of printing materials. They are the prerequisite to accurately reproduce color [BAU15, ABTU15, SRB*19], translucency [HFM*10a, DWP*10b] or joint color and translucency [BATU18] in multi-material 3D printing. Achieving an optical printer model that has a sufficient accuracy for most applications remains challenging, and particularly difficult for color- and translucency-critical applications such as 3D printed prosthetic eyes or dental restorations.

In this chapter, we propose two deep learning-based printer models that both achieve *high* accuracies with a moderate number of required training samples. The first model is a *Pure Deep Learning* (PDL) model that is essentially a black-box without any physical ground, and the second model is a *Deep-Learning-Linearized Cellular Neugebauer* (DLLCN) model that uses deep-learning to multidimensionally linearize the tonal-value-space of a cellular Neugebauer model [RR10, AA98].

We make the following contributions:

1. We propose a deep learning model (PDL) for accurately predicting optical properties of a print from tonal values and a learning strategy requiring only a small number of training samples.
2. We propose a deep-learning-based approach (DLLCN) for multidimensionally linearizing the cellular Neugebauer model to improve its prediction accuracy requiring only a small number of training samples.

3. We propose a loss function for training both models balancing reflectance, color and translucency errors using the ratio of the corresponding just noticeable differences under indoor viewing conditions.

Experiments on two six-material polyjetting 3D printers show that both models can achieve accuracies sufficient for most applications with much fewer training prints compared to a regular cellular Neugebauer model.

3.2. The Pure Deep Learning (PDL) Model

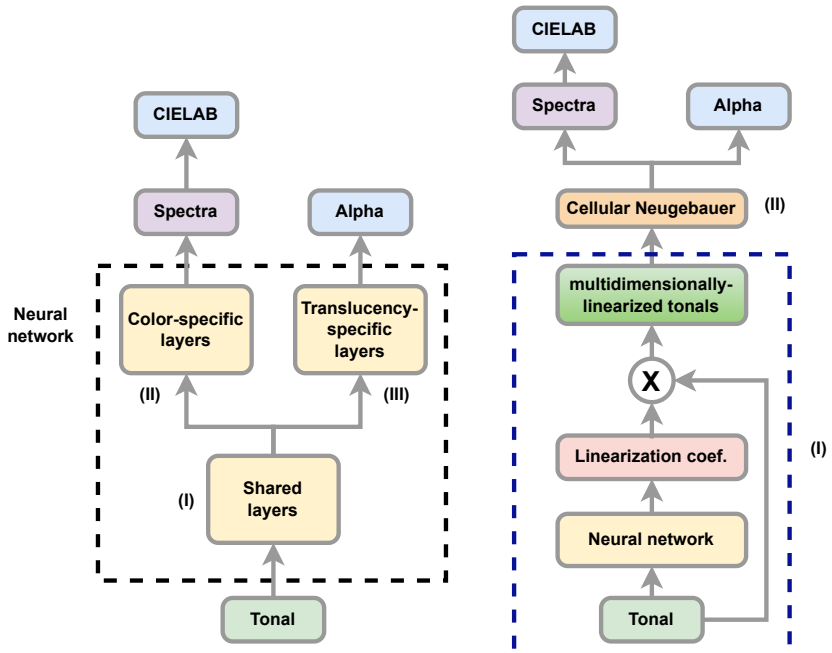
The first deep learning-based model we propose is the *Pure Deep Learning* (PDL) model. It is a prediction function $\Omega_\theta : \mathcal{T} \mapsto \mathcal{S} \times \mathcal{A}$, where Ω_θ is a neural network and θ is the neural network's parameters to be trained.

3.2.1. Multi-path fully-connected neural network

The core of the PDL model is a fully-connected neural network that, as shown in Fig. 3.1 (a), has multiple paths each corresponding to one predicting task: Built upon the input layer (tonal features) is the "trunk" (shown as part (I) in Fig. 3.1 (a)) that consists of a number of hidden layers, and it splits into two "branches" corresponding to the two tasks (i.e. the reflectance-predicting task (shown as part (II) in Fig. 3.1 (a)) and the translucency-predicting task (shown as part (III) in Fig. 3.1 (a))) each has its own hidden layers till its own output layer. The trunk is to learn generic representations shared across tasks, and each branch is to learn task-specific representations in order to make final predictions for the individual task. Specifically in our model, the trunk has 4 hidden layers each has 200, 1500, 1500 and 1500 neurons respectively. The reflectance branch has one hidden layer with 200 neurons and an output layer with N neurons corresponding to N-dimensional reflectance that is then converted to 3-dimensional CIELAB, and the translucency branch has one hidden layer with 30 neurons and an output layer with 1 neuron corresponding to the α value.

We show the more detailed structure of the neural network used by the PDL model in Appendix B.1.1.

For hidden layers, nonlinear activation functions are used in order to enable non-linearity capacity in the model. Specifically, our model uses *leaky Rectified Linear Unit* (leaky-ReLU) [MHN13] activation function $\mathbf{LeakyReLU}_a(x) = \max(ax, x)$ where $a \in \mathbb{R}_+$ is set to be a small value (typically 0.01). The standard ReLU $\mathbf{ReLU}(x) = \max(0, x)$ can be regarded as, loosely, a special case of leaky-ReLU with $a = 0$ except that $a > 0$ (or more specifically, $0 < a \ll 1$) is used by leaky-ReLU. The advantage of leaky-ReLU is it avoids the *dying ReLUs* problem that the standard ReLU suffers from: If the input of a neuron's ReLU activation function, i.e. the weighted summation of all input neurons (Eq. 2.12), becomes negative after a certain training step, the neuron will start outputting 0. Once this occurs, it is unlikely the neuron will come back to "life" (i.e. outputting other values than 0), because the gradient of ReLU is 0 for a negative input, meaning the neuron can not provide any gradient to propagate backward



(a) the PDL model **(b)** the DLLCN model

Figure 3.1.: **(a)** The core of the PDL model is a fully-connected neural network that consists of a "trunk" with hidden layers (shown as part (I) in (a)) to learn generic representations shared across tasks, and two "branches" with hidden layers to learn task-specific representations for two prediction tasks (i.e. the reflectance-predicting task (shown as part (II) in (a)) and the translucency-predicting task (shown as part (III) in (a))), respectively. **(b)** The DLLCN model is a composition of two functions: the first function $\mathbf{DLL} : \mathcal{T} \mapsto \widetilde{\mathcal{T}}$ (shown as part (I) in (b)) distorts the tonal value space aiming to multi-dimensionally linearize the domain of the second function, the cellular Neugebauer model $\mathbf{CN} : \widetilde{\mathcal{T}} \mapsto \mathcal{S} \times \mathcal{A}$ (shown as part (II) in (b)) that predicts both reflectance and translucency.

through lower layers to update weights. The neuron also fails to provide any meaningful information to propagate forward through higher layers. The neuron (or its ReLU activation function) is effectively dropped from the neural network, or "dies" (thus the name "dying ReLUs"). Therefore, to avoid this problem, we use leaky-ReLU for hidden layers. $a = 0.01$ is set in our model.

3.2.2. A horizontally-shifted sigmoid activation function

For the two output layers of the 2 branches, the sigmoid activation function

$$\sigma(x) = \frac{1}{1+\exp(-x)} \tag{3.1}$$

is used to ensure the predicted values of reflectances and α are within $(0,1)$. However, in experiments we observed the sigmoid function excessively outputs near boundary reflectances and α values i.e. lower bound 0.0 or upper bound 1.0, leading to big prediction errors, as shown as the red line in Fig. 3.7. We speculate that during the training the gradient-based parameter update overshoots into the near boundary regime of sigmoid. In this regime, sigmoid is very flat and its derivative

$$\frac{\partial\sigma(x)}{\partial x} = \sigma(x)(1-\sigma(x)) \tag{3.2}$$

vanishes to be too small to update neural network parameters via backpropagation. Once the training overshoots into this "saturation" regime, it's difficult to shoot back to the non-saturating regime due to the aforementioned vanishingly small derivatives [GB10, XHL16]. Such overshooting can happen because of specific strategies of neural network parameter initialization and input feature normalization. For simplicity of analysis, let assume an MLP neural network with L hidden layers as presented in Section 2.1.3. On the one hand, the neural network weights, i.e. $\mathbf{W}^{[i]}$ and $\mathbf{b}^{[i]}$ for each i^{th} hidden layer as well as $\mathbf{W}_{out}$ and $\mathbf{b}_{out}$ for the output layer, are typically initialized to values randomly drawn from a *zero-centered* uniform or Gaussian distribution, e.g. the commonly-used Glorot Normal Initialization [GB10] (which is also used in this thesis) or similar initialization methods [HZRS15]. And on the other hand, the input data is also usually scaled or normalized to be near/centered at zero [ZC18] (e.g. image RGB values in computer vision tasks are commonly scaled to range [0, 1], and tonals in this thesis is also in range [0, 1]). According to Eq. (2.13), the first hidden layer with leaky-ReLU activation function procudes output features $\mathbf{h}^{[1]} = \mathbf{LeakyReLU}(\mathbf{W}^{[1]\top}\mathbf{x} + \mathbf{b}^{[1]})$ near zero due to the near-zero inputs $\mathbf{x}$ and near-zero weights (i.e. $\mathbf{W}^{[1]}$ and $\mathbf{b}^{[1]}$). This near-zero output situation successively happens on all the successive hidden layers, including the final hidden layer i.e. the L^{th} hidden layer.

The near-zero output features from the final hidden layer, i.e. $\mathbf{h}^{[L]}$, are the inputs of the output layer. According to Eq. (2.14), with sigmoid activation function, the output layer produces prediction values $\hat{\mathbf{y}} = \sigma(\mathbf{W}_{out}^{\top}\mathbf{h}^{[L]} + \mathbf{b}_{out})$ around 0.5 due to the near-zero inputs (i.e. $\mathbf{h}^{[L]}$ from the final hidden layer) and near-zero weights (i.e. $\mathbf{W}_{out}$ and $\mathbf{b}_{out}$). Such initial prediction values around 0.5 might be far away from the groundtruth values e.g. 0.8 (Note that an absolute error of 0.3 already counts for 30% in the output range [0.0, 1.0]). Furthermore, the near-zero input of the sigmoid activation function is near

the maximum point $x = 0$ of sigmoid's derivative (Eq. (3.2)). Then the large prediction error together with the large derivative could cause overshooting (e.g. in the direction from an initial prediction value of 0.5 to the groundtruth value of 0.8), surpassing the groundtruth value (e.g. 0.8) and ending up in the aforementioned "saturation" regime (e.g. near the upper bound 1.0) which is difficult to "escape" from even after a number of training steps due to the vanishing gradient. Similarly, this saturation problem can also happen near the lower bound 0.0. Due to this problem, the neural network produces problematic near-boundary predictions, as shown as the red line in Fig. 3.7.

To alleviate this saturation problem, we altered sigmoid by

$$\widetilde{\sigma}(x) = \sigma(x + \sigma^{-1}(\mu_t)) \tag{3.3}$$

where μ_t ($0 < \mu_t < 1$) is the training data's average for the corresponding quantity (reflectance value per wavelength, α). Effectively this horizontally shifts the sigmoid function by an offset of $\sigma^{-1}(\mu_t)$ so that for a near-zero input x we have $\widetilde{\sigma}(x) = \sigma(x + \sigma^{-1}(\mu_t)) \approx \sigma(0 + \sigma^{-1}(\mu_t)) = \sigma(\sigma^{-1}(\mu_t)) = \mu_t$, leading to lower initial prediction error. The maximum point of sigmoid's derivative is also accordingly moved from $x = 0$ to $x = -\sigma^{-1}(\mu_t)$, leading to smaller sigmoid derivative for a near-zero input x. The lower initial prediction error together with the smaller sigmoid derivative reduce the chance of overshooting into sigmoid's saturation regime. As shown as the blue lines in Fig. 3.7 and will be further discussed in the experiment section, the horizontally-shifted sigmoid resolves the issue of aforementioned problematic near-boundary predictions.

Both the regular sigmoid and our proposed horizontally-shifted sigmoid are special cases of a more general *logistic function* $f(x) = \frac{a}{1+\exp(-k(x-x_0))}$ where $a = 1$ and $k = 1$ are set in both these two special cases. While $x_0 = 0$ is set for the regular sigmoid, $x_0 = -\sigma^{-1}(\mu_t)$ is set for the horizontally-shifted sigmoid based on the theoretical analysis of the aforementioned saturation problem and the empirically-observed property of the training data.

3.3. The Deep-Learning-Linearized Cellular Neugebauer (DLLCN) Model

The second deep learning-based model we propose is the *Deep-Learning-Linearized Cellular Neugebauer* (DLLCN) model, which combines deep learning with the traditional cellular Neugebauer model.

Fig. 3.1 (b) shows the structure of the DLLCN model. The DLLCN model is a composition of two functions

$$\mathbf{DLLCN} = \mathbf{CN} \circ \mathbf{DLL} : \mathcal{T} \mapsto \mathcal{S} \times \mathcal{A}. \tag{3.4}$$

The first function $\mathbf{DLL} : \mathcal{T} \mapsto \widetilde{\mathcal{T}}$ (shown as part (I) in Fig. 3.1 (b))) distorts the tonal value space aiming to multi-dimensionally linearize the domain of the second function, the cellular Neugebauer model $\mathbf{CN} : \widetilde{\mathcal{T}} \mapsto \mathcal{S} \times \mathcal{A}$ (shown as part (II) in Fig. 3.1 (b))) that predicts both reflectance and translucency. In this way, the degrees-of-freedom are reduced to a convex combination of Neugebauer cell-primaries ensuring plausible results. We use a deep neural network as the core of **DLL**.

3.3.1. The structure of the linearization function

The core of the **DLL** is a fully-connected neural network $\mathbf{Z} : \mathcal{T} \mapsto \mathbb{R}_+^M$, where M is the dimensionality of the tonal value space $\mathcal{T}$. It has 5 hidden layers with 200, 400, 800, 400, and 200 neurons respectively and an output layer with M neurons. In order to avoid cross-contamination of tonals not present in the input, $\mathbf{Z}$ does not directly distort the tonal value space but computes distortion coefficients of the input tonals $t \in \mathcal{T}$ defining **DLL** as follows:

$$\mathbf{DLL}(t) = \mathbf{C}_{[0,1]}[\mathbf{Z}(t) \odot t], \tag{3.5}$$

where $\odot$ is the Hadamard product (element-wise multiplication) and $\mathbf{C}_{[0,1]}$ is an operator that clips each entry to the range [0,1]. If, for instance, just two tonals corresponding to cyan and magenta materials are used in the input, i.e. tonal entries are all zero except cyan and magenta, the definition of **DLL** in Eq. (3.5) prevents the usage of any Neugebauer cell-primary with yellow or black material contribution because multiplying the zero yellow or black entry with a coefficient weight still yields zero.

All hidden layers use leaky-ReLU activation function, and the output layer uses the linear activation function.

We show the more detailed structure of the neural network used by the DLLCN model in Appendix B.1.1.

3.3.2. Cellular Neugebauer interpolation

The linearized tonals $\widetilde{\mathcal{T}}$ outputted by the **DLL** part are the inputs of the $\mathbf{CN} : \widetilde{\mathcal{T}} \mapsto \mathcal{S} \times \mathcal{A}$ part. The **CN** part is a traditional cellular Neugebauer model that performs just a multi-linear interpolation of the cell primaries for any linearized tonal value. It is canonically extended to predict translucency by simply appending the translucency parameter $\alpha \in \mathcal{A}$ to its cell primary reflectance. In this way, the model can easily be extended to other optical quantities. For resource efficiency we recommend using just three grid points $\{0, 0.5, 1\}^M \subset \mathcal{T}$ for the cellular model.

3.4. Network Design and Learning Strategy

Unless otherwise specified, the design factors (e.g. loss functions) described in this section apply to both the PDL and DLLCN models.

3.4.1. Remarks on the Neural Networks Design

The design of the neural networks for both models share the basic idea that the number of neurons should increase through the first layers to allow higher layers to have more capacity to capture high-level features of the tonal-optical relationship. Towards the output layer the number should decrease

because the number of neurons of the output layer is small and the hidden layers should be able to summarize the features for making final predictions.

We selected the number of layers and total number of neurons so that the neural networks have enough capacity to learn even complex tonal-optical relationships. This design decision is critical because a too large neural network is prone to overfitting given the limited amount of training data. We selected these numbers to be most likely bigger than necessary to predict reflectance and translucency and use the strategies described in the following Sections 3.4.2 and 3.4.3 to avoid overfitting.

3.4.2. Batch Normalization

Batch Normalization (BN) tackles the Internal Covariate Shift (ICS) problem [IS15] where low layers' parameter updates cause unexpected large changes of higher layers' inputs, which greatly breaks the assumption that other layers do not (or only mildly) change when a layer's parameters are updated based on gradients in gradient-based optimization algorithms (e.g. gradient descent and its variants). Batch Normalization zero-centers and normalizes the inputs before each activation function for each layer, using the empirical mean (to do zero-centering) and empirical standard deviation (to do normalization) observed from the mini-batch samples. It then applies a trainable scaling and a trainable offset on the normalized results. This significantly reduces the problem of coordinating updates across layers, thus stabilize the training and accelerate the convergence. Batch Normalization also helps tackle the vanishing/exploding gradients problem, which typically occurs for deep neural networks [GB10]. Finally, it acts like a regularization technique, reducing generalization error [GBC16].

3.4.3. Regularization to avoid overfitting

There are three major regularization techniques used in our approach:

1. Multitask learning [Car97], which allows samples to put more constraints on the part of a model that is shared across multiple tasks [GBC16]. In our proposed models, as shown in Fig. 3.1, Multitask learning is performed by simultaneously optimizing the reflectance-predicting task, the color-predicting task and the translucency-predicting task.
2. Dropout [SHK*14], a very popular regularization technique though working in a rather simple way, temporarily "drops" each neuron with a predefined probability. Each dropped neuron does not contribute to the neural network at all during that training iteration, but may recover (if not dropped again) in the next training iteration. This intuitively force the model to not excessively rely on (thus becomes excessively sensitive to small changes of) only a small amount of neurons, but rather spread the impact across many neurons. From another perspective, the effectiveness of dropout comes from that it's approximately performing bagged ensemble on exponentially many thinned sub-networks (due to random disappearance of neurons) that share parameters [SHK*14, GBC16].

3. Early Stopping, probably the most commonly used form of regularization technique in deep learning [GBC16], is used in our model training. It considers a situation which is often observed in large models [GBC16] that the training error decreases over time, but gradually the error on the validation set starts rising and the model starts overfitting. The training should be terminated when such a situation occurs, and the model parameters should return to the point where the validation loss is minimal. See Section 3.5.5 for more details of our experiment settings.

3.4.4. Loss Function

The loss function serves two purposes:

1. It should ensure a high accuracy of predicting reflectances and translucency.
2. It should ensure a high color accuracy for specified viewing conditions, such as those used by the International Color Consortium (ICC) for defining the profile connection space [ICC10]. These are the CIE D50 illuminant and the CIE 2° observer. For accurate color reproduction under selected viewing conditions this loss is very important because Euclidean distances in spectral space and perceived color differences are not well correlated (see e.g. [Ber00]).

Therefore, our loss function has three parts:

1) Loss for reflectance-predicting task: The reflectance loss is computed using the Root-Mean-Square Error (RMSE) averaged over all the n samples:

$$\mathbf{E}_{\text{ref}} = \frac{1}{n}\sum_{j=1}^{n}\sqrt{\frac{1}{N}\|\hat{\mathbf{r}}^{(j)} - \mathbf{r}^{(j)}\|_2^2} \tag{3.6}$$

where $\hat{\mathbf{r}}^{(j)}, \mathbf{r}^{(j)} \in \mathcal{S}$ are the predicted and the measured reflectances respectively for the j^{th} sample.

2) Loss for color-predicting task: The color loss is computed by the CIEDE2000 formula [CIE01]:

$$\mathbf{E}_{\text{col}} = \frac{1}{n}\sum_{j=1}^{n}\Delta E_{00}(\mathbf{LAB}(\hat{\mathbf{r}}^{(j)}), \mathbf{LAB}(\mathbf{r}^{(j)})) \tag{3.7}$$

where $\mathbf{LAB} : \mathcal{S} \mapsto$ CIELAB is the function that computes (See Section 1.1.2.2) CIELAB from reflectances assuming specified viewing conditions. We use the CIE D50 illuminant and the CIE 2° observer.

3) Loss for translucency-predicting task: The translucency loss is computed as the absolute error:

$$\mathbf{E}_{\text{tra}} = \frac{1}{n}\sum_{j=1}^{n}|\hat{\alpha}^{(j)} - \alpha^{(j)}| \tag{3.8}$$

where $\hat{\alpha}^{(j)}, \alpha^{(j)} \in \mathcal{A}$ are the predicted and the measured translucency α-values [UTB*19] respectively for the j^{th} sample.

Note that α was designed to be nearly perceptually uniform for a set of reference materials with varying absorption and scattering coefficients and isotropic phase function under front/side lit conditions [UTB*19]. Brunton *et al.* used also absolute α-differences to optimize translucency in 3D printing [BATU18].

The final loss function is an weighted summation of the three loss functions:

$$\mathbf{E} = \mathbf{E}_{\text{col}} + a_1 \mathbf{E}_{\text{ref}} + a_2 \mathbf{E}_{\text{tra}} \tag{3.9}$$

where the weights $a_1, a_2 \in \mathbb{R}_+$ are hyper-parameters that should be selected based on the relative magnitudes and importance of the loss functions. A color difference of CIEDE2000 = 1 is just very slightly above the *Just Noticeable Difference* (JND) and almost not detectable in a typical indoor lighting environment even if the colors are directly juxtaposed. A translucency difference of $\alpha = 0.1$ is also slightly above the JND for translucency [UTB*19]. Hence, to balance suprathreshold color and translucency differences very close to the JND in Eq. (4.10), we have to set $a_2 = 10$. Tsutsumi *et al.* balanced CIEDE2000 and spectral RMSE linearly for spectral gamut mapping and found a value of $a_1 = 50$ to be optimal [TRB08]. Their results show however that for $a_1 \in [20, 50]$ spectral and color accuracy is almost constant. An independent hyper-parameter tuning based on the validation data described in the experiments showed that $a_1 = 20$ and $a_2 = 10$ result in mean CIEDE2000 and α-errors close to JND with small spectral RMSE. We therefore recommend these values to achieve a good balance between color, spectral and translucency losses.

3.5. Experiments and Results

The purpose of the experiments is to evaluate the accuracy and resource efficiency of both models and to compare this with the traditional cellular Neugebauer (CN) model used, for instance, by Brunton *et al.* [BATU18] to reproduce jointly color and translucency in 3D printing. Results and learning procedure can be reproduced as we show in Code File 1 (Ref. [CU]).

3.5.1. 3D Printers and data sets

We have evaluated both models on two state-of-the-art material-jetting 3D-printers employing six materials: the Stratasys J750 and the Mimaki 3DUJ-553. Both printers use Cyan (C), Magenta (M), Yellow (Y), Black (K), White (W), Transparent (T) build materials and one support material. The Mimaki 3DUJ-553 printer was equipped with the MH100 CMYKWCl materials, and the Stratasys J750 with Vivid CMY, Vero KCl and VeroPureWhite. The tonal value space $\mathcal{T} = \text{CMYKT}$ has five dimensions, where the final dimension $\text{T} \in [0, 1]$ corresponds to the ratio of clear and white materials within the unit volume not occupied by the CMYK materials. The smaller T the larger is the fraction of white material. Very different halftoning algorithms were used for the two printers taking into account different

Table 3.1.: Dataset for Mimaki 1 (grid point locations)

	T=0	T=245	T=253	T=254	T=255
C	0,62,119,159,191,221,255	0,137,199,238,255	0,143,210,239,255	0,137,206,240,255	0,141,212,246,255
M	0,51,97,140,177,221,255	0,120,200,236,255	0,137,205,240,255	0,134,205,241,255	0,126,195,234,255
Y	0,90,115,128,148,171,255	0,172,224,244,255	0,184,225,244,255	0,184,236,250,255	0,161,224,250,255
K	0,75,142,170,203,230,255	0,200,237,249,255	0,193,241,252,255	0,190,241,253,255	0,155,236,252,255

material opacities, absorption levels, droplet sizes and droplet positioning capabilities of the machines, resulting in very different material arrangements and optical properties for the same tonal values. It is worth mentioning that because of the high opacity and high absorption of the Mimaki prints the halftoning algorithm always adds some amount of clear material to avoid graininess and that this clear material has a slightly yellow tinge.

For the transformation $\mathbf{Q}$ introduced in Section 1.1.3 we used a two-dimensional transformation per CMYK-tonal described in [BATU18] and for T the identity.

Data was collected by printing flat targets with patches each corresponding to a specific tonal value. Reflectances were measured in 45/0 geometry with a Barbieri Spectro LFP qb spectrophotometer equipped with a 19mm aperture of illumination and 2mm of detection to avoid edge loss for translucent materials. The translucency parameter α was measured with a Barbieri Spectro LFP according to [UTB*19]. The tonal values were selected for the initial purpose of fitting and investigating the performance of different cellular Neugebauer models/sub-models, which is the reason why the tonal value sets (reported here as 8-bit values) differ between printers.

Dataset for the Stratasys J750: We used a regular grid of tonal values $\{0,64,128,191,255\}^5 \subset$ CMYKT to print and measure 3125 patches in total.

Dataset for the Mimaki 3DUJ-553: The data was created by two Mimaki 3DUJ-553 printers with slightly different settings (firmware and post process) and will be therefore treated separately. We will refer to these printers as Mimaki 1 and Mimaki 2.

Mimaki 1: Samples were collected at 7 regular grid points in each of the CMYK dimensions for $T = 0$, and five regular grid points in each of the CMYK dimensions for T=245, 253, 254 and 255 respectively (see Table 3.1 for the grid point locations). This leads to $7^4 + 5^4 * 4 = 4901$ patches in total. Since the samples do not form a regular grid, the CN and DLLCN can't be used and only the performance of the PDL model is evaluated.

Mimaki 2: Only opaque samples corresponding to $T = 0$ were collected. We used a regular grid $\{0,43,85,128,170,213,255\}^4 \subset$ CMYK of tonal values. Another 1099 CMYK random samples were also collected. Together, $7^4 + 1099 = 3500$ patches were printed.

Table 3.2.: Number of samples used for training ($|\mathcal{Q}|, |\mathcal{P}|$ is the cardinality of $\mathcal{Q}$ and $\mathcal{P}$)

Printer	$\|\mathcal{Q}\|$	$\|\mathcal{P}\|$ for DLLCN and PDL	$\|\mathcal{P}\|$ for CN
Stratasys	3125	300, 400, 576, 1024, 1280, 1600, 2000, 2500 only PDL: 243, 350; only DLLCN: 500, 700	243, 576, 768, 1024, 1280, 1600, 2000, 2500
Mimaki 1	4901	200, 700, 1200, 1700, 2200, 2700	/
Mimaki 2	3500	500, 800, 1296, 1700, 2058, 2401 only PDL: 256; only DLLCN: 300	256, 500, 750, 1296, 2058, 2401

3.5.2. Training and test data

We split for each printer the tonal value set $\mathcal{Q} \subset \mathcal{T}$ of printed and measured samples (see Section 3.5.1) into two sets: The training/validation set $\mathcal{P}$ and the test set $\mathcal{E}$ with $\mathcal{P} \cap \mathcal{E} = \emptyset$. $\mathcal{P}$ contains always the set $\mathcal{P}_0$ of the necessary training samples of each model, i.e. the cellular Neugebauer primaries for the DLLCN (see Sec. 3.3.2) and the Neugebauer primaries for the PDL model. We first select 300 samples for $\mathcal{E} \subset \mathcal{Q} \setminus \mathcal{P}_0$. For the Stratasys and Mimaki 1 printers $\mathcal{Q} \setminus \mathcal{P}_0$ was split into 5 sets each corresponding to one of the five T-levels and 60 samples were selected randomly from each of these sets for $\mathcal{E}$. For the Mimaki 2 printer $\mathcal{E}$ was selected randomly from the 1099 random samples. Note that for all tonals in $\mathcal{Q}$, printed and measured reflectances and α-values are available (*groundtruth*), except for the "Mimaki 2" dataset where all samples are opaque and thus translucency α-measurements were not collected.

In order to investigate the influence of the number of training samples on model performance, samples in $\mathcal{P}$ are increased starting with $\mathcal{P} = \mathcal{P}_0$ as follows: $\mathcal{P}_{i+1} := \{t_i\} \cup \mathcal{P}_i$, where the tonal value t_i is randomly selected from $\mathcal{Q} \setminus \{\mathcal{E} \cup \mathcal{P}_i\}$ (Note that the sampling is always without replacement). For the Mimaki 1 printer the new tonal value t_i is randomly selected from the set $\{(C,M,Y,K,T) \in \mathcal{Q} \setminus \{\mathcal{E} \cup \mathcal{P}_i\} \mid T = \mathrm{T}_{(i \bmod 5)}\}$. This cyclic sampling is to avoid sampling bias due to the imbalanced sample distribution over the T levels. For the Mimaki 2 printer, t_i is first randomly selected from the 1099 random samples not included in $\mathcal{E} \cup \mathcal{P}_i$. Later on, if all these 1099 samples are included in $\mathcal{E} \cup \mathcal{P}_i$, t_i is selected randomly from $\mathcal{Q} \setminus \{\mathcal{E} \cup \mathcal{P}_i\}$.

The traditional CN model requires a regular grid. To allow a comparison with a smaller amount of data, we used sub-grids as separately detailed in Appendix B.1.2. For the non-aligned data of the Mimaki 1 printer, the CN and DLLCN models cannot be evaluated. Table 3.2 column 3 summarizes the varying numbers of training samples for all three models.

3.5.3. Computing and evaluating predictions

For training the DLLCN and PDL models, $\mathcal{P}$ is split into two disjoint sets, the training set $\mathcal{P}_\mathrm{t}$ and the validation set $\mathcal{P}_\mathrm{v}$, with $\mathcal{P} = \mathcal{P}_\mathrm{t} \cup \mathcal{P}_\mathrm{v}$ and $\mathcal{P}_0 \subset \mathcal{P}_\mathrm{t}$. $\mathcal{P}_\mathrm{t}$ is used for training the model and the validation set $\mathcal{P}_\mathrm{v}$ is used for hyper-parameter training (e.g. for early stopping). $\mathcal{P}_\mathrm{v}$ consists of 5% randomly selected samples from $\mathcal{P} \setminus \mathcal{P}_0$. The random splitting of $\mathcal{P}$ into $\mathcal{P}_\mathrm{t}$ and $\mathcal{P}_\mathrm{v}$ is performed 10 times resulting in 10 slightly differently trained models. Model predictions (reflectance, α) of the 10 models for $\mathcal{E}$ are averaged and used as the final prediction to evaluate the performance of the model trained on $\mathcal{P}$. Results

of such averaged predictions are less biased by a single specific splitting of $\mathcal{P}$. Note that $\mathcal{P} \cap \mathcal{E} = \emptyset$, hence the test set $\mathcal{E}$ is always unseen during the training.

3.5.4. Software and hardware setup

Our model is implemented with TensorFlow 2.2.0, and is trained on an NVIDIA GeForce RTX 2080 SUPER GPU on a PC that has 64 GB RAM memory and an Intel(R) Core(TM) i9-9900K CPU (3.6GHz). Training a model takes roughly 120s with 500 training samples, and roughly 290s with 2500 training samples. Predicting on 300 samples takes roughly 0.03s.

3.5.5. Training method

Glorot normal initializer [GB10] is used to initialize neural network weights. Dropout rate is set to 0.1 for PDL, and 0.05 for DLLCN. Adam optimizer [KB14] is used to train the models. The number of training epochs is 15,000. Batch size is equal to the size of the training set. The initial learning rate is 0.009. Learning rate decay is adopted, specifically, the learning rate is divided by a factor of 3 if the loss on validation set has not gone down for 1000 epochs. Early stopping is used during the training, specifically, the training is terminated if the loss on validation set has not gone down for 2000 epochs, and the model is rolled back to the point where the loss on validation set was minimal.

3.5.6. Results for the Stratasys printer

The benefit of using the deep learning-based models compared to the CN model can be seen by looking at accuracy vs. dataset size: To achieve a similar mean CIEDE2000 accuracy (see Fig. 3.2 (a)) of the best result of the CN approach, which is CIEDE2000 = 2.4 corresponding to 2500 samples, the DLLCN approach only requires 500 samples which is a 5.0 times improvement in data efficiency, and the PDL approach further reduces the required number of samples to 300 which is an 8.3 times improvement. If all approaches use 2500 samples, the DLLCN achieves CIEDE2000 = 0.9 and the PDL approach CIEDE2000 = 0.4 as the mean prediction error, which are a 2.7 and a 6 times improvements in accuracy compared to the CN model, respectively. We see a similar trend for the 90^{th} percentile (see Fig. 3.2 (b)), which is CIEDE2000 = 4 for the CN model at 2500 training samples and almost 2.2 times as big as for the DLLCN model that has CIEDE2000 = 1.8. The PDL approach achieves even CIEDE2000 = 0.7. To achieve a similar 90^{th} percentile accuracy as the CN model for 2500 samples, the DLLCN model requires only 700 and the PDL model 350 samples, which are an approx. 3.6 and a 7.1 times improvements.

Mean and 90^{th} percentile spectral RMSE as well as α errors follow the same trend as shown in Fig. 3.2 (c)-(f). Note that the 90^{th} percentile α-errors of the DLLCN and PDL model are below the just noticeable α-difference of approx. 0.1 for any considered training/validation set $\mathcal{P}$ (except for $|\mathcal{P}| = 300$ for DLLCN), indicating a sufficient prediction accuracy for nearly all applications. The CN model needs 768 training samples to pass this threshold.

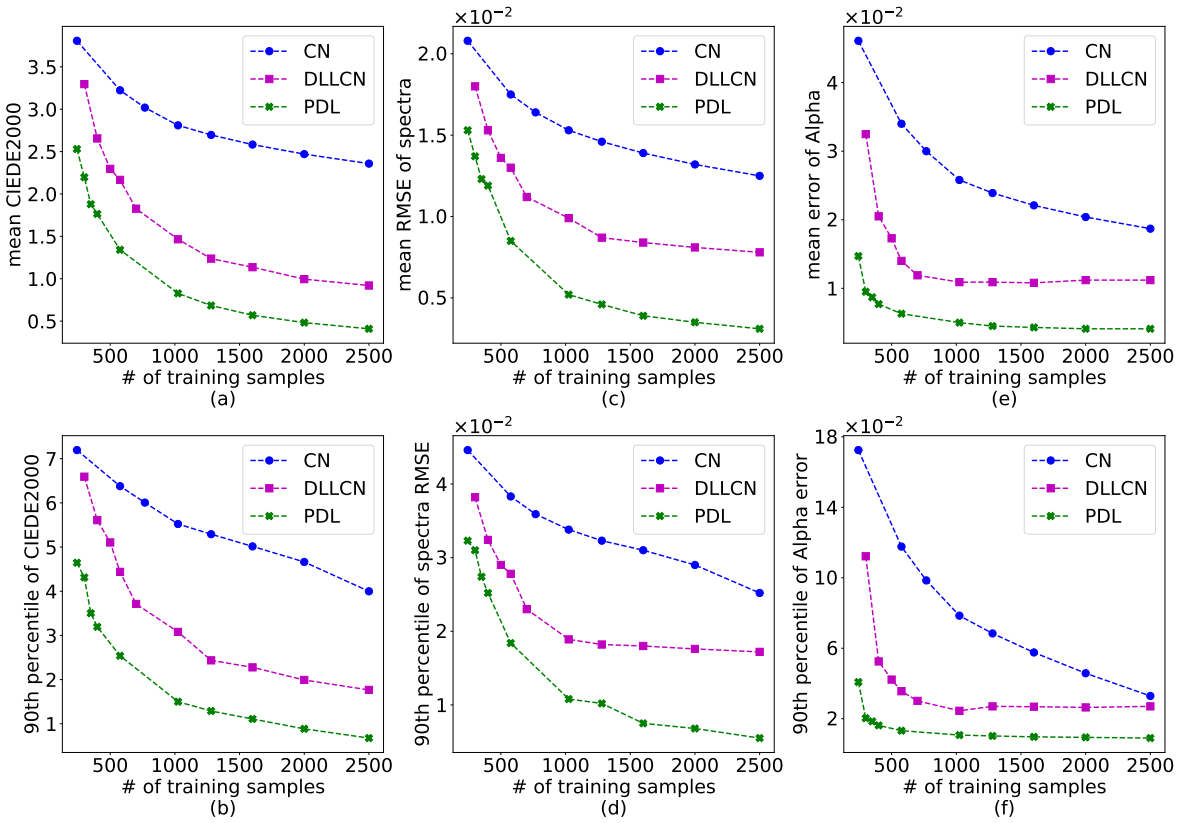


Figure 3.2.: Accuracy vs. dataset size plots for the Stratasys printer. The first row shows the mean prediction errors in color (the first column), spectral (the second column), and translucency (the third column), respectively. The second row shows the 90$^{\text{th}}$ percentile of errors.

3.5.7. Results for the Mimaki 1 printer

For the Mimaki 1 printer only the PDL model can be evaluated as described in Sec. 3.5.1. Due to the highly opaque white material and the resulting halftone method that uses almost always a fraction of clear material to avoid graininess, the relationship between tonals and optical properties of the resulting print is much more complex than for the Stratasys printer. Even though the accuracy vs. dataset size curves in Fig. 3.3 show the same trend as for the Stratasys printer, the accuracy of the PDL model is worse than on the Stratasys printer. For the Mimaki 1 printer, 700 training samples are required to reach an average CIEDE2000 error of 2 and an 90$^{\text{th}}$ percentile error of 4. For the Stratasys printer this accuracy was reached with only 350 training samples. The α prediction is still below JND even for $|\mathcal{P}| = 200$, the smallest number of training samples.

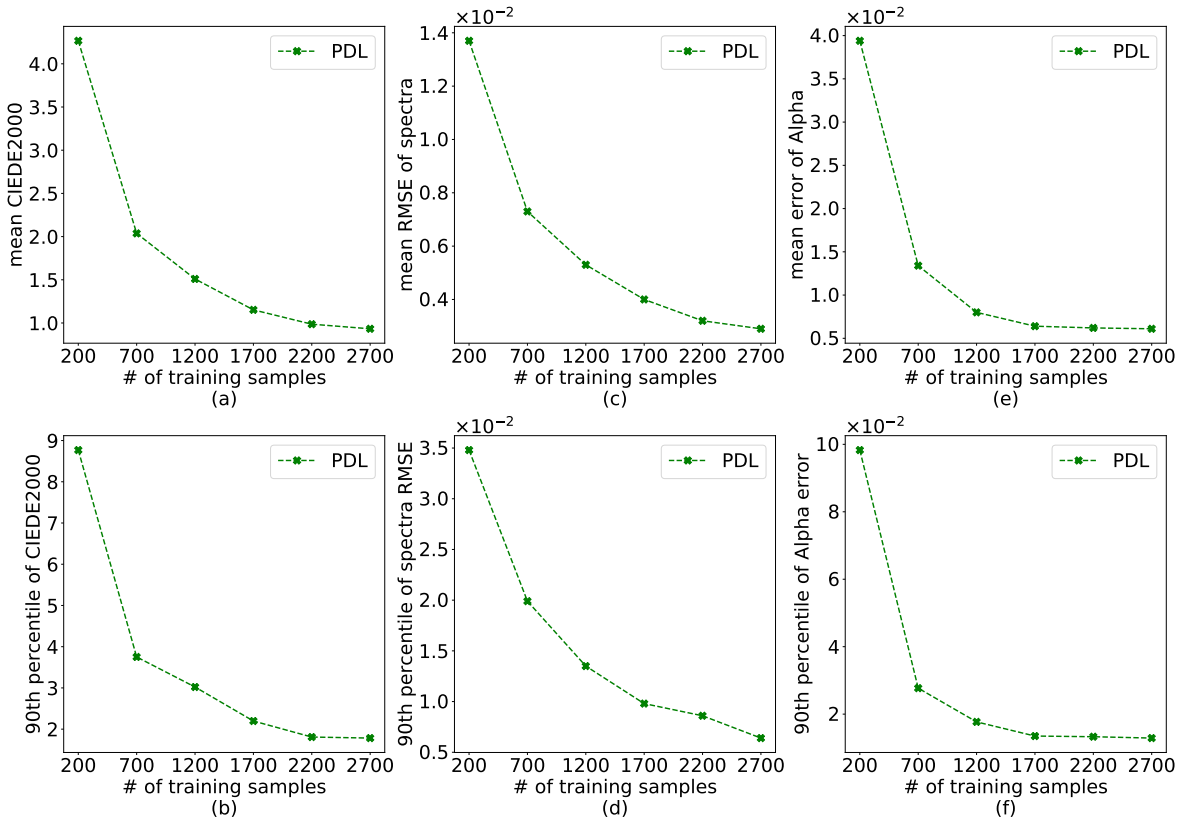


Figure 3.3.: Accuracy vs. dataset size plots for the Mimaki 1 printer. Note that for the Mimaki 1 printer only the PDL model can be evaluated due to the non-aligned data as described in Sec. 3.5.1.

3.5.8. Results for the Mimaki 2 printer

For the Mimaki 2 printer the same halftone method was used as for the Mimaki 1 printer, i.e. even though only opaque samples (T = 0) are considered the yellow-tinged clear material is used. In Fig. 3.4, similar as for the other printers, we observe diminishing returns of improvement for the DLLCN and PDL model as the training set size increases, i.e. in the beginning error rates drop strongly but the curve flattens for more training samples.

As shown in Fig. 3.4 (a), the DLLCN model needs just 800 samples and the PDL model only 256 samples to reach a similar average color accuracy of CIEDE2000 = 1.9 as the CN model using 2401 samples. These are 3 and 9.4 times improvements in resources efficiency.

An interesting observation is that for the DLLCN model the spectral RMSE drops to a plateau and becomes even worse than for the CN model if more than 1296 training samples are used (See Fig. 3.4 (c), (d)). This is because the degrees-of-freedom of the DLLCN model is limited to the spectral

space spanned by the Neugebauer cell-primaries. Due to the contribution of the yellow-tinged clear material by the halftoning, some reflectances cannot be described by the linearized cellular Neugebauer model. We tested also a DLLCN model with four grid points (4GP) increasing the model's degrees-of-freedom to reproduce reflectances. It can be seen that even for the same number of samples the spectral RMSE is much smaller for the 4GP DLLCN model confirming the above statement. Interestingly, average color errors are not as much affected by the Neugebauer cell-primaries and both DLLCN models show similar performance (see Fig. 3.4(a)), indicating the benefit of optimizing models w.r.t. to both color and reflectances via multi-task learning. In contrast to the 4GP DLLCN model however, small improvements in average color accuracy of the 3GP DLLCN model for larger training data are at the expense of increased 90^{th} percentile errors due to the model's limited degrees-of-freedom (see Fig. 3.4(b)).

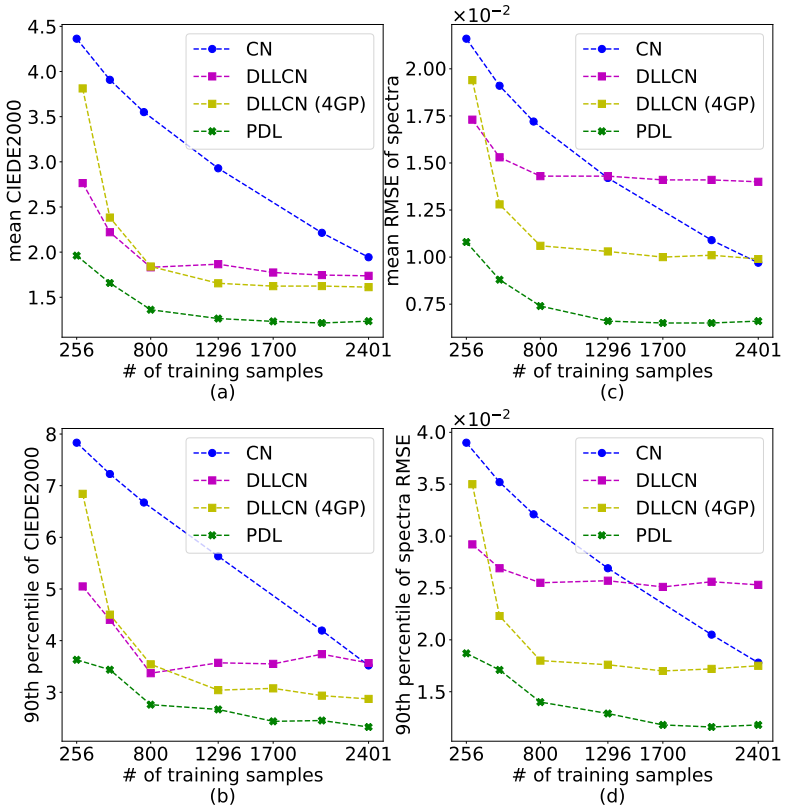


Figure 3.4.: Accuracy vs. dataset size plots for the Mimaki 2 printer. Note that translucency α measurements are not available for Mimaki 2, thus there are no translucency α plots.

3.5.9. Visualization of spectral predictions

In Fig. 3.5, we visualize the spectral predictions corresponding to the median and 99th percentile spectral RMSE error. For the latter case, the sRGB colors of its ground truth (GT) and its prediction are also shown. The figure shows that even using small training data the spectral predictions are quite consistent with the ground truths, and that the DLLCN and especially PDL models outperform the CN model.

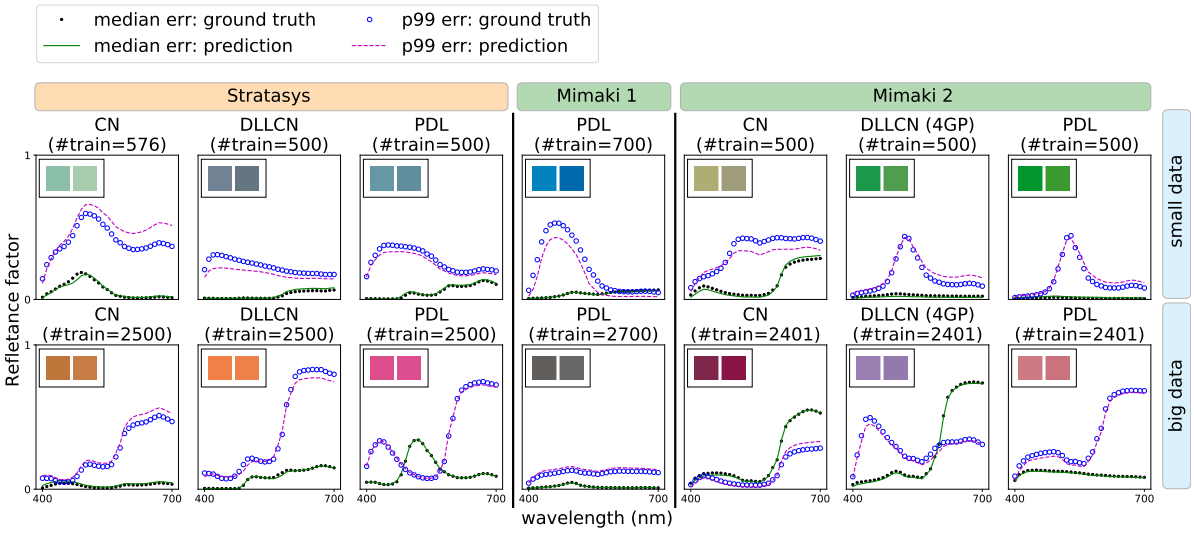


Figure 3.5.: Spectral predictions using different training data sizes.

3.5.10. Benefits of using the horizontally-shifted sigmoid

Fig. 3.6 shows the plots of accuracy vs. dataset size for the Stratasys printer, with and without horizontally-shifted sigmoid defined by Eq. (3.3). It shows that the horizontally-shifted sigmoid leads to similar median errors as the regular sigmoid, but much lower maximal errors.

Fig. 3.7 visualizes the spectra predictions of 3 samples, with the sRGB colors shown for the ground truth, the predicted spectra from the horizontally-shifted sigmoid, and the predicted spectra from the regular sigmoid, respectively. It shows that the regular sigmoid leads to problematic near-boundary predictions for some samples where all the spectra prediction values are either almost the upper bound 1.0 or almost the lower bound 0.0 due to the saturation problem of sigmoid as discussed in Section 3.2.2. The horizontally-shifted sigmoid, in contrast, avoids the problematic near-boundary predictions. This is because, as discussed in Section 3.2.2, the horizontally-shifted sigmoid reduces the initial prediction error and avoids the large gradient, thus reduces the chance of overshooting into sigmoid's saturation regime.

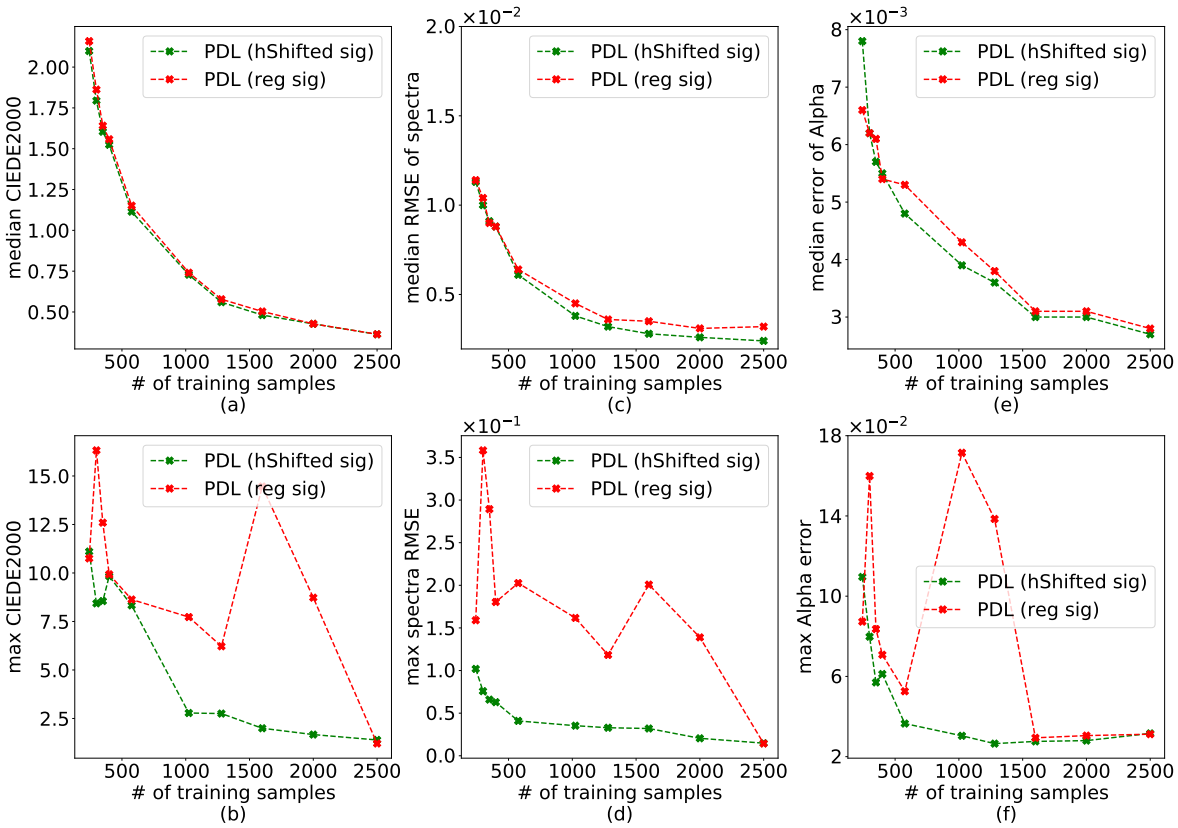


Figure 3.6.: Accuracy (median and max error) vs. dataset size plots for the Stratasys printer, with and without horizontally-shifted sigmoid.

3.6. Limitations

With the cellular Neugebauer approach as a part, the DLLCN model still requires a set of cellular Neugebauer primaries that grows exponentially w.r.t. the number of printing materials, though not as large as required by the standard cellular Neugebauer model that requires much denser regular grids. Although DLLCN already outperforms the standard cellular Neugebauer model by a large margin in accuracy, the more accurate PDL would be recommended when prediction accuracy is of higher importance.

3.7. Conclusion

For optically characterizing multi-material 3D printers, we proposed two deep learning approaches, the *Deep-Learning-Linearized Cellular Neugebauer* (DLLCN) approach and the *Pure Deep Learning*

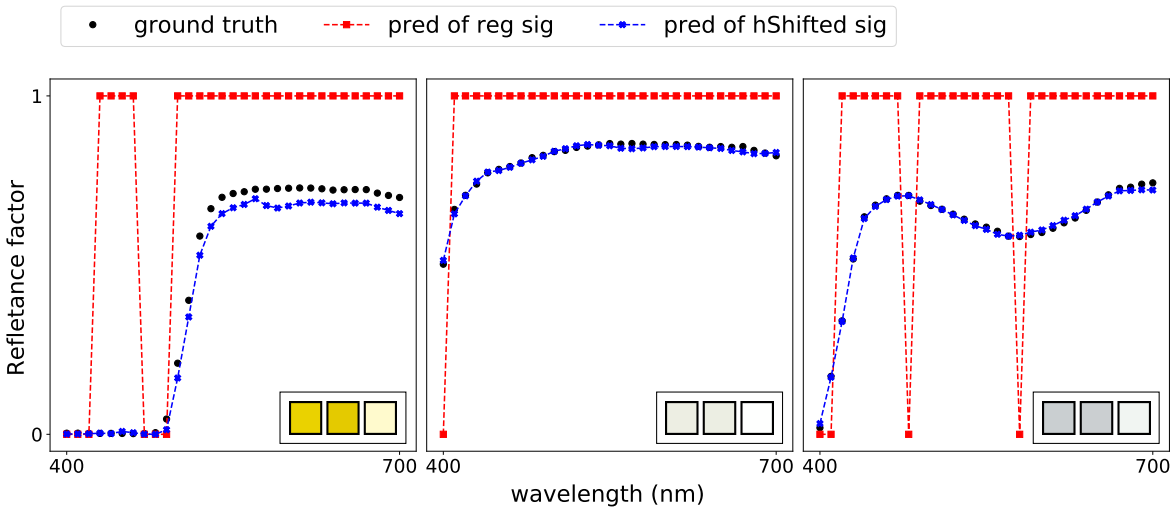


Figure 3.7.: Spectra predictions of several cases where horizontally-shifted sigmoid avoids problematic near-boundary predictions resulted from the regular sigmoid

(PDL) approach, and presented a learning strategy to avoid overfitting. Both approaches outperform the standard cellular Neugebauer approach by large margins in terms of accuracy and data efficiency in predicting joint color and translucency validated on two state-of-the-art 6-material 3D printers. For the Stratasys J750 3D printer, they require up to 8.3/10.3 times fewer samples to achieve similar accuracy in color/translucency (α-value), or are up to 6/4.6 times more accurate when using the same amount of samples. For the Mimaki 3DUJ-553 printer, possessing a very complex tonal-optical relationship due to yellow-tinged clear material contribution in the 3D-halftoning, they need up to 9.4 times fewer samples to reach a similar color accuracy. Note that exactly the same model structures were used for both printers. Weights and hyperparameters were automatically determined by the described learning strategy.

The proposed deep-learning-based models possess a few intrinsic advantages compared to other phenomenological models. First, to achieve a sufficient accuracy in most applications they do not require dense regular grids as required by the cellular Neugebauer model. Second, they can be updated with more data that is *scattered* instead of gridded, which is interesting for improving prints in particular optical regions of interest, e.g. colors and translucencies of prosthetic implants (teeth, eyes). Furthermore, they learn (in an end-to-end manner from the tonal inputs to the final optical quantity outputs) the impact of material cross-contamination and post-process treatment (e.g. polishing, coating), both of which are difficult to measure and to consider by RTE-based models.

In this chapter, we have answered *RQ1* (i.e. *How to leverage deep learning to achieve* ***high accuracies*** *of optical printer models?*) by proposing two deep learning-based optical printer models that both achieve high accuracies with a moderate number of required training samples.

In the next chapter, we will answer the second research question *RQ2*:
*How to improve the **robustness and plausibility** of deep learning-based optical printer models?*

4. Inducing robustness and plausibility in deep learning optical 3D printer models

In the previous chapter, we answered *RQ1* (i.e. *How to leverage deep learning to achieve* ***high accuracies*** *of optical printer models?*) by proposing two deep learning-based optical printer models, i.e. the PDL model and the DLLCN model, which both achieve high accuracies in predicting optical or visual quantities from material arrangements. Due to larger degrees-of-freedom combined with learning strategies reducing overfitting, the PDL model is more accurate than the DLLCN model.

However, since a purely empirical deep-learning-based approach is essentially a black-box without considering any physical grounding, it is sensitive to outliers or noise of the training data and tends to create physically-implausible tonal-to-optical relationships. In this chapter, we will answer *RQ2* (i.e. *How to improve the* ***robustness and plausibility*** *of deep learning-based optical printer models?*). To this end, we propose a methodology to narrow down the degrees-of-freedom of deep learning-based optical printer models by inducing physically-plausible constraints and smoothness. We use this approach to introduce the *Robust Plausible Deep Learning* (RPDL) optical printer model enhancing robustness to erroneous and noisy training data, as well as physical plausibility of the PDL model for selected tonal-to-optical monotonicity relationships.

Experiments show that the RPDL model not only almost always corrects implausible tonal-to-optical relationships, but also ensures significantly smoother predictions, without sacrificing accuracy. On small training data, it even outperforms the PDL model in accuracy by up to 8%, indicating a better generalization ability.

This chapter is based on the published paper [CU22].

4.1. Introduction

An optical printer model is a predictive function predicting a print's optical properties given the arrangement or ratio of printing materials. An accurate optical printer model is a prerequisite to accurately reproduce color, translucency, or joint color and translucency in multi-material 3D printing.

In the previous chapter we proposed two deep-learning models to optically characterize multimaterial 3D printing systems: First, the *Pure Deep Learning* (PDL) model [CU21] that does not rely on any physical grounding; Second, the *Deep-Learning-Linearized Cellular Neugebauer* (DLLCN) model [CU21] that uses deep learning to multi-dimensionally linearize the tonal-value-space of a cellular Neugebauer model. Both models achieve high accuracies with a moderate number of training

Figure 4.1.: Lightness L^* as a function of black material K with other materials fixed on a tonal case of the *Mimaki 2* dataset (the fixed tonal values are shown at the lower left of the figure), and the resulting colors. The L^* values are shown as white numerical text, with italic font indicating L^*-vs-K monotonicity violations. The PDL/RPDL prediction in this figure is from a single PDL/RPDL model respectively. The figure shows that the training data itself has errors leading to monotonicity violations, and that the PDL model overfits to the erroneous data without considering monotonicity. In contrast, the proposed RPDL model ensures monotonicity despite the data outlier, indicating better robustness against errors in training data.

prints. Due to larger degrees-of-freedom combined with learning strategies reducing overfitting, the PDL model is more accurate than the DLLCN model w.r.t. spectral, color and translucency errors.

However, a shortcoming of a purely empirical deep-learning-based approach is that it does not consider physical/perceptual knowledge of relationships between material ratios (or tonals) and the resulting optical/visual properties. This results in implausible tonal-to-optical predictions. One way to ensure physical plausibility is using a deep learning model to adjust the parameters of a physical or partly physical model such as proposed for the DLLCN model. Unfortunately, physical models may lack the degrees-of-freedom to accurately consider all influencing factors, such as complex physical material mixing (e.g. between support and build materials at the object's surface) or post-process treatment, which likely adversely impact their prediction performance.

We show how to induce physically-based heuristics into purely empirical models. A plausible heuristic is, for instance, that the print's reflectance factor (or lightness) does not increase with an increasing fraction of black material assuming that the black material has the maximum absorption of all available printing materials. Such monotonicity relationship between material ratios and measurable optical quantities applies also for translucency: Increasing the fraction of transparent material (a material with negligible absorption and scattering) in the material mixture does not decrease the translucency α-value [UTB*19] of the resulting print.

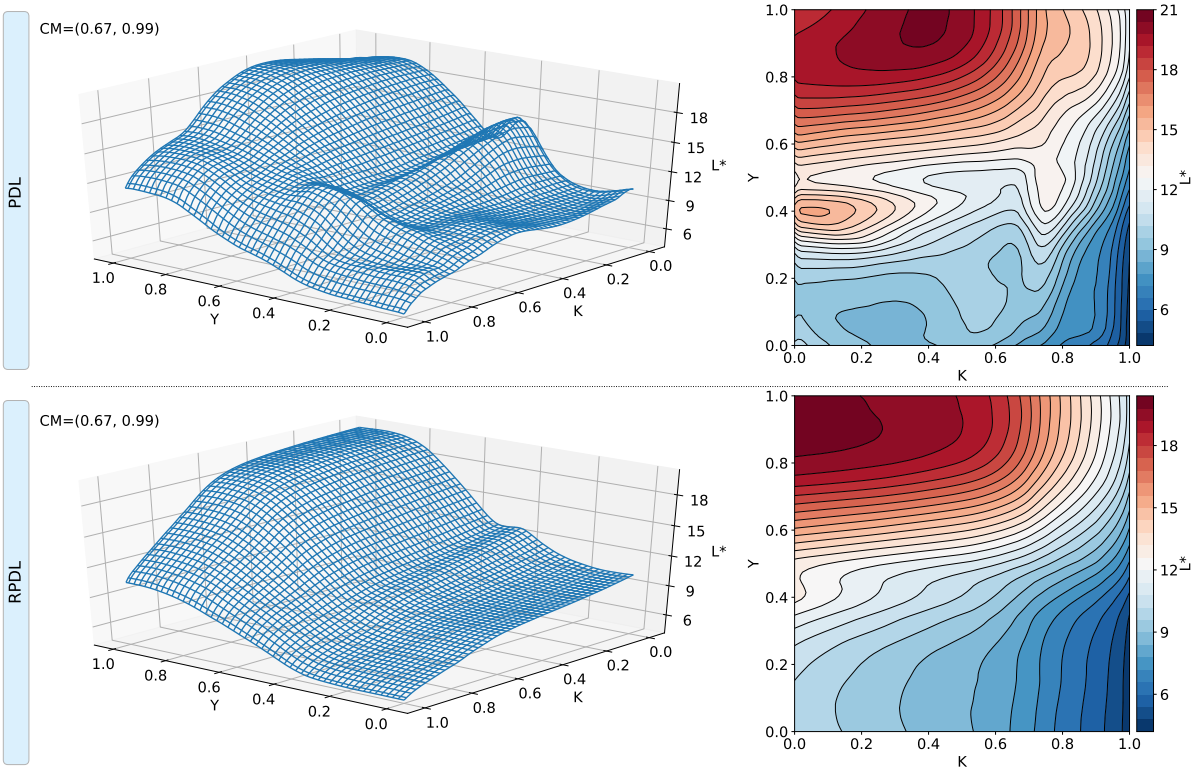


Figure 4.2.: The 3D plot and contour plot of L^* as a function of Y and K with other materials fixed on a tonal case of the "Mimaki 2" dataset (The fixed values are shown at the upper left corner). The first row corresponds to the PDL model and the second to the proposed RPDL model. Notice the skewed curves and isolated "islands" in the contour plot for the PDL predictions. Drawing a profile across such a contour island along the K direction will result in a bumpy L^*-vs-K curve violating monotonicity. In contrast, the RPDL predictions do not possess such islands and show much smoother contour curves ensuring monotonicity, indicating better robustness and plausibility.

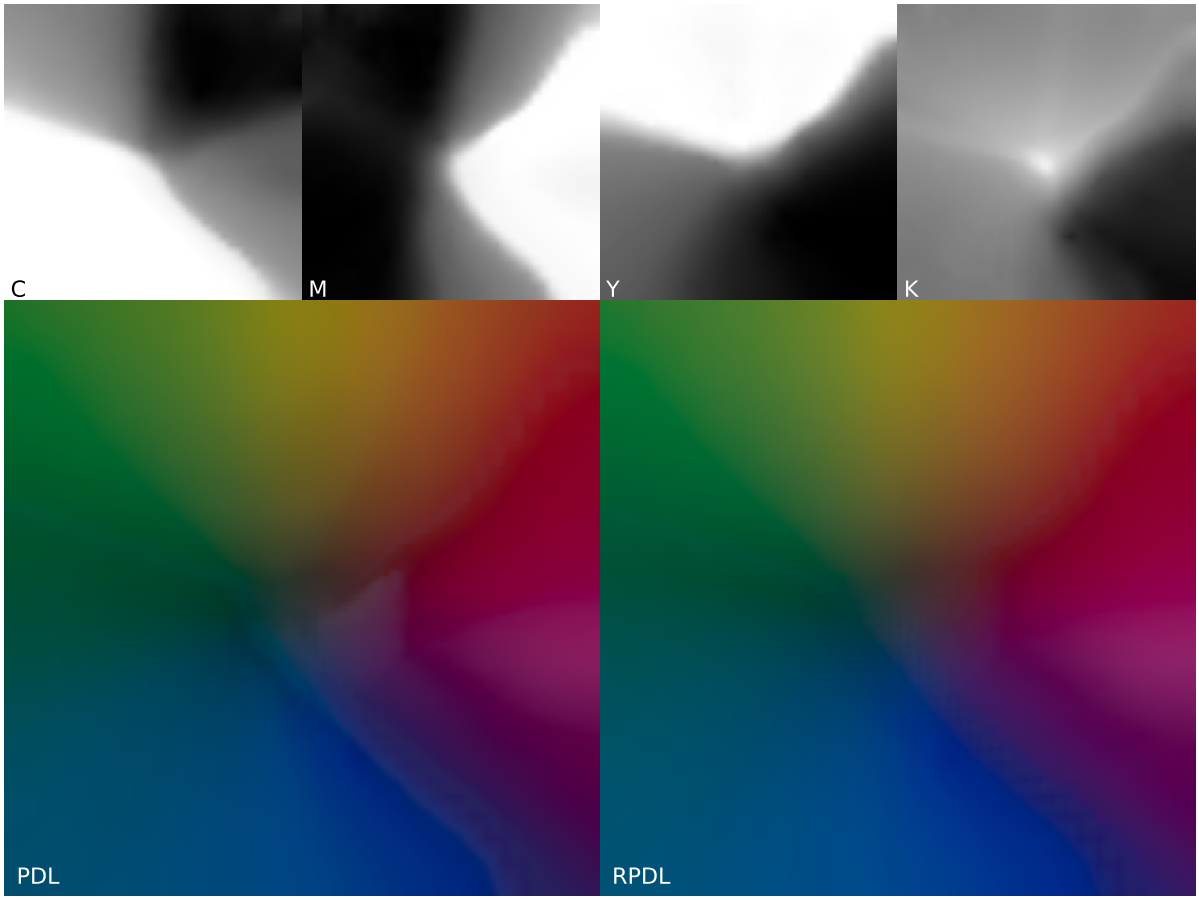


Figure 4.3.: C, M, Y, and K separations of the gamut mapped CIE-a^*b^* plane for CIE-$L^* = 20$ for the Mimaki 3DUJ-553 3D printer based on the RPDL model and the color predictions of the PDL and RPDL models.

Another issue of any optical characterization process is the quality and plausibility of the data to fit or train the model. Printing and measurement errors may cause implausible training data violating the heuristics mentioned above. Fig. 4.1 shows an example where the training data has errors leading to violations in the monotonic relationship between lightness and black material usage. Printer variability (spatial, temporal) also induces noise into the training data, aggravating the challenge of noisy predictions from optical printer models. Fig. 4.2 shows an example where a purely empirical PDL model has bumpy and implausible predictions. For reproducing a distinct optical quantity, an optical printer model needs to be inverted using constraint optimization to obtain the corresponding material ratios or tonals [TRB06, UG06, URB07, UR07] – this process is called *separation*. Due to local non-monotonicity caused by noisy training data, such inversions will cause banding artifacts in gradients [MAA*08] reducing the print quality [ZNS*12]. A relatively simple solution to reduce noise and outliers in the training data is printing and measuring the same sample multiple times and only consider the median for training. This reduces data noise but increases data collection efforts by a multiple.

Fig. 4.3 shows a smooth C,M,Y,K separation example, which would result in an artifact-free physical printout. In contrast, the PDL prediction shows banding artifacts that would not appear in the physical printout. This indicates that a separation (i.e. inversion) based on the PDL model possesses such artifacts as well, which will be reflected in physical printouts and is unacceptable for color critical applications in which texture detail preservation is crucial, such as for reproducing prosthetic eyes [Fraa] that must match the companion (healthy) eye.

We propose a methodology to narrow down the degrees-of-freedom of deep-learning based optical printer models by inducing physically-plausible constraints and smoothness. Our methodology does not need any additional printed samples for training. We use this approach to introduce the Robust Plausible Deep Learning (RPDL) optical printer model enhancing robustness to erroneous and noisy training data as well as physical plausibility of the PDL model for selected tonal-to-optical relationships. In particular, we make the following contributions:

1. We introduce a learning strategy to induce monotonicity heuristics into the PDL model by proposing a new derivative-based loss function that is evaluated in the training process by random tonal value re-sampling.
2. We select physical plausible monotonicity relationships between lightness (CIE L^*) and black material, as well as between translucency (α-value) and transparent material.
3. To make the PDL model more robust to noisy input data, we induce a smoothness heuristic of the tonal-to-optical relationship by a new second-derivative-based loss function that is evaluated during training by random tonal value re-sampling.
4. We propose an automatic hyper-parameter optimization strategy to combine the new loss functions and PDL's original loss functions considering an upper threshold for color accuracy losses.

We show on four datasets from state-of-the-art multimaterial 3D printers that the proposed strategy improves model plausibility and robustness without sacrificing accuracy. In our experiments, the RPDL models almost always do not show violations of the two induced monotonicity constraints, which is a

prerequisite for banding artifact-free separations and as a consequence also artifact-free physical printouts. This is crucial for color-critical applications (e.g. 3D printed prosthetic eyes) where preserving texture details is important.

4.2. The Robust Plausible Deep Learning (RPDL) optical printer model

As described in Chapter 3, the PDL model is a function $\mathbf{PDL} : \mathcal{T} \mapsto \mathcal{S} \times \mathcal{A}$ predicting spectral reflectances $\mathbf{r} \in \mathcal{S} = [0,1]^N$ and translucency $\alpha \in \mathcal{A} = [0,1]$ from tonal values $\mathbf{t} \in \mathcal{T} = [0,1]^M$. The proposed RPDL model is also a function $\mathcal{T} \mapsto \mathcal{S} \times \mathcal{A}$. However, the RPDL model is regularized by physically-plausible constraints and smoothness heuristics, leading to significantly improved robustness and smoothness. These will be detailed in the following sub-sections, respectively.

4.2.1. Injecting prior knowledge of monotonic relationships

We first conceptually point out two monotonic relationships between tonal values and resulting visual quantities: 1) Increasing the fraction of black material in a material mixture does not increase lightness CIE-L^* , and 2) Increasing the fraction of transparent material (a material with almost zero absorption and scattering) in a material mixture does not increase the translucency parameter α, where $\alpha = 0$ corresponds to a transparent material and $\alpha \approx 1$ to an opaque material. We propose injecting these two monotonicity constraints of tonal-optical relationships into the model. Note that the proposed concept can be also used for other monotonic relationships between printing materials and optical/visual quantities identified a priori.

We inject tonal-to-optical monotonicity constraints into the model by adding an extra loss that we refer to as *monotonicity loss*, to penalize monotonicity violations. We perform a derivative-based implementation, inspired by Liu *et al.* [LHZL20] who proposed a derivative-based approach to inject monotonicity to arbitrary neural networks for e.g. blog feedback regression or medical image classification. They assigned *equal* importance to all monotonic relationships. Furthermore, they used a derivative normalization that causes the loss to vanish when a small number of violations remain at the end of the model training. In contrast, our strategy automatically adjusts importances for different monotonic relationships, and uses a different derivative normalization to address the described issue of vanishing loss. As will be shown in the experiment section, our approach leads to much fewer monotonicity violations.

Note that monotonicity loss is calculated based on derivatives instead of sample labels. This allows calculating monotonicity loss on any samples that are sampled from the whole input space. To enlarge data coverage, the samples can be re-sampled differently at each training iteration thus vary from iteration to iteration. This has an advantage that theoretically we have an infinite data pool for sampling and training, aiding model generalization.

To calculate the monotonicity loss for lightness CIE-L^* w.r.t. black material K, at each training iteration we select a random set of tonals $\mathcal{M} \subset \mathcal{T}$ and extract the subset $\mathcal{M}_{\mathrm{LK}}$ with positive derivatives of CIE-L^* w.r.t. the black material, i.e.

$$\mathcal{M}_{\mathrm{LK}} = \left\{ \tau \in \mathcal{M} \mid \left. \frac{\partial \mathbf{L}\big(\mathbf{PDL}(\mathbf{t})\big)}{\partial \mathbf{t}_{\mathrm{K}}} \right|_{\mathbf{t}=\tau} > 0 \right\} \tag{4.1}$$

where $\mathbf{t} \in \mathcal{T}$, $\mathbf{t}_{\mathrm{K}}$ is the element in $\mathbf{t}$ corresponding to the black material, and $\mathbf{L} : \mathcal{S} \times \mathcal{A} \mapsto [0, 100]$ extracts lightness CIE-L^* from model predictions, i.e. it uses the predicted reflectance and computes lightness for a given viewing condition (illuminant, observer). We use CIED50 as illuminant and the CIE1931 color matching functions as observer.

The monotonicity relationship between the black tonal $\mathbf{t}_{\mathrm{K}}$ and lightness CIE-L^* is violated by the model for positive derivatives. Thus, only positive derivatives are considered in the loss function:

$$\mathbf{E}_{\mathrm{mono\_LK}}(\mathcal{M}_{\mathrm{LK}}) = \frac{1}{|\mathcal{M}_{\mathrm{LK}}|} \sum_{\tau \in \mathcal{M}_{\mathrm{LK}}} \left. \frac{\partial \mathbf{L}\big(\mathbf{PDL}(\mathbf{t})\big)}{\partial \mathbf{t}_{\mathrm{K}}} \right|_{\mathbf{t}=\tau} \tag{4.2}$$

where $|\mathcal{M}_{\mathrm{LK}}|$ is the cardinality of $\mathcal{M}_{\mathrm{LK}}$, i.e. the number of elements in the set.

Similarly, we induce the monotonic relationship between the transparent material and the translucency parameter α:

$$\mathcal{M}_{\mathrm{AT}} = \left\{ \tau \in \mathcal{M} \mid \left. \frac{\partial \mathbf{A}\big(\mathbf{PDL}(\mathbf{t})\big)}{\partial \mathbf{t}_{\mathrm{T}}} \right|_{\mathbf{t}=\tau} > 0 \right\} \tag{4.3}$$

where $\mathbf{t} \in \mathcal{T}$, $\mathbf{t}_{\mathrm{T}}$ is the element in $\mathbf{t}$ corresponding to the transparent material, and $\mathbf{A} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{A}$ extracts translucency α-values from model predictions. The loss is then computed as follows

$$\mathbf{E}_{\mathrm{mono\_AT}}(\mathcal{M}_{\mathrm{AT}}) = \frac{1}{|\mathcal{M}_{\mathrm{AT}}|} \sum_{\tau \in \mathcal{M}_{\mathrm{AT}}} \left. \frac{\partial \mathbf{A}\big(\mathbf{PDL}(\mathbf{t})\big)}{\partial \mathbf{t}_{\mathrm{T}}} \right|_{\mathbf{t}=\tau} \tag{4.4}$$

Modern deep learning tools allow conveniently computing derivatives of a neural network's output w.r.t. its input, e.g. via the *tf.GradientTape* API of TensorFlow [AAB*15].

4.2.2. Injecting smoothness heuristics

We assume that the printer's optical transfer function describing the forward relationship between tonal values and optical/visual quantities is smooth, i.e. it does not contain high-frequencies such as edges or bumps. Observed high-frequencies are rather measurement or printing noise and should not be considered by the optical printer model. Thus, our aim is to determine a model with maximally smooth predictions without significantly sacrificing accuracy.

For this, we propose a second-order derivative-based smoothing loss that we refer to as *Laplacian loss* because of its similarity to the Laplace operator. Even though this loss could be considered for all output dimensions of the optical printer model, we restrict it to operate on lightness CIE-L^* only, which is computed from the predicted reflectance for a given viewing condition. Considering just lightness minimizes computational effort and allows model optimization w.r.t. the perceptually most relevant contrast-related quantity, since the human visual system's sensitivity to high-frequency achromatic contrasts is larger than to high-frequency chromatic contrasts [VB67, Mul85].

The Laplacian loss is computed as:

$$\mathbf{E}_{\text{lap}}(\mathcal{M}) = \frac{1}{M|\mathcal{M}|} \sum_{\tau \in \mathcal{M}} \sum_{i=1}^{M} \log \left(\left| \frac{\partial^2 \mathbf{L}(\mathbf{PDL}(\mathbf{t}))}{\partial \mathbf{t}_i^2} \bigg|_{\mathbf{t}=\tau} \right| + 1 \right) \tag{4.5}$$

where M is the dimension of the tonal space and $\mathbf{t} = (\mathbf{t}_1, \ldots, \mathbf{t}_M)^T \in \mathcal{T}$. Since the magnitude and not the sign of second-order derivatives is a measure of smoothness, we use their absolute value. Very large second-order derivatives might impair learning by overshooting. Therefore, we take the logarithm to reduce such overshooting risk. We constrain the lower-bound of the logarithm to 0 by adding 1, so that vanishing second-order derivatives do not contribute to the loss.

Similarly as for the monotonicity losses, the samples to compute the Laplacian loss can be re-sampled differently at each training iteration.

4.2.3. Model structure

The RPDL model shares the basic neural network structure with the PDL model described in Chapter 3. In contrast to the PDL model, the number of neurons is smaller but sufficient to obtain similar results. The preference to this smaller model complexity is also motivated by a principle known as *Occam's razor* [GBC16, AMMIL12] stating that an explanation (e.g. a neural network model in our case) to known observations (e.g. samples in our case) should be "trimmed" (conceptually by the "razor") down to the simplest that explains the known observations equally well as more complex explanations. Specifically, the neural network trunk of RPDL has 3 hidden layers each with 200, 300, and 300 neurons respectively. The reflectance-predicting branch has one hidden layer with 100 neurons and an output layer with 31 neurons to predict spectral reflectances $r \in \mathcal{S} = [0,1]^{31}$. The translucency-predicting branch has one hidden layer with 30 neurons and an output layer with 1 neuron to predict translucency $\alpha \in \mathcal{A} = [0,1]$. The RPDL model uses the same activation functions as the PDL model. The detailed network structure is shown as a diagram in Appendix B.2.1. Reducing the network's capacity was done as a hyper-parameter optimization on validation sets.

In our experiments this network structure is also used for the PDL model because it does not adversely impact the model's prediction accuracy.

4.2.4. Loss function and hyper-parameter optimization

Recall that the loss function of the PDL model, denoted as $\mathbf{E}_{\text{PDL}}$, consists of three parts: the spectral Root-Mean-Square Error (RMSE) $\mathbf{E}_{\text{ref}}$, the CIEDE2000 error $\mathbf{E}_{\text{col}}$ computed for specific viewing conditions (illuminant, observer), and the α-based translucency error $\mathbf{E}_{\text{tra}}$. The final loss function is a weighted summation of these three loss functions:

$$\mathbf{E}_{\text{PDL}} = \mathbf{E}_{\text{col}} + a_1\mathbf{E}_{\text{ref}} + a_2\mathbf{E}_{\text{tra}}, \quad \text{with} \tag{4.6}$$

$$\mathbf{E}_{\text{ref}} = \frac{1}{n}\sum_{j=1}^{n}\sqrt{\frac{1}{N}\|\hat{\mathbf{r}}^{(j)} - \mathbf{r}^{(j)}\|_2^2} \tag{4.7}$$

$$\mathbf{E}_{\text{col}} = \frac{1}{n}\sum_{j=1}^{n}\Delta E_{00}(\mathbf{LAB}(\hat{\mathbf{r}}^{(j)}), \mathbf{LAB}(\mathbf{r}^{(j)})) \tag{4.8}$$

$$\mathbf{E}_{\text{tra}} = \frac{1}{n}\sum_{j=1}^{n}|\hat{\alpha}^{(j)} - \alpha^{(j)}|. \tag{4.9}$$

where $\hat{\mathbf{r}}^{(j)}, \mathbf{r}^{(j)} \in \mathcal{S}$ are the predicted and the measured reflectances respectively for the j^{th} sample within all the n samples, $\mathbf{LAB} : \mathcal{S} \mapsto$ CIELAB is the function that computes CIELAB values from reflectances assuming specified viewing conditions, ΔE_{00} means CIEDE2000 color difference, $\hat{\alpha}^{(j)}, \alpha^{(j)} \in \mathcal{A}$ are the predicted and the measured translucency α-values respectively for the j^{th} sample. The weights $a_1, a_2 \in \mathbb{R}_+$ are hyper-parameters and are set as $a_1 = 50$ and $a_2 = 10$ according to the relative magnitudes and importance of the loss functions as described in Chapter 3.

RPDL's loss function is a weighted summation of PDL's loss $\mathbf{E}_{\text{PDL}}$ (Eq. (4.6)) and the three extra losses defined by equations (4.2), (4.4) and (4.5):

$$\begin{aligned}\mathbf{E}_{\text{RPDL}} &= \mathbf{E}_{\text{PDL}} + a_3\mathbf{E}_{\text{mono\_LK}} + a_4\mathbf{E}_{\text{mono\_AT}} + a_5\mathbf{E}_{\text{lap}} \\ &= \mathbf{E}_{\text{col}} + a_1\mathbf{E}_{\text{ref}} + a_2\mathbf{E}_{\text{tra}} + a_3\mathbf{E}_{\text{mono\_LK}} + a_4\mathbf{E}_{\text{mono\_AT}} + a_5\mathbf{E}_{\text{lap}}\end{aligned} \tag{4.10}$$

where weights $a_3, a_4, a_5 \in \mathbb{R}_+$ are hyper-parameters that are adjusted to the printer as follows: The weights are initialized to a very small value $\varepsilon > 0$ (e.g. $\varepsilon = 0.001$), so that the prediction accuracy on validation data is similar to the model that is just trained using $\mathbf{E}_{\text{PDL}}$ [CU21]. Then, the weights are increased until the accuracy of the model on validation data starts decreasing. Specifically, we increase the weights by a factor of 3 until the average color prediction error (CIEDE2000) on validation data increases by 5% of the minimum error achieved so far. We refer to Appendix B.2.2 for more details on the optimization effort.

Adding loss terms $\mathbf{E}_{\text{mono\_LK}}$, $\mathbf{E}_{\text{mono\_AT}}$ and $\mathbf{E}_{\text{lap}}$ can be interpreted as implicitly inducing hypothesis prior preference (Section 2.1.1) over the hypothesis space, to favor hypotheses possessing physically-plausible properties (tonal-optical monotonicity and smoothness) over other hypotheses.

4.3. Experiments

The purpose of the experiments is to evaluate the robustness of the RPDL model, and to compare it with the state-of-the-art model i.e. the PDL model [CU21] presented in Chapter 3.

4.3.1. Data sets

Our experiments use all the three datasets that were used in Chapter 3 to characterize state-of-the-art material-jetting 3D-printers employing six materials (Cyan (C), Magenta (M), Yellow (Y), Black (K), White (W), Transparent (T)): One dataset to characterize a Stratasys J750 printer and two datasets to characterize two Mimaki 3DUJ-553 printers. The datasets consist of reflectance and α-measurements of printed flat targets with known tonal values, except for the "Mimaki 2" dataset where all samples are opaque and thus α-measurements were not collected and not available for experiments. We refer to Section 3.5 for details on the set of sampled tonal values. We also collected a new dataset from a second Stratasys J750 printer using the same measurement procedure (See Section 1.1.2.1). We denote the first Stratasys dataset as *Stratasys 1* and the newly collected dataset as *Stratasys 2*. These two Stratasys datasets are obtained from two different J750 printers using the same materials. In addition to inter-machine variability the datasets deviate also in the function H used to transform tonals to material ratios (see Section 1.1.3).

Stratasys 2: We list the tonal values encoded in 8 bit. The dataset consists of a regular grid $\{0,85,170,255\}^5 \subset$ CMYKT of tonal values, 976 random CMYKT-samples, 1500 random opaque CMYK-samples (i.e. T $= 0$), and 500 random CMYK-samples with T $= 255$. In total there are $4^5 + 976 + 1500 + 500 = 4000$ samples.

4.3.2. Computing and evaluating predictions

Similar to Chapter 3, 300 samples are held-out as the test set, and the remaining samples are split into a validation and a training set: The validation set consists of 10% of these samples to fit the hyper- and regularization parameters and the training set consists of the remaining 90% samples to fit the neural network weights. We refer to the union of these training and validation sets as *big data*, to distinguish from a much smaller data i.e. the *small data* described next. The *small data* consists of only 10% of the big data, and is used to investigate the influence of the proposed approach on a much smaller dataset. The training set always contains the Neugebauer primaries. Small and big data for *Mimaki 1*, *Mimaki 2* and *Stratasys 1* were selected similarly as in Section 3.5. In addition to the Neugebauer primaries, small data for *Stratasys 2* contains randomly-selected samples from the aforementioned 976 random CMYKT-samples described in the previous section, yielding in total 10% of the big data. To compute predictions for PDL or RPDL models, we averaged 10 predictions computed by respective models trained on different decompositions in validation and training sets. We report these so-called *10-fold* predictions, unless explicitly specify *1-fold*. Note that for computing *10-fold* predictions, all 10 models are trained on exactly the same training data. Test set is always unseen during the training.

4.3.3. Software and hardware setup

Our model is implemented with TensorFlow 2.2.0, and is trained on an NVIDIA GeForce RTX 2080 SUPER GPU or an NVIDIA GeForce RTX 3090 GPU. Training a RPDL model takes approx. 420s on 2825 training samples and 170s on 282 samples. Predicting on 300 samples takes approx. 0.005s.

4.3.4. Training method

The neural network structure described in section 4.2.3 is used for both the PDL and RPDL models, but the proposed extra losses (i.e. monotonicity loss and Laplacian loss) are used only for the RPDL models. The initial learning rate is 0.003, with the same learning rate decay used in Section 3.5. The original losses adopted from PDL are calculated on the fixed training samples for both models, while the extra losses are calculated on extra training samples (without using the measurements). These extra data is re-sampled randomly from the whole input space at each training iteration, thus is varying from iteration to iteration. Specifically, at each iteration, 2000 samples are uniformly randomly sampled from tonal space to compute the monotonicity losses, and similarly another 2000 samples for the Laplacian loss. *Early stopping* [GBC16] and *dropout* [SHK*14] regularization strategies are employed to avoid overfitting as described in Section 3.4.3 for both approaches.

4.3.5. RPDL preserves accuracy, or even improves it on small data

With improvements in robustness and plausibility (that will be shown in next section), RPDL preserves accuracy or even improves it on small data as shown in Table 4.1 which summarizes the accuracy comparison between the PDL approach and the proposed RPDL approach. We mark in bold face those model results that outperform the counterparts by > 5 % w.r.t. color accuracy. The results show that the proposed RPDL approach leads to similar accuracy on big data. For small data, RPDL results in an accuracy improvement of at least 5 % in all cases (about 8% smaller average CIEDE2000 errors for Stratasys 1, Stratasys 2, and Mimaki 2) except for the Mimaki 1 average color error, for which we see still an improvement but less than 5 % . Even though our objective is not to improve accuracy, but to achieve better robustness and plausibility preserving accuracy, the improved accuracy on small data indicates that the RPDL model has a better generalization ability than the PDL model. The 90^{th} percentile α-errors of both strategies are below the just noticeable difference (approx. $\Delta\alpha = 0.1$ [UTB*19]) even on small data, meaning the α-accuracy is already sufficient for most applications.

4.3.6. RPDL improves robustness and plausibility

We further evaluate the PDL and RPDL models with respect to violations in L^*-vs-K and α-vs-T monotonicity, as well as smoothness. For this, we randomly selected one million samples from the tonal space $\mathcal{M} \in \mathcal{T}$ and compute $\mathcal{M}_{\text{LK}} \subset \mathcal{M}$ and $\mathcal{M}_{\text{AT}} \subset \mathcal{M}$ according to Eq. (4.1) and (4.3), respectively. The number of violations are reported by the positive derivatives, i.e. the cardinality of $\mathcal{M}_{\text{LK}}$ and

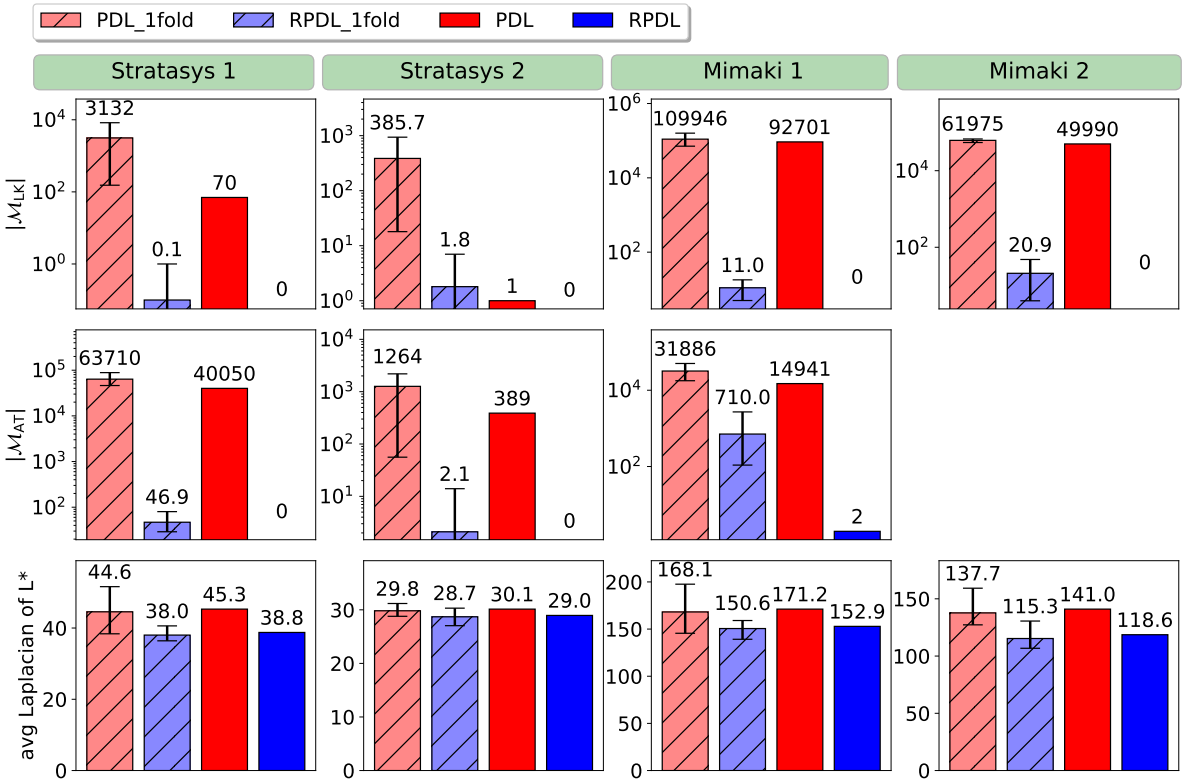


Figure 4.4.: Bar plots of monotonicity violations (L^*-vs-K and α-vs-T) and average Laplacian (original definition). Each quantitative result is shown above the corresponding bar. For 1-fold models, each shown quantitative result is the averaged performance through 10 models, with a whisker to show the maximum and the minimum (Note that for some bars for the 1-fold RPDL, the minimum is zero thus can't be displayed due to log scale in y-axis). Numerical results larger than 1000 are rounded to integer.

Table 4.1.: Model accuracy comparison between PDL and RPDL (A format like "0.44/0.79" means average error is 0.44 and the 90th percentile error is 0.79.)

	Stratasys 1		Stratasys 2		Mimaki 1		Mimaki 2	
	big data	small data	big data	small data	big data	small data	big data	small data
PDL	ΔE_{00} 0.437/0.785 $\Delta\alpha$ 0.0042/0.0090	ΔE_{00} 2.26/4.02 $\Delta\alpha$ 0.0123/0.0291	ΔE_{00} 1.16/2.00 $\Delta\alpha$ 0.0090/0.0183	ΔE_{00} 2.04/3.71 $\Delta\alpha$ 0.0114/0.0234	ΔE_{00} 0.794/1.40 $\Delta\alpha$ 0.0061/0.0118	ΔE_{00} 2.85/6.07 $\Delta\alpha$ 0.0202/0.0486	ΔE_{00} 1.17/2.37	ΔE_{00} 1.90/3.73
RPDL	ΔE_{00} 0.452/0.805 $\Delta\alpha$ 0.0046/0.0104	ΔE_{00} **2.07/3.83** $\Delta\alpha$ 0.0106/0.0236	ΔE_{00} **1.10/1.90** $\Delta\alpha$ 0.0087/0.0175	ΔE_{00} **1.89/3.30** $\Delta\alpha$ 0.0113/0.0261	ΔE_{00} 0.810/1.38 $\Delta\alpha$ 0.0069/0.0142	ΔE_{00} 2.81/**5.73** $\Delta\alpha$ 0.0209/0.0480	ΔE_{00} 1.18/2.38	ΔE_{00} **1.75/3.51**

$\mathcal{M}_{\text{AT}}$, respectively. Smoothness is evaluated by computing the average magnitude of the Laplacian for L^* using the original definition, i.e. without the log-attenuation as in the loss function (Eq. (4.5)). Fig. 4.4 shows the quantitative results, and we make the following observations:

1. The models evaluated on the Stratasys datasets show many more monotonicity violations in α-vs-T than in L^*-vs-K, while the models computed on the Mimaki datasets have a much larger number of violations for both relationships.
2. Model averaging tends to reduce monotonicity violations but does not always work well: e.g. for Mimaki 1, the PDL 10-fold model has a similarly large number of L^*-vs-K violations compared to the PDL 1-fold model. On the other hand, the RPDL 1-fold model already drastically reduces the number of L^*-vs-K violations to 11 which is a much bigger reductions compared to that by model averaging. Furthermore, the 10-fold version of RPDL completely removes violations in both L^*-vs-K and α-vs-T for all the 4 datasets, except for in α-vs-T for Mimaki 1 where there are 2 violations. However, the 2 violations have positive derivatives below 10^{-5} which is significantly smaller compared to up to 25 from the PDL 10-fold model.
3. RPDL always reduces the average Laplacian, e.g. by up to 16% compared to PDL for Mimaki 2, indicating much smoother predictions than PDL.

Fig. 4.5 shows three typical L^*-vs-K monotonicity violations for each of the four datasets with the other tonals held constant. It shows the PDL has L^*-vs-K violations and bumpiness, and model averaging reliefs the issues but fail to completely resolve them, while the RPDL fully removes them all. Fig. 4.6 shows a similar trend in α-vs-T. Note that the RPDL achieves better robustness and plausibility without sacrificing accuracy as summarized in Table 4.1.

Fig. 4.1 illustrates color ramps with L^*-vs-K violations predicted by PDL on the Mimaki 2 dataset. The figure shows that the PDL model overfits to the erroneous data violating monotonicity, while despite the data outlier the RPDL model ensures monotonicity indicating better robustness.

Fig. 4.7 shows a color ramp with L^*-vs-K monotonicity violations predicted by the 1-fold PDL model on the Stratasys 1 dataset. It shows that the PDL-predicted ramp slightly increases in L^* for increasing

K values in the range [0.9, 1.0], i.e. 10% of the K scale, and has a sharp turning point at $K = 0.9$, while the 1-fold RPDL ensures monotonicity with improved smoothness.

Fig. 4.2 visualizes predicted L^* values as a function of K and Y tonals with other materials constant. It shows that the PDL model has bumpy 3D surface and skewed contours violating L^*-vs-K monotonicity, while the RPDL model has much smoother predictions ensuring monotonicity.

We refer to Appendix B.2 for more comparisons that further demonstrate the robustness and plausibility improvements from RPDL.

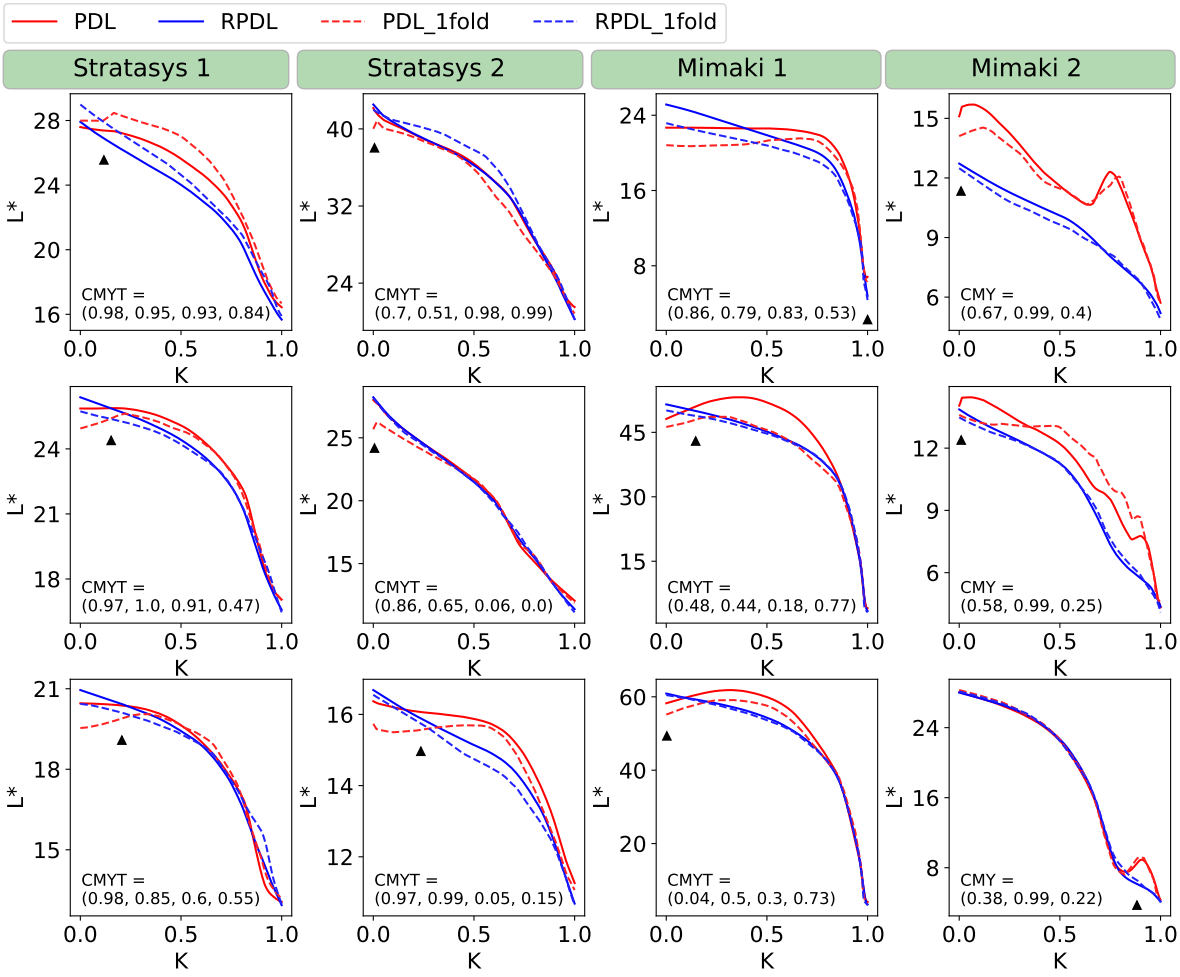


Figure 4.5.: RPDL model resolves L^*-vs-K violations in PDL model. A small black arrow is used to indicate the violation's location.

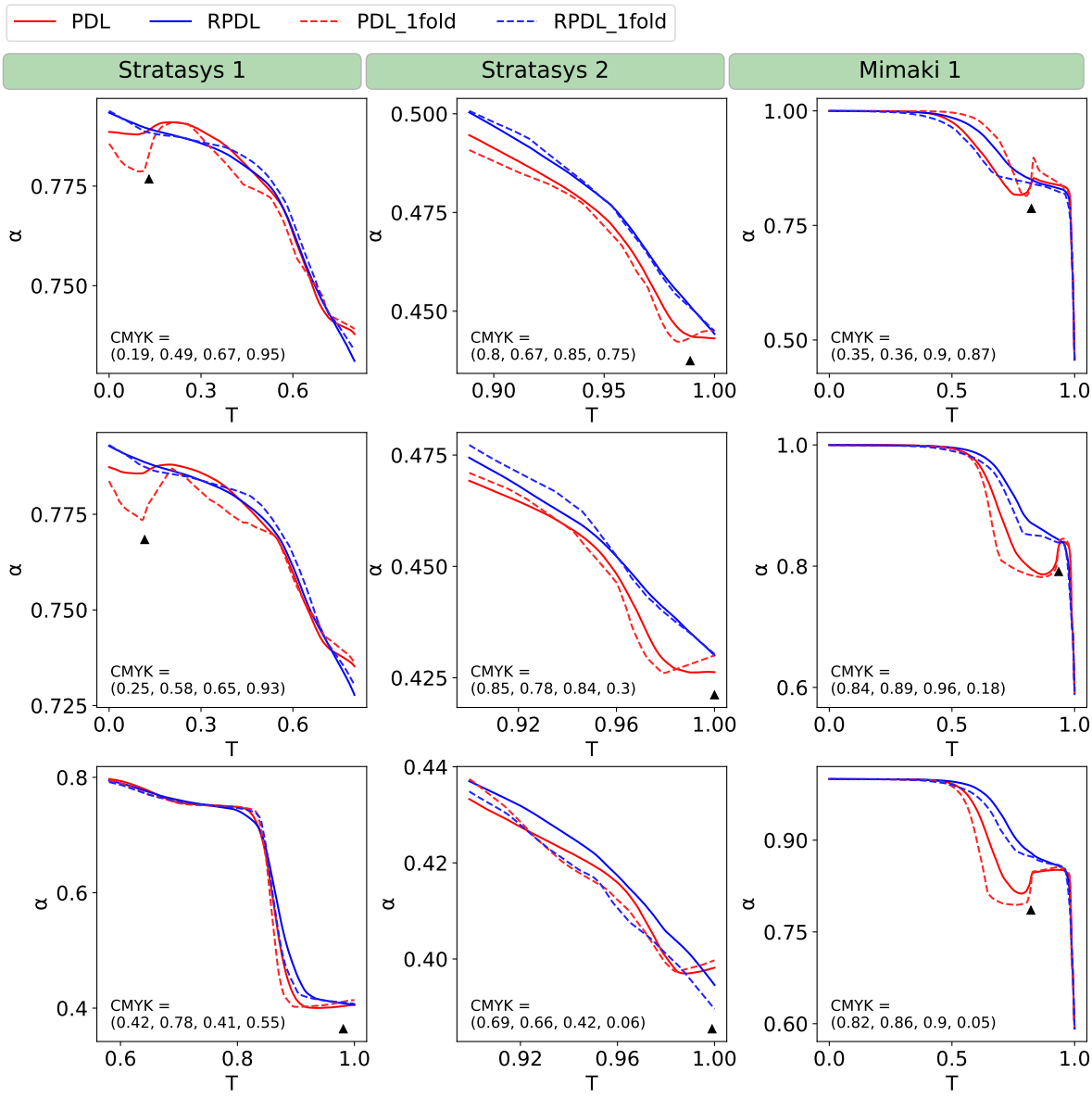


Figure 4.6.: RPDL model resolves α-vs-T violations in PDL model. Note that the Mimaki 2 dataset is not available for the α-vs-T analysis since it only includes fully opaque prints.

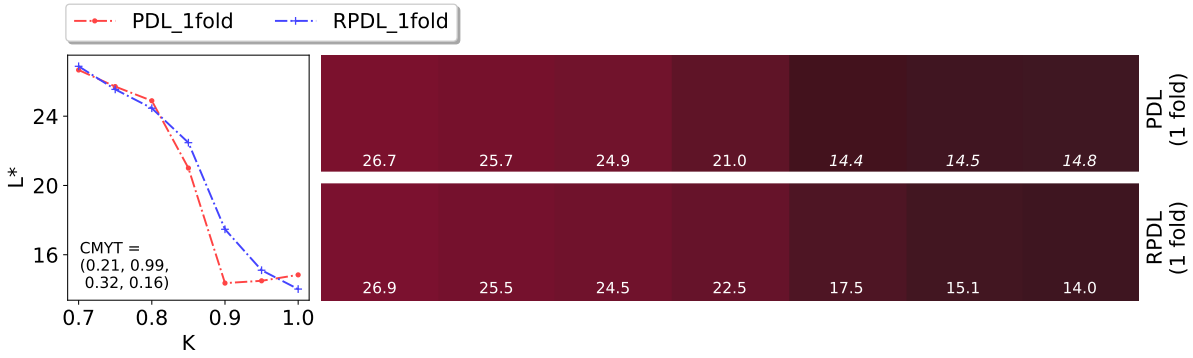


Figure 4.7.: L^* as a function of K with other materials fixed on a tonal case of the *Stratasys 1* dataset, and the resulting colors corresponding to the K values.

Table 4.2.: Model accuracy comparison between models using loss function from Liu *et al.* [LHZL20] and RPDL (A format like "0.44/0.79" means average error is 0.44 and the 90$^{\text{th}}$ percentile error is 0.79.)

	Stratasys 1 (small data)	Stratasys 2 (small data)	Mimaki 1 (small data)	Mimaki 2 (small data)
model with loss from Liu *et al.* [LHZL20]	ΔE_{00} 2.24/4.16 $\Delta\alpha$ 0.0123/0.0281	ΔE_{00} 1.97/3.54 $\Delta\alpha$ 0.0115/0.0231	ΔE_{00} 2.87/5.79 $\Delta\alpha$ 0.0218/0.0507	ΔE_{00} 1.79/3.47
RPDL	ΔE_{00} 2.04/3.58 $\Delta\alpha$ 0.0110/0.0258	ΔE_{00} 1.91/3.41 $\Delta\alpha$ 0.0111/0.0256	ΔE_{00} 2.79/5.67 $\Delta\alpha$ 0.0203/0.0448	ΔE_{00} 1.77/3.55

4.3.7. Compararison to Liu *et al.* regarding injecting monotonic constraints

In Eq. (11) of Liu *et al.* [LHZL20], derivative violations (e.g. negative derivatives when non-decreasing monotonicity is wanted) are summed up and divided by the number of all derivatives (no matter negative, positive or zero). This denominator is equal to the batch size (i.e. the number of training samples) at each training iteration, and it's a constant value e.g. 1024 in their settings. During the training process, the number of derivative violations decreases and becomes much smaller than the aforementioned big constant denominator. This causes the monotonicity loss to vanish when a small number of violations remain at the end of the training.

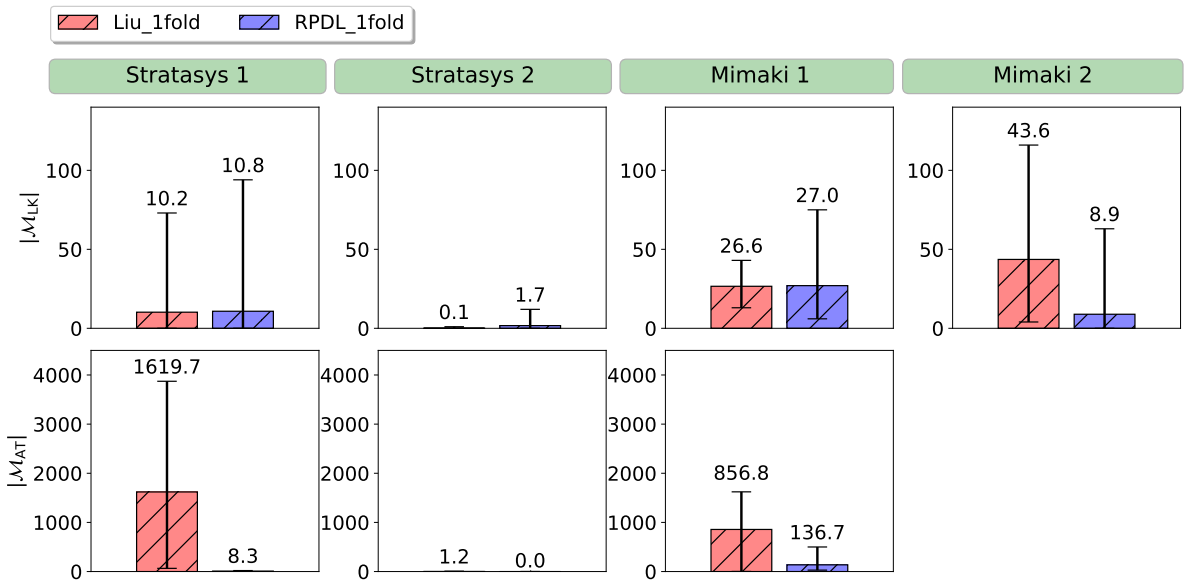


Figure 4.8.: Bar plots of monotonicity violations (L^*-vs-K and α-vs-T). Each quantitative result shown above the corresponding bar is the averaged performance through 10 models, with a whisker to show the maximum and the minimum. Each row is set to have a unified y-axis display range for better comparison.

In contrast, in Eq. (4.2) and Eq. (4.4) of our proposed RPDL, the constant denominator is replaced with the dynamic number of derivative violations at each iteration. This adaptive denominator decreases as the training progresses, thus gives enough weight on the monotonicity loss to resolve the remaining violations. In addition, in contrast to Mean Square Error (MSE) used by Liu *et al.* [LHZL20], we use Mean Absolute Error (MAE) that is less sensitive to outliers and can avoid over-shooting during optimization, especially considering very large derivatives that are observed at some training iterations.

For comparison, we performed experiments on small data (10% data) and trained the model with the monotonicity loss function of Liu *et al.* [LHZL20] and the model with RPDL loss function to have their respective best accuracy. Table 4.2 shows that, our approach leads to better model accuracies compared to Liu *et al.* approach (e.g. 9% accuracy improvement on the Stratasys 1 dataset). Fig. 4.8 shows that, our approach leads to much fewer monotonicity violations compared to Liu *et al.* approach (e.g. up to 200 times fewer monotonicity violations for Stratasys 1).

4.4. Limitations and future work

We propose a learning strategy, i.e. a methodology, of how to induce monotonicity relationships between tonal values and visual quantities as well as smoothness into deep learning-based optical printer

models. We see the particular selection of the constraints as a minor contribution compared to the methodology of how these constraints are induced into the model. We select two monotonic tonal-optical relationships i.e. between lightness (CIE L^*) and black material as well as between translucency (α-value) and transparent material. This approach can be extended canonically to induce other tonal-optical monotonicity relationships into the model. Future work shall focus on exploring new monotonicity relationships and investigating how inducing such prior knowledge into the model can improve its robustness and generalization ability.

4.5. Conclusion

In this chapter, we answered *RQ2* (i.e. *How to improve the* ***robustness and plausibility*** *of deep learning-based optical printer models?*) by inducing physically-plausible constraints and smoothness into models. To address physically-implausible and noisy predictions from the *Pure Deep Learning* (PDL) optical printer model [CU21] described in Chapter 3, we propose a methodology to induce physical heuristics into the model via new loss functions that do not rely on additional printed samples for training: a derivative-based monotonicity loss to induce *a priori* knowledge of tonal-to-optical monotonicity relationships, as well as a Laplacian-based smoothness loss to induce smoothness. For the monotonicity relationships we select L^*-vs.-K and α-vs.-T. We use this approach to introduce the *Robust Plausible Deep Learning* (RPDL) optical model via a learning strategy by combining these loss functions with PDL's original loss functions, using an automatic hyper-parameter optimization considering an upper threshold for color accuracy losses.

Our experiments on four state-of-the-art 6-material 3D printers show that the RPDL optical model is more robust to data outliers and creates much smoother predictions ensuring monotonicity. The improvement in robustness and plausibility is not at the cost of accuracy downgrade, and it even yields up to 8% higher accuracy on small training data indicating a better generalization ability.

In this thesis, as of this point, we have exploited the training data from one *single* printer to tackle accuracy (*RQ1*) and robustness (*RQ2*) of optical printer models. We will exploit data from *other* printers when we will answer in the next chapter the third research question *RQ3*:

How to exploit other printers' data to improve ***data efficiency*** *of the characterization for a particular printer?*

5. Multi-printer learning framework for efficient optical printer characterization

In the previous chapter, we answered *RQ2* (i.e. *How to improve the* ***robustness and plausibility*** *of deep learning-based optical printer models?*) by inducing physically-plausible constraints and smoothness into models.

All deep learning-based optical printer models (i.e. DLLCN, PDL, RPDL) proposed in previous chapters learn on data from only one *single* printer. However, in practice, besides the data collected from the printer to be characterized, we might also have access to historical characterization data of *other* printers. Naturally, one might ask: Given the similarity between some printers, is the historical data from other printers helpful at all for the characterization of a printer of our interest? If so, how could we make use of these datasets? This leads to the *open* research question *RQ3*: *How to exploit other printers' data to improve* ***data efficiency*** *of the characterization for a particular printer?*

In the next chapter, we will answer this open question by proposing a *Multi-Printer Deep Learning* (or MPDL) framework that improves printer models' data efficiency by employing supporting data from other printers.

Experiments demonstrate that the MPDL framework can significantly reduce the number of training samples thus the overall characterization efforts (including sample printing, post-processing, and measuring). With much less efforts and costs for each characterization, it's economically feasible to frequently characterize 3D printers to achieve a high optical reproduction accuracy consistent across different printers and over time, which is crucial for color- and translucency-critical applications.

This chapter is based on the published paper [CU23].

5.1. Introduction

Reproducing visual attributes (color, gloss, or translucency) via 3D printing requires optical printer models that accurately predict optical properties of the printed output from the input signals (e.g. material ratios or arrangements) controlling the printer.

The objective of developing a printer model is to maximize prediction performance while minimizing optical characterization effort.

The PDL model [CU21] presented in Chapter 3 achieved the state-of-art model accuracy, and the RPDL model [CU22] presented in Chapter 4 enhanced model robustness of the PDL model with a similar or slightly higher accuracy. These deep learning-based optical printer models learn on data

from only one *single* printer. Typically these models' accuracies improve as more data is collected from the printer. In this chapter, we will present a new approach to improve accuracy without requiring more data from the printer.

We propose a framework that employs existing characterization data from *other* printers to improve the prediction accuracy of deep learning-based models on a *targeted* printer. This approach improves accuracy without requiring more data from the targeted printer. In turn, it needs drastically less data to achieve a similar accuracy. To our knowledge, this is the first approach for optically modeling a printer using characterization data from other printers, even though these printers might differ from the targeted printer significantly as described in the following.

We denote by a *printing system* the combination of hardware, software and used printing materials. Even if controlled with equal input signals, different 3D printing systems produce outputs that can significantly differ in appearance. This is not surprising because

1. printing material sets can significantly differ in intrinsic optical (refractive index, scattering, absorption, phase function) and mechanical properties (viscosity, surface tension), impacting light transport and physical material mixing behavior,
2. the software can use halftoning algorithms resulting in different material arrangements, or interlacing strategies for material placement,
3. the hardware may differ in material placement accuracy or other factors such as UV exposure in material jetting.

For these reasons, an optical printer model fitted to one printing system performs usually poorly if applied to a different printing system.

On the other hand, even two instances of the *same* printing system (e.g. two printers using the same material set, software and hardware, but located at two different geographic locations) may create deviating outputs for the same control signal because of material lot variability, different maintenance state or deviating environmental factors (humidity or temperature). We call it *inter-printer variability*. The inter-printer variability can be regarded as a case of a more general phenomenon called *distribution shift* [AOS*16, YZLL22] where the data distribution $P_{\text{test}}(X,Y)$ over $\mathcal{X} \times \mathcal{Y}$ (Generally, $X \in \mathcal{X}$ denotes the input, and $Y \in \mathcal{Y}$ denotes the target variable, or label/annotation) of the test data differs from the distribution $P_{\text{train}}(X,Y)$ of the training data used to train predictive models. With $P(X,Y) = P(Y|X)P(X)$, distribution shift of $P(X,Y)$ can occur on either or both of $P(X)$ and $P(Y|X)$. On the one hand, distribution shift of printer characterization data can occur on $P(X)$ when the (training) tonals (e.g. only opaque samples) used to characterize a printer are from a different distribution than the one of the (testing) tonals (e.g. samples with different translucency levels) used by the targeted printer. Another subtle cause is that material lot variability or even different used materials (e.g. the mechanically-rigid *VeroCyan-V* material and the flexible *AgilusCyan* material are used by two printers respectively) leads to the underlying domain shift of the tonal space. On the other hand, distribution shift can occur on $P(Y|X)$ due to e.g. hardware/software difference or simply geographic environment difference (e.g. temperature and humidity) that leads to appearance deviation for the same input tonal. Distribution shift due to inter-printer variability (hardware, software, or materials) usually causes a printer model

fitted for a certain printing system suffer from model deterioration when deployed on other printing systems, as will be shown in our experiment section.

Moreover, these factors (e.g. maintenance state, or environment temperature) causing inter-printer variability may also cause a printing system to deviate over time, which we call *intra-printer variability*. The intra-printer variability can be regarded as a case of a more general phenomenon called *concept drift* [SJ86, LLD*19] which means the statistical properties of the target variable (i.e. the conditional probability distribution $P(Y|X)$ of the target variable Y) gradually changes *over time* in unforeseen ways. In printer characterization, concept drift (e.g. due to machinery maintenance state decaying or temperature/humidity changing over time on the same printer) could cause performance deterioration of printer models trained on historical data from the same printer.

Optical printer models must be re-fitted to account for inter- and intra-printer variability to ensure maximum prediction performance. The re-characterization typically requires a full new round of printing, post-processing, and measuring which is time-consuming and expensive w.r.t. machine and material costs.

Yet, another important factor causing deviations in the visual appearance of the printed object is *post-processing*, such as polishing, clear coating or applying chemical treatments. Fig. 5.1 shows the color difference histogram of the same set of physical samples measured before and after applying a silicone-based plastic care treatment that seals the surface of the prints preventing them from drying out. The average color difference between post-processed and non-processed prints is large, particularly for dark samples. Existing optical printer models do not model postprocess treatments. They consider postprocess treatments only implicitly because their parameters are fitted to printed and measured samples on which the postprocess has been applied. If the postprocess changes, the models must be re-fitted, requiring a complete new set of printed, post-processed and measured samples.

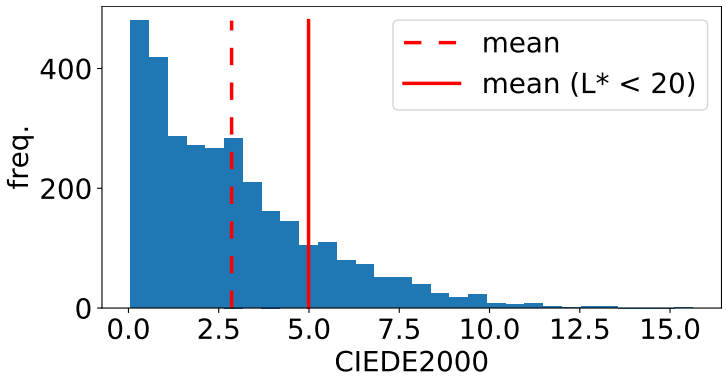


Figure 5.1.: Histogram of CIEDE2000 color differences between printed samples with and without applying plastic care post-process treatment. The dashed line represents the average color difference, and the solid line represents the average color difference for dark samples with $L^* < 20$.

For the sake of simplicity, in the following we will use the term *different printers* to mean all these settings: *different printing systems*, *different instances of the same printing system* and *printing systems with deviating post-process treatments*.

Despite all the factors of different printers causing the appearance of their outputs to deviate, various optical and mechanical mechanisms are shared across such prints. These similarities may include physics of light transport or the volumetric material arrangement for prints which only deviate in post-process treatments. The similarities dominate for similar printing systems even though the inter- and intra-printer variability might be noticeable. This leads to the open research question of how to exploit data from other printing systems possessing these similarities to 1) improve an optical model's prediction accuracy for a particular printer, or 2) to reduce the characterization effort by using fewer training samples without sacrificing accuracy.

We propose a methodology answering this research question. We make the following contributions:

1. A dataset embedding-based learning framework to improve prediction accuracy and data efficiency of deep-learning optical printer models, by using existing characterization data from different printers.
2. A contrastive learning-based approach to learn dataset embeddings that are used as extra inputs (besides material arrangement or tonals) of the above dataset embedding-based learning framework.
3. A learning strategy for training the model and tuning its hyperparameters.

The proposed methodology can drastically reduce the number of required samples that have to be printed, measured and postprocessed to characterize a targeted printing system for achieving an application-specific prediction accuracy. This makes it economically feasible to frequently (e.g. upon performance deterioration alarms, or regularly every month/season instead of every one/several years) characterize machineries of 3D printers to achieve more consistent output across different printers and over time with smaller tolerances, which is crucial for color- and translucency-critical individualized mass production.

5.2. Problem formulation

As presented in Section 1.1.3, an optical printer model is a predictive function $\Phi : \mathcal{T} \mapsto \mathcal{S} \times \mathcal{A}$ that predicts spectral reflectances $\mathbf{r} \in \mathcal{S} = [0,1]^N$ and translucency $\alpha \in \mathcal{A} = [0,1]$ from material arrangement i.e. tonals $\mathbf{t} \in \mathcal{T} = [0,1]^M$.

As described in our previous works [CU21, CU22] presented in Chapters 3-4, different printing systems have different transforms to convert nominal tonals to effective tonals (e.g. 1D-per-tonal curves as described in [WB00]) for final printing. The previous works simply used the nominal tonals and this was not an issue because the optical printer characterizations for different printers are independent. However, in the setting of cross-printer generalization, using the same effective tonal space will reduce the input discrepancy among different printers. For this reason, we use effective tonals.

To characterize a new printer, existing datasets from other printing systems could be helpful in addition to the dataset collected for the targeted printer. Suppose there are m datasets $\{\mathcal{D}_i\}_{i=1}^{m}$ collected from m printers respectively. $\mathcal{D}_i$ is the dataset from the i^{th} printer, consisting of n_i samples, i.e. $\mathcal{D}_i = \{(\mathbf{t}_i^{(j)}, \mathbf{r}_i^{(j)}, \alpha_i^{(j)})\}_{j=1}^{n_i}$, where $\mathbf{t}_i^{(j)} \in \mathcal{T} = [0,1]^M$ is the tonal vector of the j^{th} sample in $\mathcal{D}_i$, $\mathbf{r}_i^{(j)} \in \mathcal{S} = [0,1]^N$ is the corresponding spectral reflectances vector, and $\alpha_i^{(j)} \in \mathcal{A} = [0,1]$ is the corresponding translucency α-value. Without loss of generality, we regard the Γ^{th} ($1 \leq \Gamma \leq m$) printer as the targeted printer that we want to characterize using its dataset $\mathcal{D}_\Gamma$, as well as all datasets from other printers i.e. $\{\mathcal{D}_i | i \neq \Gamma\}_{i=1}^{m}$ as supporting data. In practice, the supporting datasets are from historical characterizations of other printers.

To facilitate reading, we summarize symbols in Table A.1 in Appendix A.

5.3. A dataset embedding-based multi-printer learning framework

We propose a dataset embedding-based *Multi-Printer Deep Learning framework* (or MPDL) to improve characterization accuracy and data efficiency, employing data from other printers. In the MPDL framework, each dataset $\mathcal{D}_i$ is implicitly represented by a pre-learnt vector (referred to as *dataset embedding*) $\mathbf{e}_i \in \mathcal{E} = \mathbb{R}^K$ (e.g. K=256). The dataset embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ ($i \neq j$) for 2 dissimilar printers $\mathcal{D}_i$ and $\mathcal{D}_j$ should be different enough in order to help discriminate printers, while for similar printers the embeddings should be similar to reflect printer similarity. Dataset embeddings $\{\mathbf{e}_i \in \mathcal{E}\}$ are fed into a single neural network to discriminatively learn diverse optical and mechanical mechanisms of multiple printers simultaneously. For this, the optical printer model's input domain is extended by the dataset embedding space $\mathcal{E}$, i.e. $\Omega_\theta : \mathcal{T} \times \mathcal{E} \mapsto \mathcal{S} \times \mathcal{A}$, where Ω_θ is an extended neural network-based optical printer model and θ is the neural network's parameters to be trained.

The proposed MPDL framework is general and not limited to a certain way of obtaining dataset embedding $\mathbf{e}_i \in \mathcal{E}$. In Section 5.4 we will propose a contrastive learning-based approach to learn such dataset embeddings.

5.3.1. Dataset-aware feature learning

To allow the neural network to learn not only printer-dependent mechanisms but also printer-independent behavior (considering underlying similarities among printers), as shown in Fig. 5.2, we designed a 2-path feature learning: A path with hidden layers to learn dataset-aware features from dataset embedding $\mathbf{e}_i$ shown in Fig. 5.2 (I), and the other path with hidden layers to learn dataset-agnostic features from tonals $\mathbf{t}$ shown in Fig. 5.2 (II). The learnt features from the 2 paths are merged via concatenation and fed to a sub-network shown in Fig. 5.2 (III) that predicts the final appearance (translucency, spectral, and CIELAB color). The dataset-agnostic path and the dataset-aware path both consist of one hidden layer with 200 neurons, so the 2-path merged features contains 400 entries. The sub-network has the same structure as PDL or RPDL models, except that the first hidden layer of PDL and RPDL is discarded and that the sub-network takes as input the aforementioned 2-path merged features instead. Specifically,

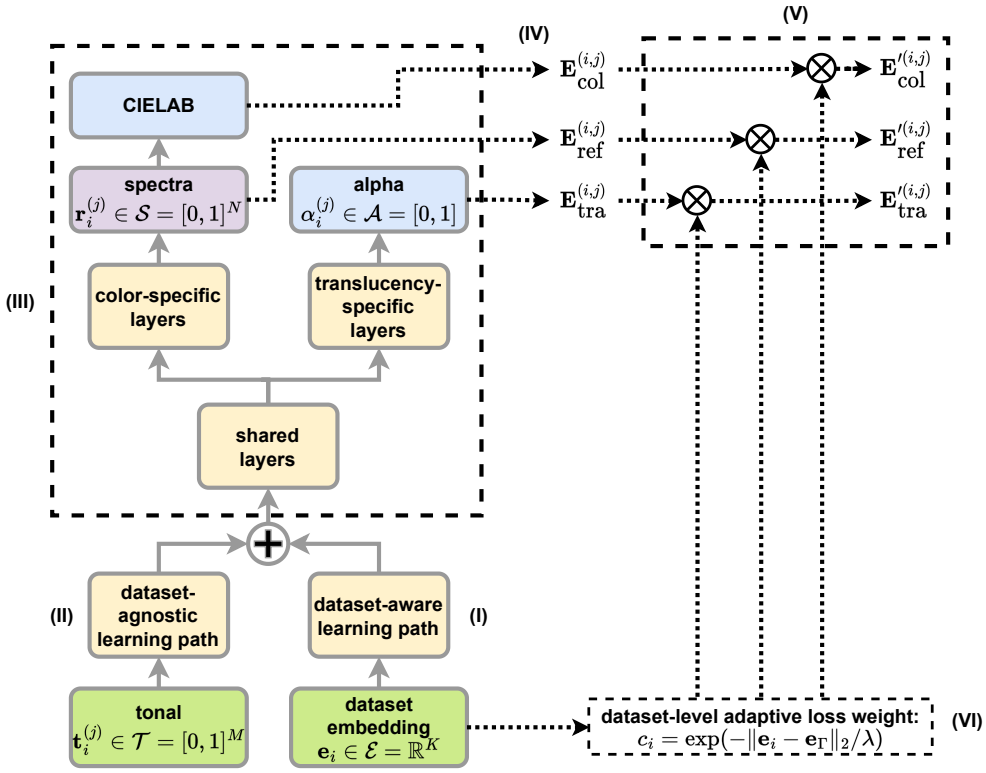


Figure 5.2.: Overview of the proposed dataset embedding-based learning framework. A dataset-aware learning path (I) that learns from dataset embedding $\mathbf{e}_i$ the printer-dependent features, complementing a dataset-agnostic learning path (II) that learns from tonals the printer-independent features. The features learnt by the two paths are concatenated to be fed into a sub-network (III) that predicts the resulting spectral, CIELAB color and translucency α. On these predictions, we calculate (IV) the spectral loss $\mathbf{E}_{\text{ref}}$, the color loss $\mathbf{E}_{\text{color}}$, and the translucency loss $\mathbf{E}_{\text{tra}}$. Superscript (i,j) corresponds to the j^{th} sample in the i^{th} dataset $\mathcal{D}_i$. These losses are adjusted (V) by multiplying with our proposed adaptive loss weight c_i (VI) that depends on dataset similarity measured by L^2 distance between the i^{th} dataset $\mathcal{D}_i$'s embedding $\mathbf{e}_i$ and the targeted dataset $\mathcal{D}_\Gamma$'s embedding $\mathbf{e}_\Gamma$: the smaller this embedding distance, the bigger weight c_i is assigned to every sample of $\mathcal{D}_i$.

the trunk of the sub-network has 2 hidden layers each with 300 neurons. The reflectance-predicting branch has one hidden layer with 100 neurons and an output layer with 31 neurons to predict spectral reflectances $\mathbf{r} \in \mathcal{S} = [0,1]^{31}$. The translucency-predicting branch has one hidden layer with 30 neurons and an output layer with 1 neuron to predict translucency $\alpha \in \mathcal{A} = [0,1]$. Note that PDL and RPDL take as input only the dataset-agnostic tonals and thus are not able to perform dataset-aware feature learning, i.e. are not able to discriminatively learn diverse printer behaviors.

The dataset-agnostic path, the dataset-aware path, and the sub-network are trained together in an end-to-end manner with tonals and dataset embeddings as inputs, and with appearance (translucency, spectral, and CIELAB color) as output.

The whole framework contains around 0.3 million neural network parameters. We show a more detailed structure of the neural network in Appendix B.3.1.

5.3.2. Dataset embedding-based adaptive loss weights

The state-of-the-art deep learning optical model, RPDL, as described in Chapter 4, used a loss that is a weighted summation of 6 loss terms:

$$\mathbf{E}_{\text{RPDL}} = \mathbf{E}_{\text{col}} + a_1\mathbf{E}_{\text{ref}} + a_2\mathbf{E}_{\text{tra}} + a_3\mathbf{E}_{\text{mono\_LK}} + a_4\mathbf{E}_{\text{mono\_AT}} + a_5\mathbf{E}_{\text{lap}} \tag{5.1}$$

where $\mathbf{E}_{\text{col}}$ is the CIEDE2000 color error computed for specific viewing conditions (illuminant, observer), $\mathbf{E}_{\text{ref}}$ is the spectral reflectances Root-Mean-Square Error (RMSE), $\mathbf{E}_{\text{tra}}$ is the α-based translucency absolute error, $\mathbf{E}_{\text{mono\_LK}}$ is monotonicity loss for lightness CIE-L^* w.r.t. black material K, $\mathbf{E}_{\text{mono\_AT}}$ is monotonicity loss for translucency α w.r.t. transparent material T, and $\mathbf{E}_{\text{lap}}$ is Laplacian smoothness loss. RPDL adopted the first three loss terms from PDL described in Chapter 3. Weights $a_{1-5} \in \mathbb{R}_+$ are printer-agnostic hyper-parameters and are selected as described in Chapter 4, in particular $a_1 = 50, a_2 = 10, a_3 = 0.1, a_4 = 0.1$, and $a_5 = 10^{-5}$. For more details on loss terms and automatic hyper-parameter tuning we refer to Section 4.2.4.

In contrast to RPDL where all samples are treated equally in the loss function, we treat samples discriminatively based on their underlying contributions to the characterization of the targeted Γ^{th} printer. We apply a function adaptation to assign bigger weights to samples of those datasets that are more similar to the targeted dataset than those that are less similar. Dataset similarity is evaluated via distance in dataset embedding space. Specifically, we adapt $\mathbf{E}_{\text{ref}}$, $\mathbf{E}_{\text{col}}$ and $\mathbf{E}_{\text{tra}}$ in Eq. (5.1) by applying a per-sample coefficient c_i (Fig. 5.2 (V) and (VI)) as follows:

$$\mathbf{E}'_{\text{ref}} = \frac{1}{\sum_{i=1}^{m} n_i} \sum_{i=1}^{m} c_i \sum_{j=1}^{n_i} \mathbf{E}_{\text{ref}}^{(i,j)} \tag{5.2}$$

$$\mathbf{E}'_{\text{col}} = \frac{1}{\sum_{i=1}^{m} n_i} \sum_{i=1}^{m} c_i \sum_{j=1}^{n_i} \mathbf{E}_{\text{col}}^{(i,j)} \tag{5.3}$$

$$\mathbf{E}'_{\text{tra}} = \frac{1}{\sum_{i=1}^{m} n_i} \sum_{i=1}^{m} c_i \sum_{j=1}^{n_i} \mathbf{E}_{\text{tra}}^{(i,j)} \tag{5.4}$$

where superscript (i, j) corresponds to the j^{th} sample in the i^{th} dataset $\mathcal{D}_i$, and c_i is a dataset-dependent weight shared by all the above three loss terms for samples within the i^{th} dataset $\mathcal{D}_i$, with c_i defined as:

$$c_i = \exp(-\|\mathbf{e}_i - \mathbf{e}_\Gamma\|_2/\lambda) \tag{5.5}$$

where $\|\mathbf{e}_i - \mathbf{e}_\Gamma\|_2$ represents the L^2 distance between the dataset embeddings of datasets $\mathcal{D}_i$ and the targeted dataset $\mathcal{D}_\Gamma$ (Note that the Γ^{th} dataset is from the targeted printer), and $\lambda \in \mathbb{R}_+$ is a hyper-parameter shared across all printers.

Since $\|\mathbf{e}_i - \mathbf{e}_\Gamma\|_2 \in [0, \infty)$, we have $c_i \in (0, 1]$ with $c_i = 1$ for the targeted dataset itself. The more similar a dataset $\mathcal{D}_i$ is to the targeted $\mathcal{D}_\Gamma$, the smaller is the L^2 distance between their dataset embeddings (i.e. the smaller $\|\mathbf{e}_i - \mathbf{e}_\Gamma\|_2$), the bigger is the loss weight c_i for dataset $\mathcal{D}_i$. In this way, the contribution of characterization data from different printers to train the optical model for the targeted printer is considered based on their similarity to the targeted printer's characterization data.

The parameter λ in Eq. (5.5) is the only hyper-parameter to tune in the loss function. When the targeted Γ^{th} printer has sufficient samples in its dataset $\mathcal{D}_\Gamma$, samples from supporting printers might have adverse influences by over-constraining the neural network's capacity, in which case we need to reduce the influence from those supporting printers. On the other hand, when the targeted Γ^{th} printer has only scarce data, the neural network suffers from overfitting, in which case we need to increase the influence from those supporting printers that could provide extra constraints on the neural network. In the MPDL framework, the influence from those supporting printers is automatically adjusted via the tunable λ parameter: The larger λ, the larger is the influence from supporting printers, and vice versus. Note that λ is shared across all printers as a global controller. The automatic tuning process simply selects the $\lambda \in \{0.01, 0.1, 1, 10\}$ that minimizes the CIEDE2000 color difference on the validation data.

The final loss function in our MPDL framework is specified by replacing $\mathbf{E}_{\text{ref}}$, $\mathbf{E}_{\text{col}}$ and $\mathbf{E}_{\text{tra}}$ in Eq. (5.1) with $\mathbf{E}'_{\text{ref}}$, $\mathbf{E}'_{\text{col}}$ by $\mathbf{E}'_{\text{tra}}$ as defined in Eq. (5.2)-(5.4):

$$\mathbf{E}_{\text{MPDL}} = \mathbf{E}'_{\text{col}} + a_1\mathbf{E}'_{\text{ref}} + a_2\mathbf{E}'_{\text{tra}} + a_3\mathbf{E}_{\text{mono\_LK}} + a_4\mathbf{E}_{\text{mono\_AT}} + a_5\mathbf{E}_{\text{lap}} \tag{5.6}$$

5.3.3. Remarks on the necessity of dataset embeddings

One benefit of conditioning the MPDL framework on dataset embeddings is that dataset embeddings allow a dataset-aware learning path (Section 5.3.1) to learn printer-dependent behaviors, complementing the dataset-agnostic learning path that learns only printer-independent behaviors (considering underlying printer similarities).

Another benefit of using dataset embeddings is that dataset embeddings allow adaptive loss weights (Section 5.3.2) for different datasets based on dataset embedding similarity between each supporting dataset and the targeted dataset. This adaptively adjusts the supporting datasets' impacts on the characterization of the targeted printer.

5.4. Contrastively-learnt dataset embeddings

To obtain dataset embeddings $\{\mathbf{e}_i\}$ that are fed into the proposed MPDL framework, we propose a contrastive learning-based strategy. Contrastive learning has recently become a dominant technique in self-supervised learning for natural language processing (NLP) and computer vision. It typically

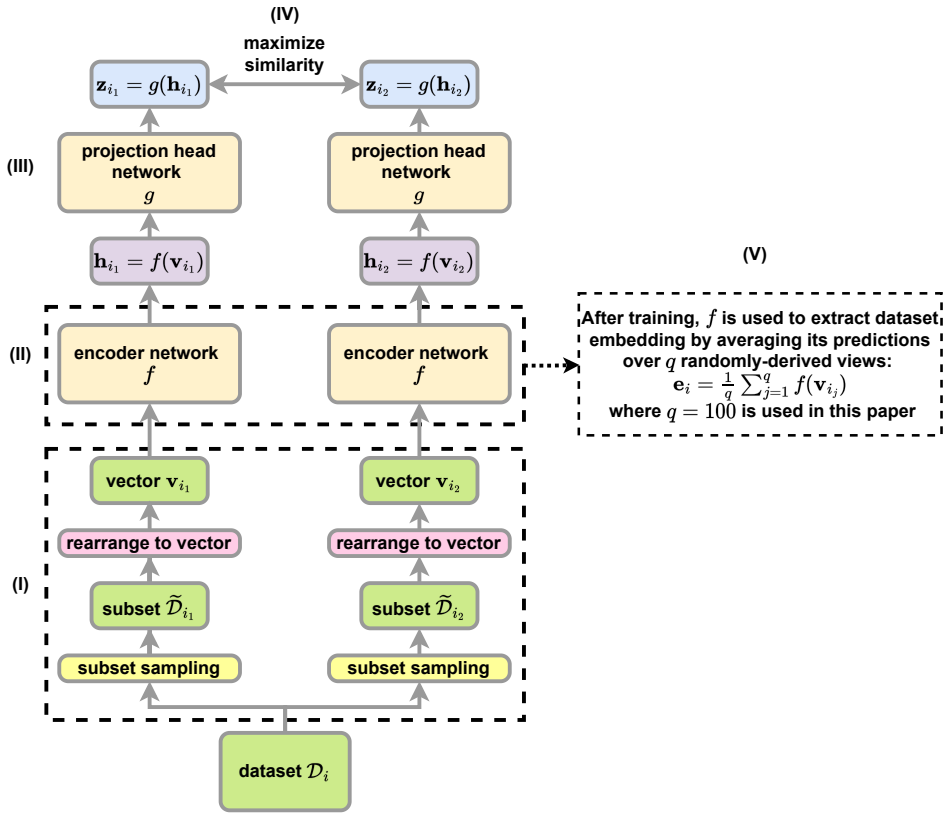


Figure 5.3.: Overview of a proposed strategy to learn dataset embeddings. Each dataset $\mathcal{D}_i$ is processed by a transform module (I) that consists of a subset sampling step resulting in 2 sub-datasets i.e. $\widetilde{\mathcal{D}}_{i_1}$ and $\widetilde{\mathcal{D}}_{i_2}$ and a successive step that rearranges elements of each subset into a vector views $\mathbf{v}_{i_1}$ or $\mathbf{v}_{i_2}$. The 2 views are fed into (II) a neural network *encoder* f to produce intermediate representations $\mathbf{h}_{i_1}$ and $\mathbf{h}_{i_2}$. The intermediate representations $\mathbf{h}$ is further mapped by (III) a small neural network *projection head* g to high-level features $\mathbf{z}_{i_1}$ and $\mathbf{z}_{i_2}$. A *contrastive loss function* (IV) is calculated on $\mathbf{z}$ to encourage the 2 derived views from the same dataset to be similar in $\mathbf{z}$ feature space. After training, dataset embedding (V) is obtained by averaging $\mathbf{h}$ over multiple randomly-derived views.

learns to represent each input sample as a latent vector (also referred to as an embedding or representation) [CKNH20, HFW*20, BJTM22]. The learnt latent vector instead of the original sample is then used as an input of another learning task (referred to as a *downstream* task) e.g. an image classification task [DDS*09]. Contrastive learning requires none or just few data annotations (e.g. Images are annotated to belong to a certain class) which could be difficult or expensive to collect in many applications, and this allows it to learn on a large amount of unlabeled data (e.g. numerous unlabeled images on the Internet) to extract useful information for a downstream task whose training data is labeled

but significantly smaller. To learn useful representations for downstream image classification tasks, Chen *et al.* [CKNH20] proposed a self-supervised (i.e. learning on unlabeled data) contrastive learning framework employing a neural network to predict similar intermediate representations (i.e. the outputs of a hidden layer) for any two views derived from the same image (derived views are 2D images resulted from image augmentations e.g. random clipping, rotation, color distortion etc. on the original image) and dissimilar intermediate representations for any two views derived from two different images. With the learnt intermediate representations as inputs of downstream tasks, their framework considerably outperforms state-of-the-art self-supervised and semi-supervised learning counterparts on ImageNet dataset [DDS*09]. To alleviate the need of a big training batch as needed in [CKNH20], He *et al.* [HFW*20] built a dynamically-updated representation pool and a momentum-based moving averaged query encoder. Bahri *et al.* [BJTM22] extended contrastive learning to tabular classification datasets where the inputs were structured scalar features and the derived views were resulted from corrupting a random subset of features.

These works focused on learning representations at sample level to effectively represent each sample for downstream learning tasks e.g. classification. To the best of our knowledge, none have targeted learning *dataset*-level representation (i.e. dataset embedding) thus none could provide dataset embeddings that are extra needed inputs of the MPDL framework. In contrast to [CKNH20, HFW*20, BJTM22] where views are derived from each *sample* (e.g. an 2D image or a scalar feature vector) by image augmentations (e.g. randomly cropping an image and then resizing) or scalar feature corruption (e.g. randomly replacing a feature value with the median across the dataset), we propose deriving views from each dataset instead of each sample, enabling learning *dataset-level* embeddings for the MPDL framework.

Similar to [CKNH20], our strategy to learn dataset embeddings comprises four major components:

1. A transformation module shown in Fig. 5.3 (I) that randomly transforms any dataset $\mathcal{D}_i$ into two vectors $\mathbf{v}_{i_1}$ and $\mathbf{v}_{i_2}$, that we call *views*. To derive a view from dataset $\mathcal{D}_i$, two successive steps are executed: 1) uniform-randomly sample a subset $\widetilde{\mathcal{D}}_{i_1} \subset \mathcal{D}_i$ of cardinality p (We use $p = 100$); 2) rearrange elements of $\widetilde{\mathcal{D}}_{i_1}$ to a vector $\mathbf{v}_{i_1} = (\mathbf{t}_{i_1}^{(1)}, \mathbf{r}_{i_1}^{(1)}, \alpha_{i_1}^{(1)}, ..., \mathbf{t}_{i_1}^{(p)}, \mathbf{r}_{i_1}^{(p)}, \alpha_{i_1}^{(p)})$ where $\mathbf{t}_{i_1}^{(j)}$ is the tonal of j^{th} sample of $\widetilde{\mathcal{D}}_{i_1}$, $\mathbf{r}_{i_1}^{(j)}$ is the corresponding spectral reflectance, and $\alpha_{i_1}^{(j)}$ is the corresponding translucency. Note that two views $\mathbf{v}_{i_1}$ and $\mathbf{v}_{i_2}$ derived from the same dataset can be correlated because the corresponding subsets $\widetilde{\mathcal{D}}_{i_1}, \widetilde{\mathcal{D}}_{i_2} \subset \mathcal{D}_i$ may have elements in common. A pair of views, $(\mathbf{v}_{i_1}, \mathbf{v}_{i_2})$, derived from the same dataset is called a *positive pair*.
2. A neural network *encoder* f shown in Fig. 5.3 (II) that produces intermediate representations from dataset views (i.e. from $\mathbf{v}_{i_1}$ and $\mathbf{v}_{i_2}$). For simplicity, we adopt the encoder architecture from [BJTM22] that is a fully-connected network comprising 4 hidden layers each with K=256 neurons.
3. A small neural network g, called *projection head*, shown in Fig. 5.3 (III) maps dataset embeddings to higher level features i.e. $\mathbf{z}_i = g(\mathbf{h}_i)$ on which a contrastive loss (see below) is calculated. We adopt the projection head architecture from [BJTM22] that is a fully-connected network comprising 2 hidden layers each with 256 neurons. With a training loss calculated on $\mathbf{z}$, [CKNH20]

and [CFGH20] observed performance gains by using $\mathbf{h}$ rather than $\mathbf{z}$ as learnt representation for downstream tasks, and based on experiments they showed that $\mathbf{h}$ contains more information for downstream tasks than the condensed $\mathbf{z}$.

4. A *contrastive loss function* used in Fig. 5.3 (IV) encourages each positive pair of views (i.e. derived from the same dataset) to be similar in $\mathbf{z}$ feature space, and each negative pair of views (i.e. derived from 2 different datasets) to be dissimilar in $\mathbf{z}$ feature space. Similar to [CKNH20, HFW*20, BJTM22], we adopt the *InfoNCE* [OLV18] loss that is commonly used for contrastive learning, and we calculate it on each positive pair of samples (i, j):

$$\mathcal{L}(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k\in\mathbf{V}|k\neq i}\exp(s_{i,k}/\tau)} \tag{5.7}$$

where $s_{i,j} = \mathbf{z}_i^T \mathbf{z}_j/(\|\mathbf{z}_i\|_2\|\mathbf{z}_j\|_2)$ measures the cosine similarity between $\mathbf{z}_i$ and $\mathbf{z}_j$, τ is a temperature hyper-parameter per [WXSL18] (We use $\tau = 1.0$ as suggested by [BJTM22]), and $\mathbf{V} = \{1_1, 1_2, ..., m_1, m_2\}$ is the index set corresponding to the $2m$ views from the m datasets. Note that negative pairs of views are used in the denominator in Eq. (5.7), i.e. each view from each considered dataset is compared with view i. Intuitively, this loss is a softmax-based cross entropy loss [Mal02] (a common loss for multi-class classification) that tries to classify $\mathbf{z}_i$ to the same class as $\mathbf{z}_j$ among all $\{\mathbf{z}_k \mid k \in \mathbf{V}, k \neq i\}$.

The networks f and g are trained jointly according to Algorithm 1. Note that the 2 derived views from each dataset are re-sampled at every training iteration in order to well cover the dataset. After training, for each dataset $\mathcal{D}_i$, we use f to obtain its dataset embedding by (Fig. 5.3 (V)) averaging f's output i.e. $\mathbf{h}$ representations predicted on q randomly-derived views of $\mathcal{D}_i$. We use $q = 100$. Specifically, $\mathcal{D}_i$'s dataset embedding $\mathbf{e}_i$ is obtained by

$$\mathbf{e}_i = \frac{1}{q}\sum_{j=1}^{q}\mathbf{h}_{i_j} = \frac{1}{q}\sum_{j=1}^{q} f(\mathbf{v}_{i_j}) \tag{5.8}$$

where $\mathbf{v}_{i_j}$ is the j^{th} of q views derived from dataset $\mathcal{D}_i$. The resulting dataset embeddings $\{\mathbf{e}_i\}$ are used in the proposed MPDL framework described in Section 5.3.

5.4.1. Remarks on obtaining dataset embeddings via representation averaging

Representation averaging was not performed in the related works where sample-level embedding was obtained by directly running the encoder network f once on the original sample. This was because in related works the original sample has the same size as the derived views (e.g. a cropped and resized image) that are used to train the neural network. However, this is infeasible for our problem because the datasets have different sizes but fully-connected neural networks require inputs of fixed size. In contrast to related works, we propose deriving views of a fixed size by sampling subsets of a fixed size despite from different datasets, and this allows a *single* neural network to learn and predict on datasets of different sizes. Furthermore, our derived views are of a much smaller size than the original dataset sizes,

Algorithm 1 Contrastive learning-based algorithm for learning dataset embeddings

input: m datasets $\{\mathcal{D}_i\}_{i=1}^{m}$, encoder network f, projection head network g, constant temperature τ, constant p
for each training step **do**
  for all $i \in \{1,...,m\}$ **do**
    uniformly sample 2 subsets $\widetilde{\mathcal{D}}_{i_1}, \widetilde{\mathcal{D}}_{i_2} \subset \mathcal{D}_i$ of cardinality p
    rearrange elements of $\widetilde{\mathcal{D}}_{i_1}$ into a vector $\mathbf{v}_{i_1} = (\mathbf{t}_{i_1}^{(1)}, \mathbf{r}_{i_1}^{(1)}, \alpha_{i_1}^{(1)}, ..., \mathbf{t}_{i_1}^{(p)}, \mathbf{r}_{i_1}^{(p)}, \alpha_{i_1}^{(p)})$. Similarly rearrange elements of $\widetilde{\mathcal{D}}_{i_2}$ into a vector $\mathbf{v}_{i_2}$ *// derive 2 views from each dataset*
    let $\mathbf{z}_{i_1} = g(f(\mathbf{v}_{i_1}))$, $\mathbf{z}_{i_2} = g(f(\mathbf{v}_{i_2}))$
  end for
  let $\mathbf{V} = \{1_1, 1_2, ..., m_1, m_2\}$ *// the index set corresponding to 2m views from m datasets*
  for all $i, j \in \mathbf{V}$ **do**
    let $s_{i,j} = \mathbf{z}_i^T \mathbf{z}_j / (\|\mathbf{z}_i\|_2 \|\mathbf{z}_j\|_2)$ *// dataset-level pairwise similarity*
  end for
  define $\mathcal{L}(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k \in \mathbf{V} | k \neq i} \exp(s_{i,k}/\tau)}$
  let $\mathcal{L} = \frac{1}{m} \sum_{k=1}^{m} [\mathcal{L}(k_1, k_2) + \mathcal{L}(k_2, k_1)]/2$ *// average loss on positive pair (k_1, k_2)*
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network f, discard network g

avoiding the need of a large encoder network and thus avoiding high computation complexity as well as convergence difficulty of training a big neural network. The dataset embedding can be obtained by averaging representations predicted on multiple derived views. The multiple predicted representations can be efficiently computed in a minibatch manner, leveraging parallel computing capacity of modern GPUs.

5.5. Experiments

The purpose of the experiments is to evaluate the data efficiency of the MPDL model, and to compare it with the state-of-the-art models i.e. PDL model [CU21] presented in Chapter 3 and the RPDL model [CU22] presented in Chapter 4.

5.5.1. Data sets

We use all four datasets described in [CU21, CU22] or Chapter 3-4 to characterize state-of-the-art material-jetting 3D-printers employing six materials (Cyan (C), Magenta (M), Yellow (Y), Black (K), White (W), Transparent (T)): Two datasets to characterize Stratasys J750 printers and two datasets to characterize Mimaki 3DUJ-553 printers, respectively. The Stratasys printers use *VeroCyan-V, VeroMag-*

enta-V, VeroYellow-V, VeroBlack, VeroPureWhite, and *VeroClear* materials and the Mimaki printers use *MH100-C-BA, MH100-M-BA, MH100-Y-BA, MH100-K-BA, MH100-W-BD*, and *MH100-CL-BD* materials. The datasets consist of reflectance and α-measurements of printed flat targets with known tonal values, except for those datasets where all samples are opaque and thus α-measurements were not collected and not available for experiments. We denote the two Stratasys datasets as *Stratasys 1* and *Stratasys 2*, and the two Mimaki datasets as *Mimaki 1* and *Mimaki 2*. We refer to Section 3.5 for details on the set of sampled tonal values. In addition, we collected four more datasets from Stratasys printers using the same measurement procedure (See Section 1.1.2.1):

Stratasys 3: The data comes from a printing system similar to Stratasys 1 except that the rigid *VeroCyan-V, VeroMagenta-V, VeroYellow-V, VeroPureWhite* materials have been replaced by flexible *AgilusCyan, AgilusMagenta, AgilusYellow, AgilusWhite* materials. The dataset consists of a regular grid $\{0, 63.75, 127.5, 191.25, 255\}^4 \subset$ CMYK of tonal values, and 1000 random CMYK-samples. All samples are opaque, i.e. $T = 0$. In total there are $5^4 + 1000 = 1625$ samples.

Stratasys 4: The data comes from a different instance of a printing system similar to Stratasys 1. It consists of 450 random CMYKT-samples and 100 CMYKT-samples randomly selected from *Stratasys 1* (i.e. The two corresponding printers printed these 100 samples using the same material arrangement controlled by tonal values, resulting in 100 printed samples from each printer, which can be used to investigate inter-printer variability). In total there are $450 + 100 = 550$ samples.

Stratasys 5: The data consists of the 450 random CMYKT-samples from *Stratasys 4* printed by a different instance of the same printing system, to investigate inter-printer variability.

Stratasys 6: The samples have been printed on a Stratasys *J850* printer with *VeroCyan-V, VeroMagenta-V, VeroYellow-V, UltraBlack, UltraWhite, VeroClear* and *UltraClear* materials. *UltraClear* and *VeroClear* are both controlled by the same tonal value T whereas UltraClear is used in the core of the prints and VeroClear close to the surface. The dataset consists of a regular grid $\{0, 127.5, 255\}^5 \subset$ CMYKT of tonal values, 1757 random CMYKT-samples, 1500 random opaque CMYK-samples with $T = 0$, 500 random CMYK-samples with $T = 255$, 187 light-color opaque CMYK-samples, and 113 random CMYKT-samples from *Stratasys 2*. In total there are $3^5 + 1757 + 1500 + 500 + 187 + 113 = 4300$ samples.

In summary, the characterization data covers a wide range of different printing systems (machinery models and materials) and includes also different instances of the same printing system, allowing investigating model performance on inter-printer dissimilarities.

5.5.2. Computing and evaluating predictions

Random samples from the dataset belonging to the targeted printer are held-out as the *test set*, and the remaining samples in this dataset are split into 2 parts: validation samples to tune the hyper- and regularization parameters and training samples to fit the neural network weights. These training samples (from the targeted printer), as well as all samples in supporting datasets (i.e. from other printers), are used for training. Validation data and test data are only from the targeted printer.

Similarly to the PDL and RPDL models, non-test samples of the targeted printer are uniform-randomly split into validation samples and training samples as per the ratio being validation:training = 10%:90%. This random validation-training splitting is performed 10 times independently with different randomizations e.g. via different random seeds across different splitting runs. For each random splitting, an MPDL model is trained on all training samples (i.e. the training samples from the targeted printer, plus all samples from all other printers), and the model's hyperparameters are fit based on the validation samples from the targeted printer. The 10 different validation-training splittings result in 10 slightly differently trained models. After training, we averaged the predictions computed by these 10 models. Note that the union of training and validation data is exactly the same for the 10 models, with the same test data always unseen.

The contrastive learning neural network is trained on the same union of training and validation data as the neural network in the MPDL framework.

5.5.3. Environment and model training setup

All neural networks are implemented with TensorFlow 2.2.0, and trained on an NVIDIA GeForce RTX 3090 GPU. Training an MPDL neural network takes approx. 300s for 282 training samples of *Stratasys 1* as targeted printer and a total of 19317 samples from supporting printers, i.e. totally 282+19317=19599 training samples. We observe a similar runtime for other targeted printers if a similar number of training samples are used. Training a contrastive learning neural network takes approx. 30s.

For the MPDL neural network, Adam optimizer with an initial learning rate of 0.003 is used. The same learning rate decay as in [CU22] (See Section 4.3.4) is used. For the contrastive learning neural network, Adam optimizer with a fixed learning rate 0.0003 is used.

5.5.4. Improvements in accuracy and data efficiency

In Fig. 5.4, we compare MPDL with the state-of-the-art RPDL model (presented in Chapter 4) in terms of prediction accuracy. All results are evaluated on the test data of the targeted printer. Models trained on the entire union of training and validation data from the targeted printer are indicated with the term *Big Data*. To investigate the performance of the models on much smaller data, we also trained some models with only 10% of this training-validation union. These models are indicated with the term *Small Data*. Note that RPDL uses big data or small data only from the targeted printer, while MPDL also employs all supporting data from other printers. To show how good RPDL generalizes across printers, we show also the best results of the RPDL model on the targeted printer while trained on another printer's dataset (The targeted printer's data was not used for training). These results are indicated by *best reference model*.

The results in Fig. 5.4 show:

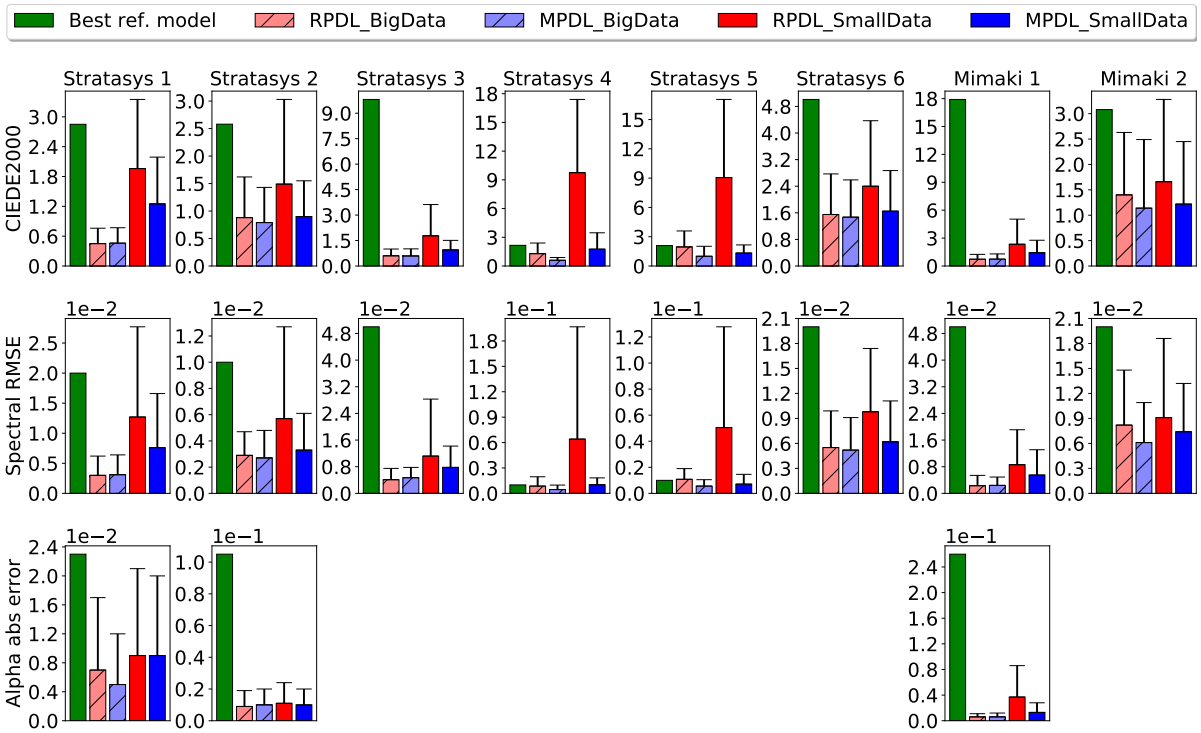


Figure 5.4.: Accuracy comparison between RPDL and MPDL. The 3 rows correspond to color, spectra, and translucency, respectively. The 8 columns correspond to 8 datasets, respectively. Each subplot shows 5 bars representing prediction errors from: the best reference RPDL model, RPDL trained on *big data* (i.e. full training data from the targeted printer), MPDL trained on *big data* (i.e. full training data from the targeted printer, plus all data from the other printers), RPDL trained on *small data* (i.e. 10% of training data from the targeted printer), MPDL trained on *small data* (i.e. 10% of training data from the targeted printer, plus all data from the other printers), respectively. Each bar shows the average prediction error, and the whisker above the bar shows the 90^{th} percentile of that prediction error. For the best reference model, only the average error is shown. Note that translucency α measurements are not available for Stratasys 3-6 and Mimaki 2, thus there are no translucency α plots for these datasets.

1. When directly reusing the best reference model, the mean color prediction error on the targeted dataset (See the first bar in each subplot in the first row) is high, ranging from CIEDE2000 = 2 to 18. One example is *Stratasys 6*, for which the best reference model is trained on *Stratasys 4* data yielding CIEDE2000 around 5. Another example is *Stratasys 3*, for which the best reference model has a very poor average CIEDE2000 prediction error of 10. These two printing systems use very different materials than all the other printing systems, as described in Section 5.5.1. A

color difference of CIEDE2000 > 5 is not acceptable for most color critical applications. This shows the poor model generalization performance across different printing systems.

Characterization datasets from different instances of the same printing system (*Stratasys 1-2* and *4-5*) indicate also a noticeable inter-printer variability because prediction errors from the best reference model are noticeably larger than from RPDL trained on the full targeted training dataset (See the second bar in each subplot). The best reference model's prediction errors range from CIEDE2000 = 2 to 3, which might not be acceptable for color critical applications, such as for prosthetic eyes or dental restorations. This shows also the limits of using an universal color profile for different instance of the same printing system.

2. MPDL outperforms the state-of-the-art RPDL model trained on just the targeted characterization data by a large margin especially for small data: For small data of *Stratasys 5*, MPDL improves color accuracy by up to 6.8 times. For *Stratasys 2*, *Stratasys 5*, and *Mimaki 2*, MPDL trained on small data leads to an accuracy comparable with RPDL trained on *big* data, meaning 10X data reduction for a similar performance. This verifies the data efficiency of MPDL employing supporting data from other printers.
3. Surprisingly and interestingly, despite *very* poor performance of the best reference models due to the differences among printers, adding data from supporting printers into MPDL can significantly improve accuracy for the targeted printer: For *Mimaki 1*, the best reference model has a very high prediction error of CIEDE2000=18, indicating that the targeted printer is very different from the other printers. The RPDL model on small data has an average prediction error of CIEDE2000=2.3, but adding supporting datasets into MPDL significantly reduces the prediction error to CIEDE2000=1.4 on small data, i.e. by about 40%. Even if printing systems are very different, their characterization data possess some underlying similarities, so it is beneficial to let the model learn printer-independent general features on multiple datasets simultaneously, because the supporting datasets provide extra constraints as regularization to improve the model's generalization performance on the targeted dataset. This verifies the effectiveness of MPDL employing supporting datasets from other printers. We speculate that underlying information shared by datasets from *very* different printers can be exploited by the MPDL framework: Potentially it is related to the physics of light transport or similar strategies for halftoning, i.e. statistically similar droplet positioning that ideally has blue-noise characteristic [LA01]. This is an interesting research question that we cannot answer in this thesis.
4. The second row of subplots shows that spectral prediction errors (Root Mean Square Error, or RMSE) have a similar trend as color prediction error (CIEDE2000) and that we can draw the same conclusions as in 1-3.
5. The third row of subplots shows that the accuracy improvement on translucency α predictions is not as large as for color or spectral predictions, even though translucency data is available only in three datasets. Nevertheless, translucency α accuracy is already much lower than the Just-Noticeable-Difference (JND) of approx $\Delta\alpha = 0.1$ [UTB*19]. This means it is sufficient for most applications.

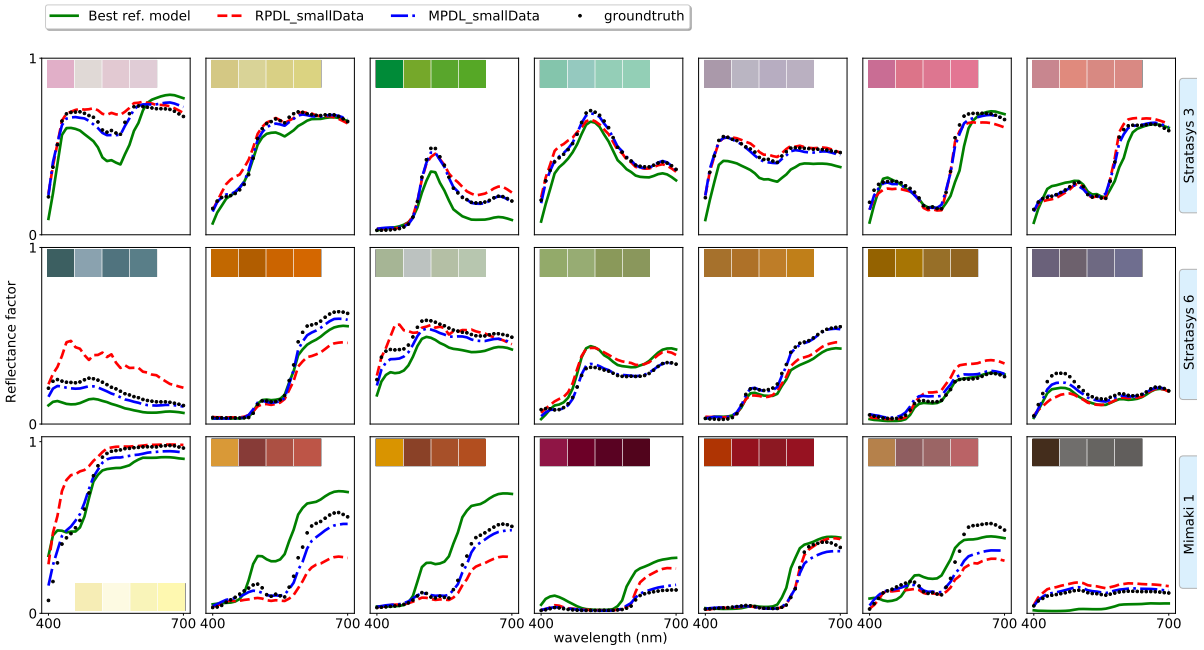


Figure 5.5.: Spectral predictions by different models. The 3 rows correspond to datasets *Stratasys 3*, *Stratasys 6*, and *Mimaki 1*, respectively. The 7 subplots in each row correspond to the 7 test samples on which RPDL and MPDL have the largest root mean square difference. The sRGB color of each spectral prediction is also shown at the upper-left corner of the corresponding subplot, in the order of the best reference model, RPDL with small data (i.e. 10% training data), MPDL with small data, and groundtruth.

5.5.5. Visualization of spectral predictions

In Fig. 5.5, we compare the proposed MPDL with RPDL and the best reference model by visualizing their spectral predictions if trained on small data (i.e. only 10% training data from the targeted printer is used for training MPDL and RPDL). The sRGB color of each spectral prediction is also shown. The figure shows that MPDL's predictions match the groundtruth significantly better than the two counterparts, e.g. in the first subplot of the first row (*Stratasys 3*) where the two counterparts' predicted spectrals and colors are both way off from the groundtruth. This indicates a much higher accuracy from MPDL.

5.5.6. MPDL captures post-processing influence

We compare MPDL trained on small data from *Stratasys 6* where a plastic care treatment was applied as post-processing, with the best reference RPDL model trained on *big data* of a post-processing-free

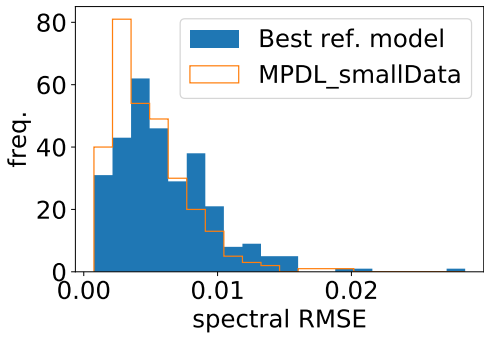


Figure 5.6.: Spectral RMSE histogram

version of *Stratasys 6* dataset. Fig. 5.6 shows histograms of the spectral prediction RMSE of these two models. It shows that even with only 10% data (i.e. small data), MPDL leads to lower spectral prediction errors on dataset *Stratasys 6* than reusing the best reference model trained on a very similar *full*-size dataset (i.e. the post-processing-free version of dataset *Stratasys 6*).

5.6. Limitations and future work

We see the dataset embedding-based learning framework MPDL as our major contribution, and regard the contrastive learning-based strategy of learning dataset embeddings a minor contribution. As a replaceable component, the contrastive learning-based strategy of learning dataset embeddings is not fully exploited, with most settings (e.g. the projection head neural network g during training) adopted from related works that are optimized for different learning tasks e.g. image classification. For future work, we plan to exploit contrastive learning-based strategies for learning dataset embeddings optimized for optical printer modeling, or explore alternative strategies to contrastive learning.

5.7. Conclusion

In this chapter, we answered *RQ3* (i.e. *How to exploit other printers' data to improve* ***data efficiency*** *of the characterization for a particular printer?*) by proposing a methodology to improve the prediction accuracy and data efficiency of an optical printer model for characterizing a targeted printing system, by employing historical characterization data from other printing systems. For that, we proposed a dataset embedding-based *Multi-Printer Deep Learning* (or MPDL) framework with a 2-path feature learning strategy that not only learns printer-independent features shared across multiple printers, but also learns from dataset embeddings the printer-dependent features. The MPDL framework also uses dataset embedding-based adaptive loss weights to balance the supporting datasets' respective influences on the model. We provided a learning strategy for training the model and tuning its hyperparameters. On the other hand, to provide dataset embeddings that are required by the MPDL framework, we

propose a contrastive learning-based approach to learn dataset embeddings. Although conditioned on dataset embeddings, the MPDL framework is general and not limited to any a certain way of obtaining dataset embeddings. To our knowledge, this is the first approach for optically modeling a printer using characterization data from other printers.

Experiments on datasets from eight state-of-the-art multi-material 3D printing systems show that the MPDL framework can drastically reduce the number of required samples that have to be printed, measured and postprocessed to characterize a targeted printing system, for achieving an application-specific prediction accuracy. For some printers, the MPDL framework requires 10% of the samples to achieve a similar accuracy as the state-of-the-art RPDL model [CU22] presented in Chapter 4. This significant improvement in data efficiency makes it economically possible to frequently characterize machineries of 3D printers to achieve more consistent output across different printers with higher accuracy over time, which is crucial for color- and translucency-critical individualized mass production, such as 3D printed prosthetic eyes or dental restorations. A smaller number of characterization samples is also very beneficial for characterizing desktop 3D printing systems that possess a very small build space and thus require more separate print runs than other printing systems. Note that each separate print run requires overhead efforts of cleaning print heads and the print tray. The data efficiency improvement from MPDL reduces the number of separate print runs and thus the overall characterization efforts.

Answering three research questions *RQ1-3* from Chapter 3 to this chapter, we have proposed methodologies improving accuracy, robustness, and data efficiency of optical printer models, respectively. In the next chapter, we will conclude the thesis and provide an outlook on future work.

6. Conclusions and Future Work

In previous chapters, we have answered the three research questions presented in Chapter 1, by proposing methodologies improving accuracy (Chapter 3), robustness (Chapter 4), and data efficiency (Chapter 5) of optical printer models, respectively. In this chapter, we will conclude the thesis and provide an outlook on future work.

6.1. Conclusion

Optical printer models are crucial for high-quality appearance reproduction via full-color 3D printing. In this thesis, we leveraged deep learning to achieve *holistically*-performant optical printer models, in terms of three performance aspects: 1) *accuracy*, 2) *robustness*, and 3) *data efficiency*. To tackle this challenging goal, we answered three corresponding research questions:

RQ1: *How to leverage deep learning to achieve* ***high accuracies*** *of optical printer models?*

RQ2: *How to improve the* ***robustness and plausibility*** *of deep learning-based optical printer models?*

RQ3: *How to exploit other printers' data to improve* ***data efficiency*** *of the characterization for a particular printer?*

6.1.1. Accuracy

Answering *RQ1*, in Chapter 3, we have proposed two deep learning-based printer models that both achieve *high* accuracies with a moderate number of required training samples. The first model is a *Pure Deep Learning* (PDL) model that is essentially a black-box without any physical ground, and the second model is a *Deep-Learning-Linearized Cellular Neugebauer* (DLLCN) model that uses deep-learning to multidimensionally linearize the tonal-value-space of a cellular Neugebauer model [RR10, AA98]. For training both models, we also proposed a loss function balancing reflectance, color and translucency errors in a multi-task learning manner. Appropriate regularization techniques were selected to prevent the neural networks from overfitting. Moreover, a horizontally-shifted sigmoid was proposed to replace the regular sigmoid for the output layers of the PDL model, resolving the issue of problematic near-boundary predictions due to sigmoid's saturation problem.

Experiments showed that both models outperform the traditional cellular Neugebauer model by large margins in terms of accuracy and data efficiency in predicting joint color and translucency validated on two state-of-the-art 6-material 3D printers. They achieve up to 6 times higher accuracy, or, require up to 10 times less data for a similar accuracy.

The proposed deep-learning-based models possess a few intrinsic advantages compared to other phenomenological models. First, to achieve a sufficient accuracy in most applications they do not require dense regular grids as required by the cellular Neugebauer model. Second, they can be updated with more data that is *scattered* instead of gridded, which is interesting for improving prints in particular optical regions of interest, e.g. colors and translucencies of prosthetic implants (teeth, eyes). Furthermore, they learn (in an end-to-end manner from the tonal inputs to the final optical quantity outputs) the impact of material cross-contamination and post-process treatment (e.g. polishing, coating), both of which are difficult to measure and to consider by RTE-based models.

We believe the achieved high accuracies (by the DLLCN and PDL models) could enhance existing or even enable new color- and translucency-critical applications of 3D printing such as prosthetic eyes or dental restorations.

6.1.2. Robustness

Answering *RQ2*, in Chapter 4, we have proposed a methodology to induce physical heuristics into the model via new loss functions that do not rely on additional printed samples for training: a derivative-based monotonicity loss to induce *a priori* knowledge of tonal-to-optical monotonicity relationships, as well as a Laplacian-based smoothness loss to induce smoothness. For the monotonicity relationships we selected lightness L^* vs. black material K, and translucency α vs. clear material T. We used this approach to introduce the *Robust Plausible Deep Learning* (RPDL) optical model via a learning strategy by combining these loss functions with PDL's original loss functions, using an automatic hyper-parameter optimization considering an upper threshold for color accuracy losses.

Adding the monotonicity losses and the smoothness loss can be viewed as implicitly inducing hypothesis prior preference (Section 2.1.1) over the hypothesis space, to favor hypotheses possessing physically-plausible properties (tonal-optical monotonicity and smoothness) over other hypotheses.

Experiments on four state-of-the-art 6-material 3D printers show that the RPDL optical model is more robust to data outliers and creates much smoother predictions ensuring monotonicity. The improvement in robustness and plausibility is not at the cost of accuracy downgrade, and it even yields up to 8% higher accuracy on small training data indicating a better generalization ability.

The enhanced model robustness and smoothness are important for reducing or avoiding unacceptable banding artifacts on textures of the final printouts, particularly for applications where texture details must be preserved. For instance, the texture of a 3D-printed prosthetic eye must be artifact-free in order to visual-pleasingly match the healthy eye.

6.1.3. Data efficiency

Answering *RQ3*, in Chapter 5, we have proposed a *Multi-Printer Deep Learning* (or MPDL) framework that significantly improves printer models' data efficiency by employing supporting data from other printers. The MPDL framework uses a 2-path feature learning strategy that not only learns

printer-independent features shared across multiple printers, but also learns from dataset embeddings the printer-dependent features. The MPDL framework also uses dataset embedding-based adaptive loss weights to balance the supporting datasets' respective influences on the model. We provided a learning strategy for training the model and tuning its hyperparameters. On the other hand, to provide dataset embeddings that are required by the MPDL framework, we propose a contrastive learning-based approach to learn dataset embeddings. Although conditioned on dataset embeddings, the MPDL framework is general and not limited to any a certain way of obtaining dataset embeddings. To our knowledge, this is the first approach for optically modeling a printer using characterization data from other printers.

Experiments on datasets from eight state-of-the-art multi-material 3D printing systems show that the MPDL framework can drastically reduce the number of required samples that have to be printed, measured and postprocessed to characterize a targeted printing system, for achieving an application-specific prediction accuracy. For some printers, the MPDL framework requires 10% of the samples to achieve a similar accuracy as the state-of-the-art RPDL model [CU22] presented in Chapter 4.

This significant improvement in data efficiency makes it economically possible to frequently characterize machineries of 3D printers to achieve more consistent output across different printers with higher accuracy over time, which is crucial for color- and translucency-critical individualized mass production. A smaller number of characterization samples is also very beneficial for characterizing desktop 3D printing systems that possess a very small build space and thus require more separate print runs than other printing systems. Note that each separate print run requires overhead efforts of cleaning print heads and the print tray. The data efficiency improvement from MPDL reduces the number of separate print runs and thus the overall characterization efforts.

6.1.4. Summary

With these three research questions answered, *holistically*-performant optical printer models can be achieved. Please note, each methodology (e.g. RPDL), while improving one performance aspect (e.g. robustness), still preserves the high performance achieved earlier (e.g. accuracy) by the predecessor methodology (e.g. PDL). In other words, the improvement in one performance aspect is not at the cost of a downgrade in other performance aspects. As a result, the final proposed methodology (i.e. MPDL) improves data efficiency to an unprecedented level, while still keeping or even improving other performance aspects (i.e. high accuracy achieved by PDL and physically-plausible robustness achieved by RPDL). In this way, a *holistic* solution is achieved, possessing advantages from all aforementioned methodologies. This benefits real-world applications of full-color 3D printing with requirements in 1) high accuracy in color and appearance reproduction, 2) robust preservation in texture without noisy or banding artifacts, and 3) high data efficiency to allow much less effort per each characterization or allow frequent characterizations for more consistent output across different printers and over time. This holistically-performant solution also sets a new state-of-the-art for future work.

6.2. Future work

Physically plausible constraints In RPDL (Chapter 4), we selected two monotonic tonal-optical relationships i.e. between lightness (CIE L^*) and black material as well as between translucency (α-value) and transparent material. This approach can be extended canonically to induce other tonal-optical monotonicity relationships into the model. Future work shall focus on exploring new monotonicity relationships and investigating how inducing such prior knowledge into the model can improve its robustness and generalization ability. For instance, spectral monotonicity relationships can be explored: A printing material having the smallest/biggest reflectance factor among all available printing materials for a distinct wavelength must decrease/increase the reflectance factor for this wavelength if its fraction increases in the material mixture (assuming non-fluorescent materials). This applies to colored printing materials if the fractions of white and black materials in the material mixture are kept constant. Besides monotonicity, it would be also very interesting to identify and induce other physical/perceptual knowledge of relationships between material ratios (or tonals) and the resulting optical/visual properties.

Strategies for learning dataset embeddings MPDL (Chapter 5), although conditioned on dataset embeddings, is a general framework that is not limited to any a certain way of obtaining dataset embeddings. As a replaceable component to the MPDL framework, the contrastive learning-based strategy of learning dataset embeddings is not fully exploited, with most settings (e.g. the projection head neural network) adopted from related works that are optimized for different learning tasks e.g. image classification. For future work, we plan to exploit contrastive learning-based strategies for learning dataset embeddings optimized for optical printer modeling, and explore alternative strategies to contrastive learning.

Deep learning-based separation For reproducing a distinct spectral or visual properties e.g. color and translucency, an optical printer model needs to be inverted using constraint optimization to obtain the corresponding material ratios or tonals [TRB06, UG06, URB07, UR07] – this process is called *separation*. One could explore deep learning-based approaches to predict material ratios or tonals from spectral or visual properties. Special considerations must be taken care of: 1) since a wanted spectral or visual quantity might be reproducible by multiple material arrangements (one-to-many relationship), for achieving a smooth separation the deep learning model needs to be appropriately regularized to avoid predicting very different material arrangements for similar spectral or visual quantities, and 2) since a wanted spectral or visual quantity might be out-of-gamut and therefore unreproducible, the deep learning model needs to consider gamut mapping.

A. Symbols

To facilitate reading, in Table A.1 we summarize symbols used in the rest of the thesis, in the alphabetical order, with Greek symbols at the end.

Table A.1.: Symbol lookup table

α	A translucency scalar. $\alpha \in \mathcal{A}$
$\alpha_i^{(j)}$	The translucency of the j^{th} sample of the i^{th} dataset.
c_i	The loss weight for the i^{th} dataset. $c_i = \exp(-\|\mathbf{e}_i - \mathbf{e}_\Gamma\|_2/\lambda)$
$\mathbf{e}_i$	The dataset embedding of the i^{th} dataset. $\mathbf{e}_i \in \mathcal{E}$
m	The number of datasets used in the proposed framework
n_i	The cardinality of the i^{th} dataset, i.e. $n_i = \|\mathcal{D}_i\|$
q	The number of randomly-derived views for calculating dataset embedding
$\mathbf{r}$	A spectral vector. $\mathbf{r} \in \mathcal{S}$
$\mathbf{r}_i^{(j)}$	The spectral of the j^{th} sample of the i^{th} dataset.
$\mathbf{t}$	A tonal vector. $\mathbf{t} \in \mathcal{T}$
$\mathbf{t}_i^{(j)}$	The tonal of the j^{th} sample of the i^{th} dataset.
$\mathbf{v}_{i_j}$	The j^{th} derived view of $\mathcal{D}_i$
$\mathcal{A}$	Translucency space. $\mathcal{A} = [0,1]$
$\mathcal{D}_i$	The dataset collected from the i^{th} printer. $\mathcal{D}_i = \{(\mathbf{t}_i^{(j)}, \mathbf{r}_i^{(j)}, \alpha_i^{(j)})\}_{j=1}^{n_i}$
$\widetilde{\mathcal{D}}_{i_j}$	The j^{th} random subset of $\mathcal{D}_i$. $\widetilde{\mathcal{D}}_{i_j} \subset \mathcal{D}_i$
$\mathcal{E}$	Dataset embedding space. $\mathcal{E} = \mathbb{R}^K$ (K=256 in the paper)
M	The dimensions of tonal space.
N	The dimensions of Spectral space, i.e. the number of considered wavelengths.
$\mathcal{S}$	Spectral space. $\mathcal{S} = [0,1]^N$
$\mathcal{T}$	Tonal space. $\mathcal{T} = [0,1]^M$
Γ	The dataset index of the targeted printer. $1 \leq \Gamma \leq m$
λ	A hyperparameter for calculating c_i. $\lambda \in \mathbb{R}_+$

B. Appendix

B.1. Supplemental details for Chapter "Deep learning models for optically characterizing 3D printers"

This section provides supplemental details for Chapter 3.

It gives details on the grid point selection of the cellular Neugebauer model used for comparison in Section 3.5.2. Moreover, high-resolution multimaterial 3D prints are presented reproducing both color and translucency in addition to shape. This is just to illustrate the application area and the need of resource-efficient, accurate optical printer models.

B.1.1. Detailed structures of the neural networks used by PDL and DLLCN

Fig. B.1 and Fig. B.2 show the detailed structure of the neural network used by PDL and DLLCN, respectively. Please zoom in for best view. The diagrams are created via the *tf.keras.utils.plot_model* API of TensorFlow. The "?" in the figure means batch size, i.e. the number of training samples at each training iteration. The number of neurons is shown after the "?".

B.1.2. Grid points for cellular Neugebauer (CN) model

The setup of training set and test set was described in Section 3.5.2. Here we provide extra details especially about the sub-grids for cellular Neugebauer (CN) models.

As mentioned in Section 3.5.2, the traditional cellular Neugebauer (CN) model requires regular grids as interpolation reference. We select a subset of the available grid points to be the training set (the reference points). There are different combinations of grid points, with each combination leading to a different training set thus a different prediction accuracy. The prediction accuracies obtained on all combinations with the same number of training samples are averaged to be the accuracy for that number of training samples. The number of training samples equals to $\prod_i k_i$, where k_i is the number of selected grid points in the i^{th} tonal dimension. Please note that for both the Stratasys dataset and the Mimaki 2 dataset, when there are more than (inclusive) 3 grid points to be selected in a tonal dimension, $\{0, 128, 255\}$ are always included for that dimension. For the Mimaki 2 dataset, when there are more than (inclusive) 4 grid points to be selected in a tonal dimension, $\{0, 85, 170, 255\}$ are always included for that dimension.

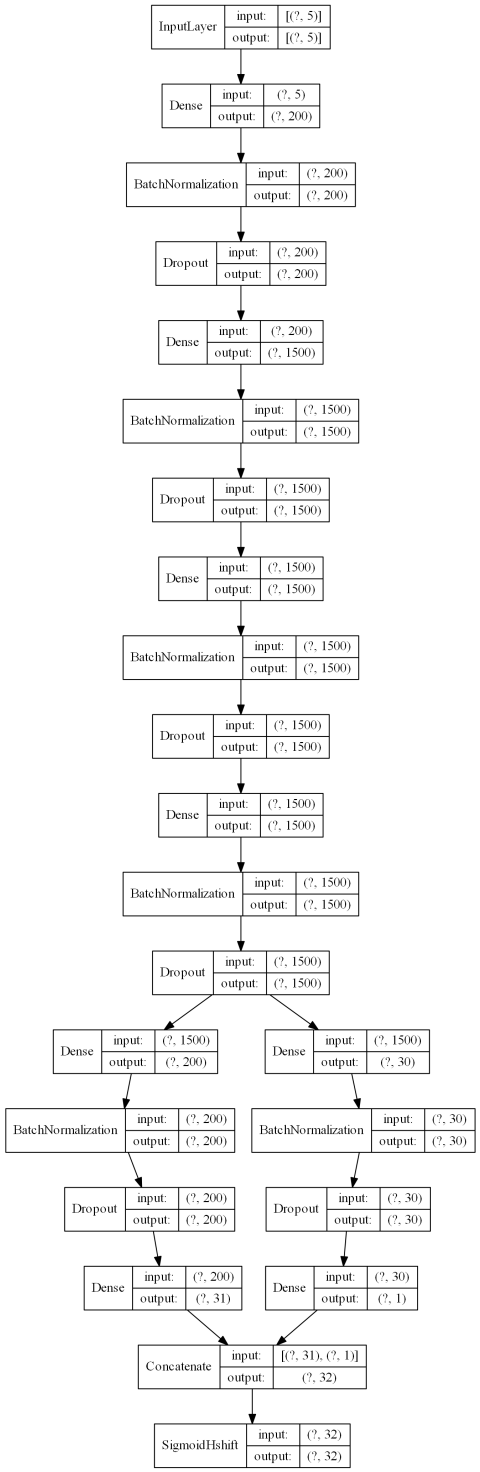


Figure B.1.: PDL neural network structure. Please zoom in for best view.

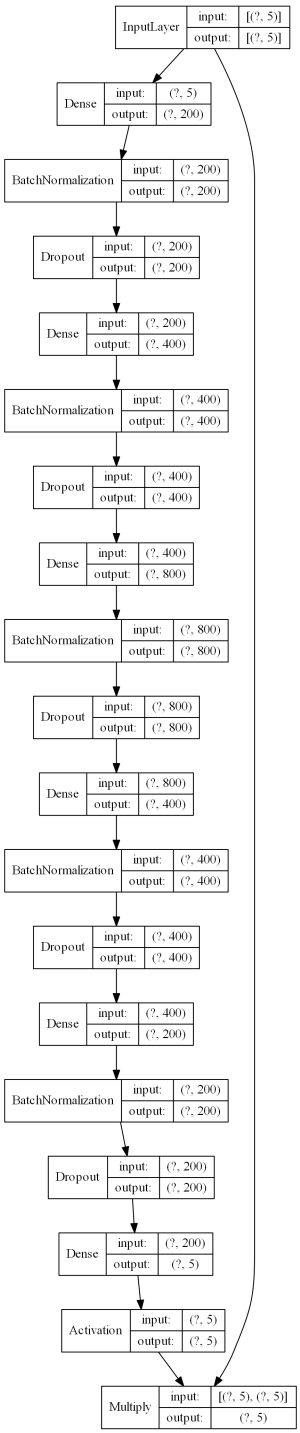


Figure B.2.: DLLCN neural network structure. Please zoom in for best view.

For CN on the Stratasys dataset, the training set is selected from $\{0,64,128,191,255\}^5 \subset$ CMYKT, and the rest of the dataset is used as the test set. In Table 3.2, 243 training samples corresponds to $\{0,128,255\}^5 \subset$ CMYKT (i.e. $\prod_i k_i = 3^5 = 243$ samples), 576 training samples corresponds to 3 grid points in 2 dimensions plus 4 grid points in the remaining 3 dimensions (i.e. $\prod_i k_i = 3^2 \times 4^3 = 576$ samples), 768 training samples corresponds to 3 grid points in 1 dimension plus 4 grid points in the remaining 4 dimensions (i.e. $\prod_i k_i = 3^1 \times 4^4 = 768$ samples), 1024 training samples corresponds to 4 grid points in each dimension (i.e. $\prod_i k_i = 4^5 = 1024$ samples), 1280 training samples corresponds to 4 grid points in 4 dimensions plus 5 grid points in the remaining 1 dimension (i.e. $\prod_i k_i = 4^4 \times 5^1 = 1280$ samples), 1600 training samples corresponds to 4 grid points in 3 dimensions plus 5 grid points in the remaining 2 dimensions (i.e. $\prod_i k_i = 4^3 \times 5^2 = 1600$ samples), 2000 training samples corresponds to 4 grid points in 2 dimensions plus 5 grid points in the remaining 3 dimensions (i.e. $\prod_i k_i = 4^2 \times 5^3 = 2000$ samples), and 2500 training samples corresponds to 4 grid points in 1 dimension plus 5 grid points in the remaining 4 dimensions (i.e. $\prod_i k_i = 4^1 \times 5^4 = 2500$ samples).

For CN on the Mimaki 2 dataset, the training set is selected from $\{0,43,85,128,170,213,255\}^4 \subset$ CMYK, and the test set consists of 300 samples that are selected randomly from the 1099 random samples. In Table 3.2, 256 training samples corresponds to $\{0,85,170,255\}^4 \subset$ CMYK (i.e. $\prod_i k_i = 4^4 = 256$ samples), 500 training samples corresponds to 4 grid points in 1 dimension plus 5 grid points in the remaining 3 dimensions (i.e. $\prod_i k_i = 4^1 \times 5^3 = 500$ samples), 750 training samples corresponds to 5 grid points in 3 dimensions plus 6 grid points in the remaining 1 dimension (i.e. $\prod_i k_i = 5^3 \times 6^1 = 750$ samples), 1296 training samples corresponds to 6 grid points in each dimension (i.e. $\prod_i k_i = 6^4 = 1296$ samples), 2058 training samples corresponds to 6 grid points in 1 dimension plus 7 grid points in the remaining 3 dimensions (i.e. $\prod_i k_i = 6^1 \times 7^3 = 2058$ samples), 2401 training samples corresponds to all the 7 grid points in each dimension (i.e. $\prod_i k_i = 7^4 = 2401$ samples).

B.1.3. 3D Printed Examples

The purpose of this section is to show a few high-resolution multimaterial 3D prints that reproduce both color and translucency in addition to shape. All prints were created by a 6-material Stratasys J750 printer. The material arrangements for these prints are computed using a recently proposed joint color and translucency multimaterial 3D printing pipeline [BATU18]. Translucency is described with the one-dimensional parameter α [UTB*19]. The prints shown here are still based on a printer characterization using a traditional five grid point cellular Neugebauer model requiring 3125 patches (see. Fig. B.3) to predict color and the translucency parameter α. The proposed deep-learning-based models achieve a similar accuracy with just a fraction of training samples. Note that the 3D printing pipeline has to invert the model to obtain the tonal values for printing (separation) and also performs color and translucency gamut mapping. Therefore, such prints cannot be used for assessing the quality of an optical printer model – they are shown here just to illustrate the application area of the proposed models.

Fig. B.4 shows the St. Lucy model printed using the sRGB and α values measured from 3 real samples (see Table B.1), with linear transitions between them. Figure B.5 shows a 10cm head model

printed with an sRGB texture and two different α values. Blurring of geometric and texture details increases for the lower α. Figure B.6 shows a 25cm head model for which color and translucency of human skin is mimicked. Figure B.7 shows a partly textured eye prosthesis that was polished in a post process.

Figure B.3.: Printed target used to fit the five grid point cellular Neugebauer model to predict both color and the translucency parameter α [UTB*19]

Table B.1.: Measurements of real materials and errors of patches printed with the same values. Table partly reproduced from supplementary material of [UTB*19].

Material	Measured from original		Errors of printed patch	
	sRGB	α	CIEDE2000	α-error
green stone	[0.39, 0.40, 0.20]	0.49	2.8690	0.2509
violet stone	[0.23, 0.05, 0.26]	0.68	3.3113	0.0460
green soap	[0.77, 0.82, 0.69]	0.157	8.1293	0.0019

B.2. Supplemental details for Chapter "Inducing robustness and plausibility in deep learning optical 3D printer models"

This section provides supplemental details for Chapter 4.

B.2.1. Detailed structure of the neural network used by RPDL

Fig. B.8 shows the detailed structure of the neural network used by RPDL. Please zoom in for best view. The diagram is created via the *tf.keras.utils.plot_model* API of TensorFlow. The "?" in the figure

Figure B.4.: The St. Lucy model printed (10cm) with varying RGBA values. At the top is that of the violet stone, middle is green stone and bottom is green soap, with linear transitions in between. Figure reproduced from Brunton *et al.* [BATU18].

means batch size, i.e. the number of training samples at each training iteration. The number of neurons is shown after the "?".

To develop the network structure we follow a similar design idea stated in Section 3.4:

"... the basic idea that the number of neurons should increase through the first layers to allow higher layers to have more capacity to capture high-level features of the tonal-optical relationship. Towards the output layer the number should decrease because the number of neurons of the output layer is small and the hidden layers should be able to summarize the features for making final predictions."

With this basic idea, we start with a big network and reduce its number of layers and neurons until the prediction accuracy on validation data drops (Please note that the validation data is split from training data, e.g. 10% of training data). For RPDL, we select a network that is smaller but sufficient to obtain similar accuracy while reducing the network's capacity to overfit.

Bigger networks have higher capacity but are prone to overfitting. We found bigger network sizes do not bring obvious benefits in accuracy, and could hurt model plausibility due to overfitting. On the other hand, the selected network for RPDL has already very low prediction error that is close to the printers' noise level on big data, and we move the focus onto enhancing model plausibility and robustness to data outliers without sacrificing accuracy.

Figure B.5.: 10cm head model printed with $\alpha = 0.786$ (left) and $\alpha = 0.518$ (right). Identical model geometry and illumination conditions. Figure reproduced from Brunton *et al.* [BATU18].

B.2.2. Runtime performance of automatic hyper-parameter optimization

We select a small value of $\varepsilon = 0.001$ as the starting point for the three weight factors for the three new losses. The three weight factors are increased until the validation color accuracy drops by 5 % compared to the best-ever accuracy during the searching process. The number of necessary combinations and runtime differ for different printers and different data sizes. We observed that typically around 100 combinations are needed and the runtime is around 20 hours on the specified hardware. Note that this needs to be done only once for a 3D printing system (consisting of hardware, material, software).

B.2.3. Experiments to compare PDL and the proposed RPDL

Fig. B.9 and B.10 show non-plausible training data violating L^*-vs-K monotonic relationship, and that the RPDL predictions are monotonic for this relationship despite data outliers in contrast to the PDL model. The right part of each figure also shows the resulting colors corresponding to the sampled values of K of the L^*-vs-K curve plot.

Fig. B.11 shows that the PDL model has predictions that violate α-vs-T monotonic relationship and are bumpy, and that the RPDL model has much smoother predictions ensuring α-vs-T monotonic relationship.

Fig. B.12 to B.14 show that the training data has errors that violate L^*-vs-K or α-vs-T monotonic relationship, and that the RPDL model is more robust to avoid being misled by the data errors when compared to the PDL model.

Figure B.6.: 25cm head model printed with $\alpha = 0.518$.

Fig. B.15 and B.16 visualize how monotonicity violations affect the resulting colors. The figures show that in the PDL strategy the colors either become brighter or almost constant as K increases indicating violations of L^*-vs-K monotonic relationship or change suddenly indicating implausibilities, while the proposed RPDL strategy overcomes these monotonicity issues and bumpiness-implausibilities. Notice that the PDL model shows the aforementioned issues in different color areas.

Fig. B.17 to B.19 visualize L^* as a function of K and another material with other materials fixed, via 3D plots and contour plots. The first row corresponds to the PDL strategy and the second row to the proposed RPDL strategy. The 3D plots show that the RPDL strategy leads to much more plausible predictions in terms of L^*-vs-K monotonicity and smoothness, compared to the PDL strategy. In addition, in the contour plots of the PDL strategy, there are skewed curves and isolated "islands". Drawing a profile across such a contour island along the K direction will result in a bumpy L^*-vs-K curve that violates monotonicity. In contrast, the proposed RPDL strategy avoids this kind of contour islands and has much smoother contour curves, indicating better robustness and plausibility.

B.2.4. Experiments to compare RPDL with conventional models

We compare RPDL with conventional models, i.e. the traditional Cellular Neugebauer (CN) model, the Deep-Learning-Linearized Cellular Neugebauer (DLLCN) model, and the Pure Deep Learning (PDL)

Figure B.7.: 3D printed partly-textured eye prosthesis that was polished in a post-process.

model. DLLCN and PDL were proposed in our previous paper [CU21], and compared with the CN model in that paper. Please note that the CN and DLLCN models require a regular grid, hence they cannot be evaluated on Mimaki 1 dataset due to the non-aligned data of the dataset. We compare these four models at different data sizes, i.e. 243 training samples (3 grid points in CMYKT space for CN) for Stratasys 1, 1024 samples (4 grid points in CMYKT space for CN) for Stratasys 2, and 2401 samples (7 grid points in CMYK for CN) for Mimaki 2.

Table B.2 shows that RPDL's accuracy is much better than that of CN and DLLCN, and similar to that of PDL.

Fig. B.20 shows that CN has very few monotonicity violations for Stratasys 1 and 2, while has a big number of violations for Mimaki 2. That's because that CN uses just a small number of grid points, i.e. 3 grid points for Stratasys 1 and 4 grid points for Stratasys 2, and because that the grid points are sparse and far from each other and thus avoid bumpy groundtruth between neighbor reference points. On the other hand, for Mimaki 2, CN uses 7 grid points which are much more dense likely leading to bumpy groundtruth between neighbor reference points. This kind of bumpiness in groundtruth causes the interpolation in CN to have lots of violations. In contrast, RPDL is stable at small number of violations, with much fewer violations than that of DLLCN and PDL, and with much better accuracy than that of CN.

B.3. Supplemental details for Chapter "Multi-printer learning framework for efficient optical printer characterization"

This section provides supplemental details for Chapter 5.

Table B.2.: Model accuracy comparison between conventional models and RPDL. We compare these four models at different data sizes, i.e. 243 training samples (3 grid points in CMYKT space for CN) for Stratasys 1, 1024 samples (4 grid points in CMYKT space for CN) for Stratasys 2, and 2401 samples (7 grid points in CMYK for CN) for Mimaki 2.

	Stratasys 1 (243 samples)	Stratasys 2 (1024 samples)	Mimaki 2 (2401 samples)
CN	ΔE_{00} 3.81/7.20 $\Delta\alpha$ 0.0461/0.1725	ΔE_{00} 5.26/9.36 $\Delta\alpha$ 0.0234/0.0514	ΔE_{00} 1.74/3.57
DLLCN	ΔE_{00} 3.80/7.57 $\Delta\alpha$ 0.0302/0.0685	ΔE_{00} 2.30/4.24 $\Delta\alpha$ 0.0129/0.0280	ΔE_{00} 1.61/2.87
PDL	ΔE_{00} 2.54/4.84 $\Delta\alpha$ 0.0140/0.0347	ΔE_{00} 1.38/2.41 $\Delta\alpha$ 0.0090/0.0184	ΔE_{00} 1.18/2.18
RPDL	ΔE_{00} 2.35/4.45 $\Delta\alpha$ 0.0113/0.0251	ΔE_{00} 1.26/2.21 $\Delta\alpha$ 0.0088/0.0170	ΔE_{00} 1.15/2.32

B.3.1. Neural network structure of the proposed MPDL framework

Fig. B.21 shows the detailed structure of the neural network used by the MPDL framework. Please zoom in for best view. The diagram is created via the *tf.keras.utils.plot_model* API of TensorFlow. The "?" in the figure stands for the batch size, i.e. the number of training samples at each training iteration. The number of neurons is shown after the "?".

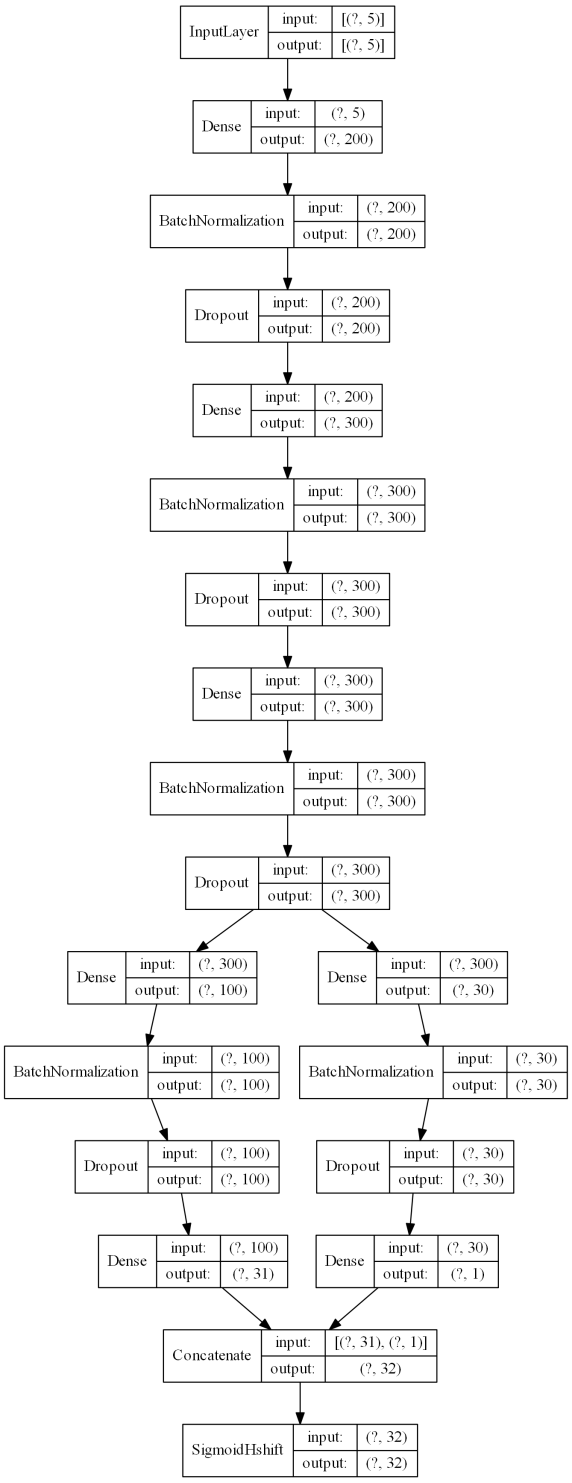


Figure B.8.: RPDL neural network structure. Please zoom in for best view.

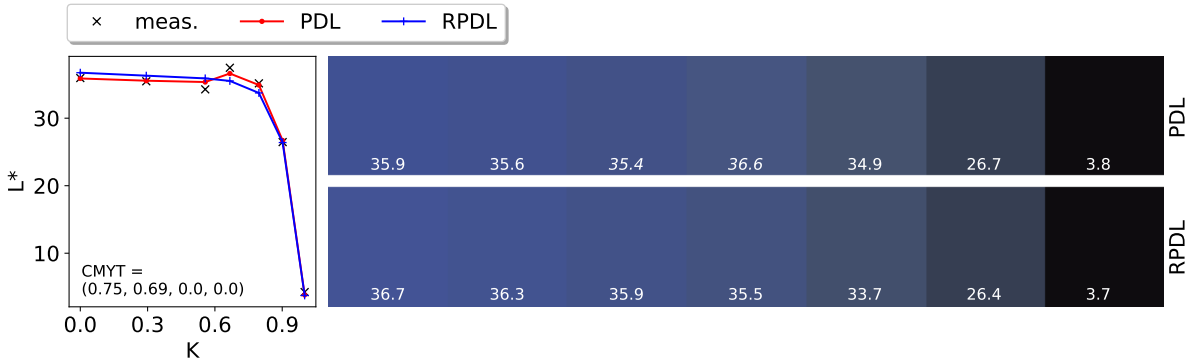


Figure B.9.: L^* as a function of K with other tonals fixed (values shown at the lower left of the plot) on a tonal case of the *Mimaki 1* dataset, and the resulting colors corresponding to the K values. The L^* values are shown as white numerical text, with italic font indicating L^*-vs-K monotonicity violations.

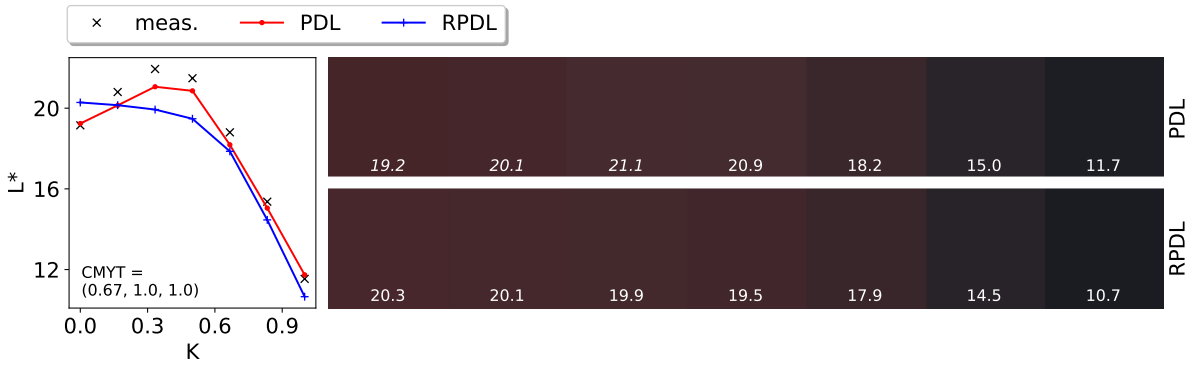


Figure B.10.: L^* as a function of K with other materials fixed (values shown at the lower left of the plot) on a tonal case of the *Mimaki 2* dataset, and the resulting colors corresponding to the K values.

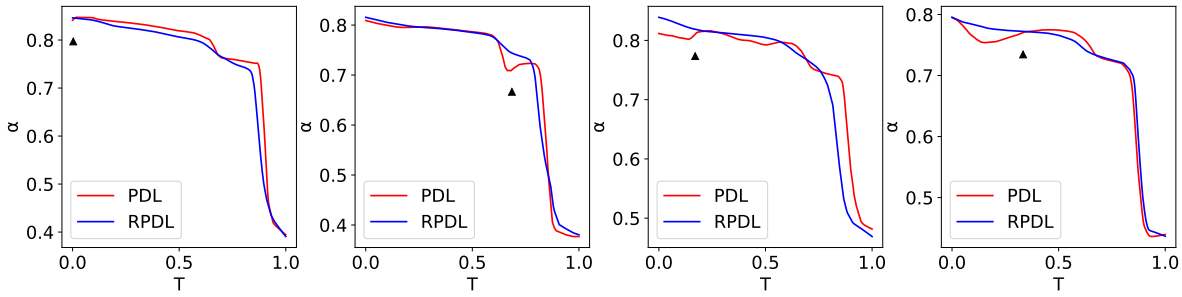


Figure B.11.: α as a function of T with other materials fixed on a tonal case of the *Stratasys 1* dataset.

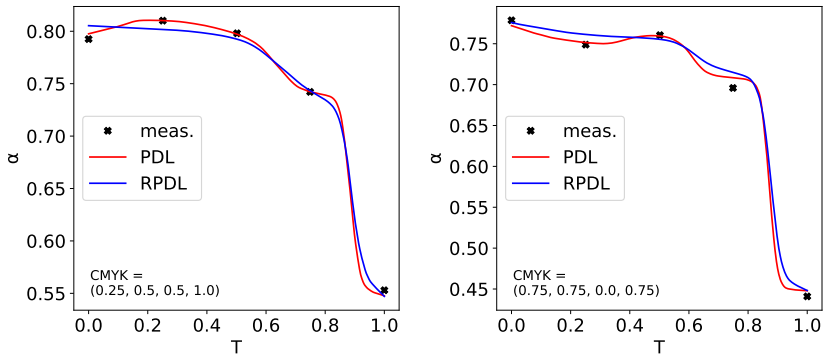


Figure B.12.: α as a function of T with other materials fixed on a tonal case of the *Stratasys 1* dataset. Please note that the measured data itself has monotonicity violations.

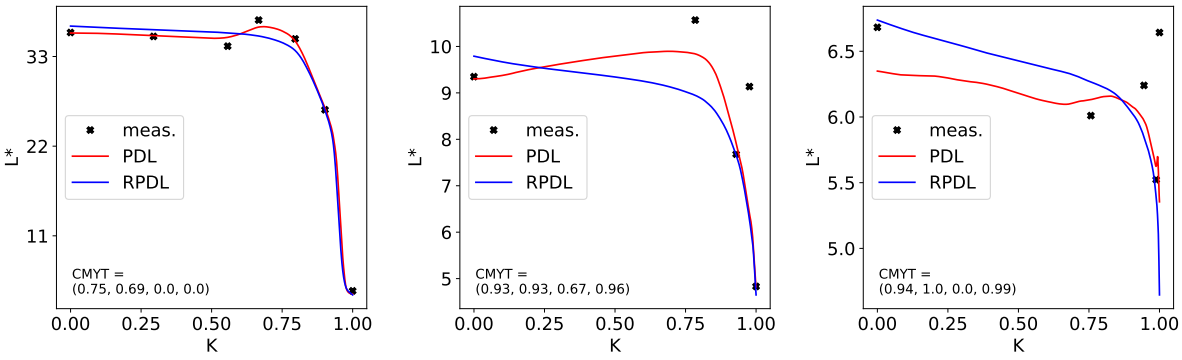


Figure B.13.: L^* as a function of K with other materials fixed on a tonal case of the *Mimaki 1* dataset.

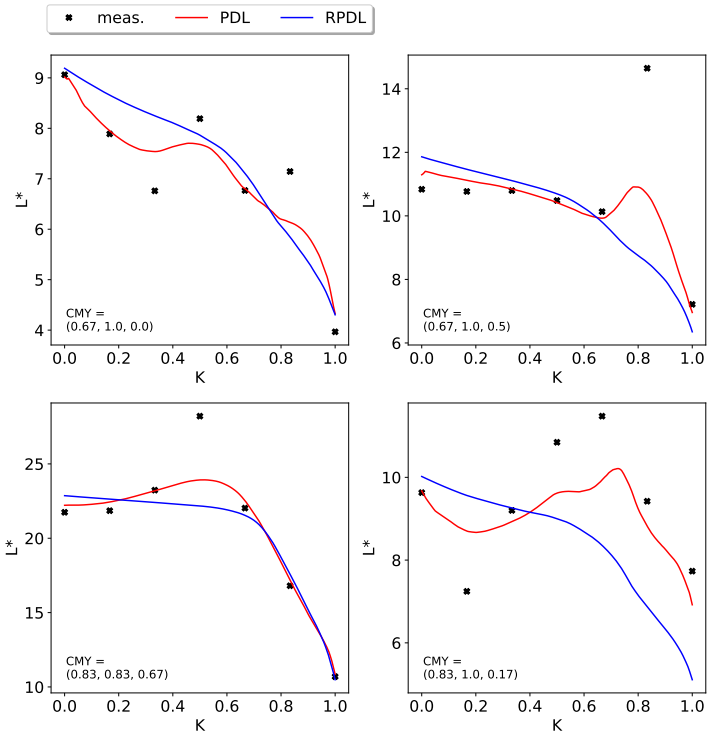


Figure B.14.: L^* as a function of K with other materials fixed on a tonal case of the *Mimaki 2* dataset.

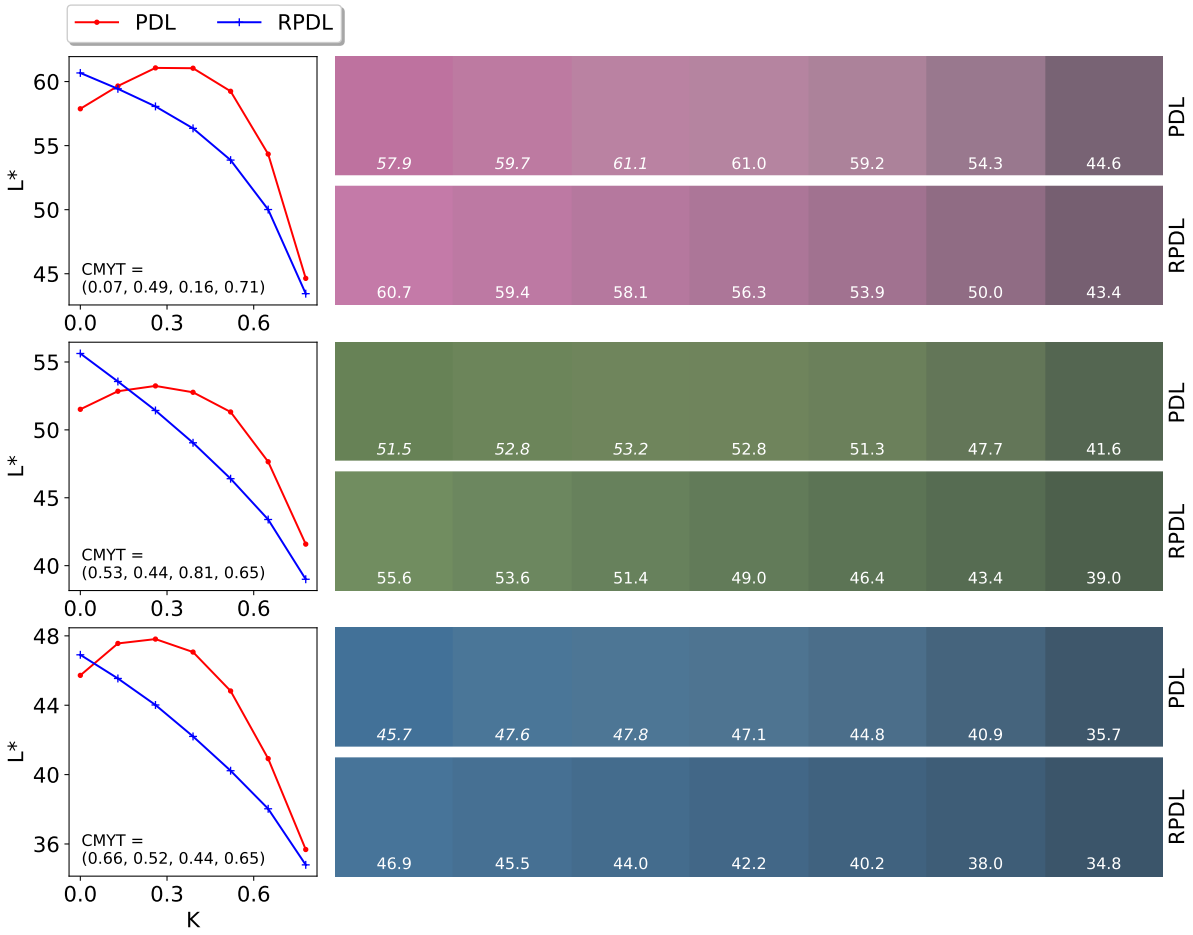


Figure B.15.: L^* as a function of K with other materials fixed on a tonal case of the *Mimaki 1* dataset, and the resulting colors corresponding to the K values.

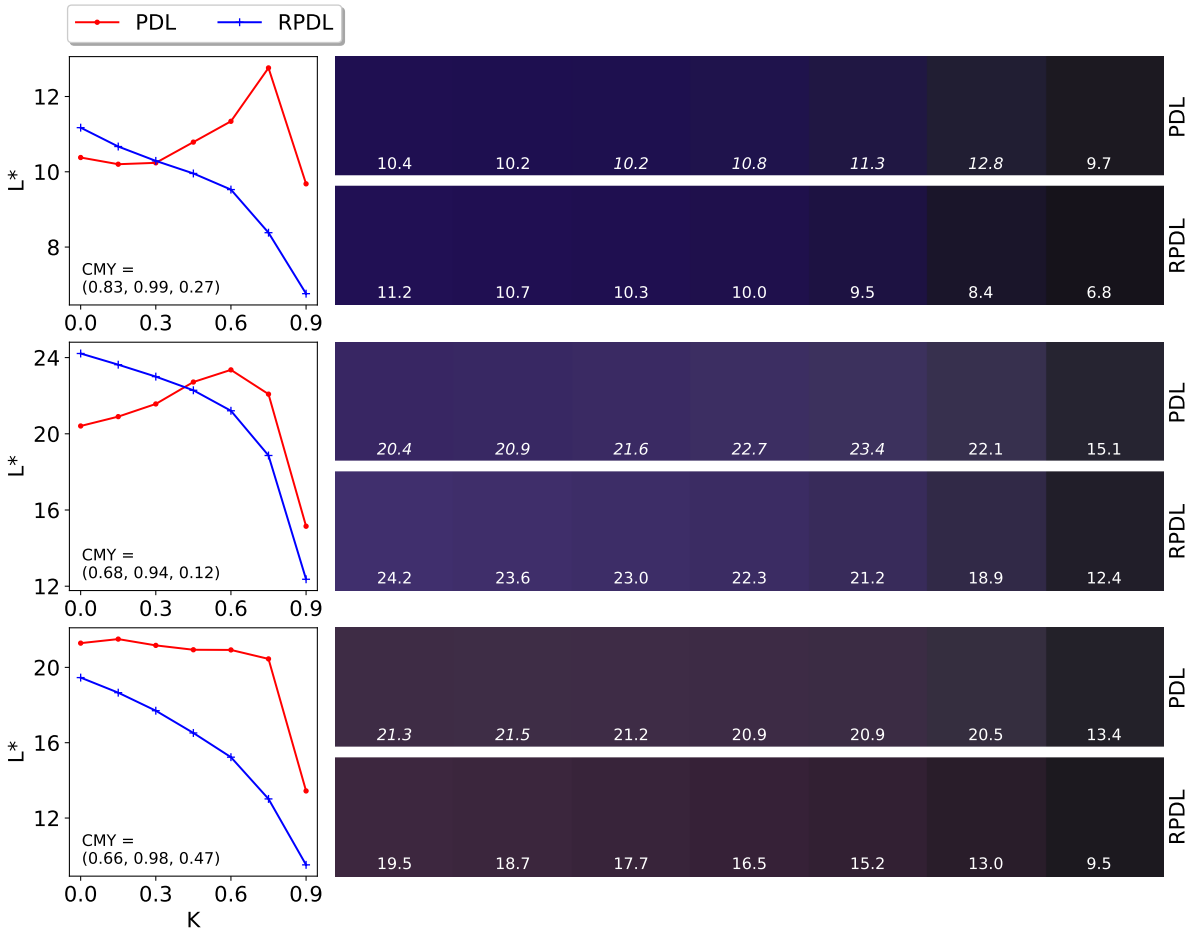


Figure B.16.: L^* as a function of K with other materials fixed on a tonal case of the *Mimaki 2* dataset, and the resulting colors corresponding to the K values.

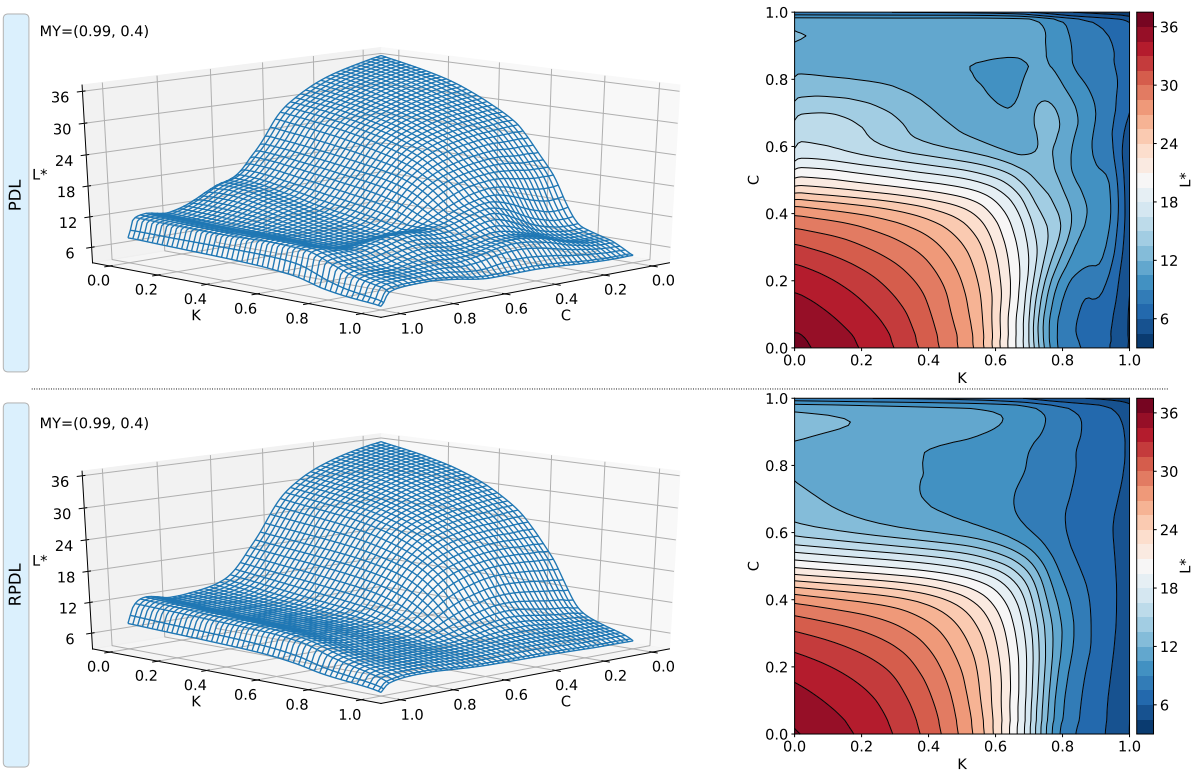


Figure B.17.: L^* as a function of K and C with other materials fixed on a tonal case of the *Mimaki 2* dataset

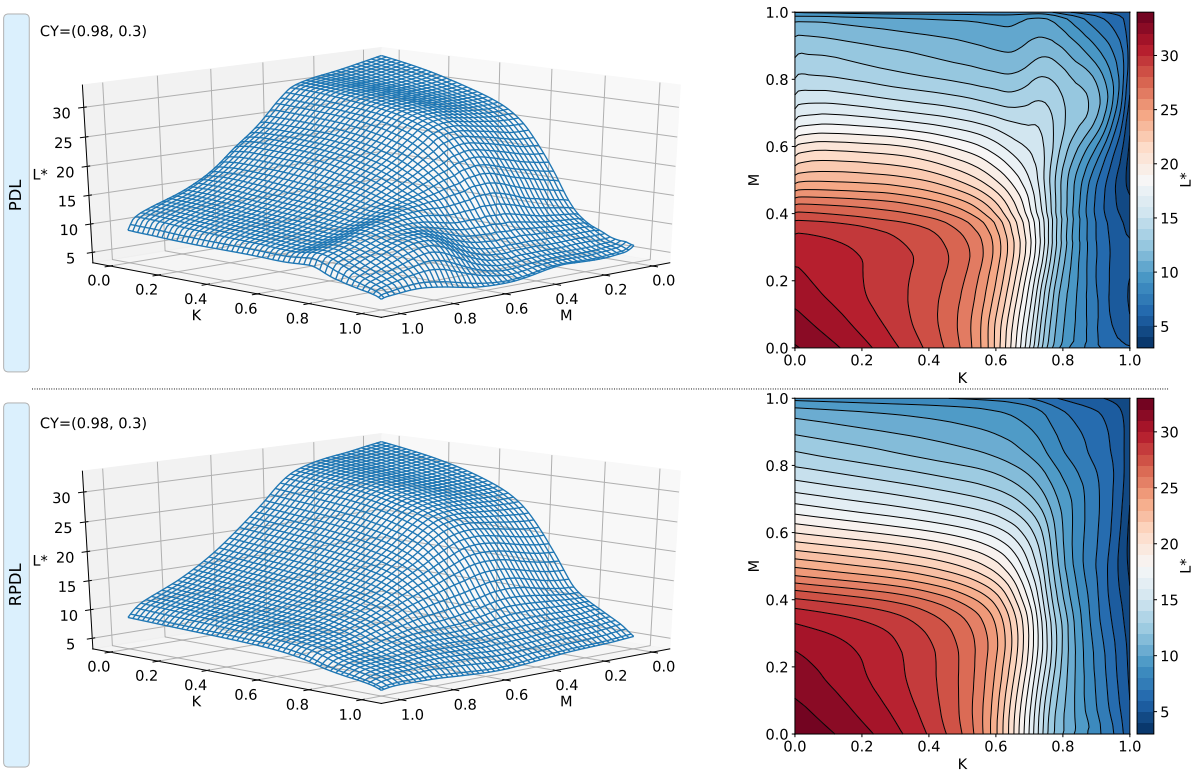


Figure B.18.: L^* as a function of K and M with other materials fixed on a tonal case of the *Mimaki 2* dataset

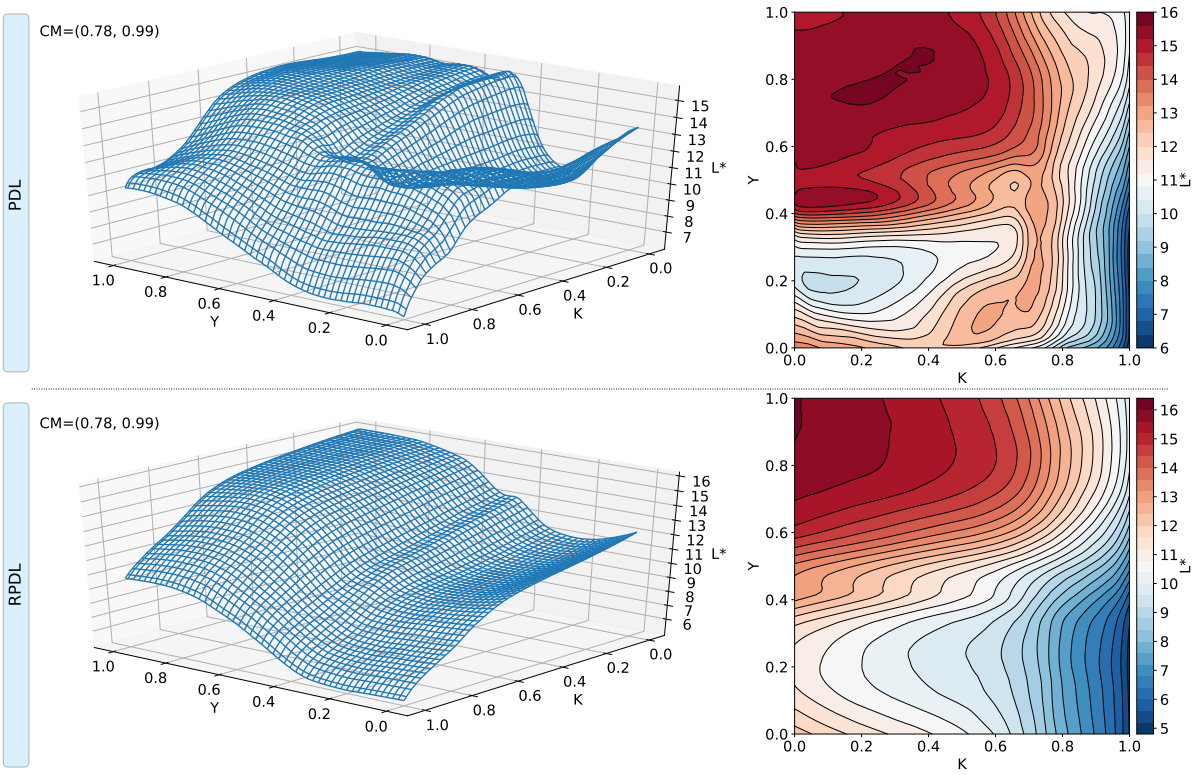


Figure B.19.: L^* as a function of K and Y with other materials fixed on a tonal case of the *Mimaki 2* dataset

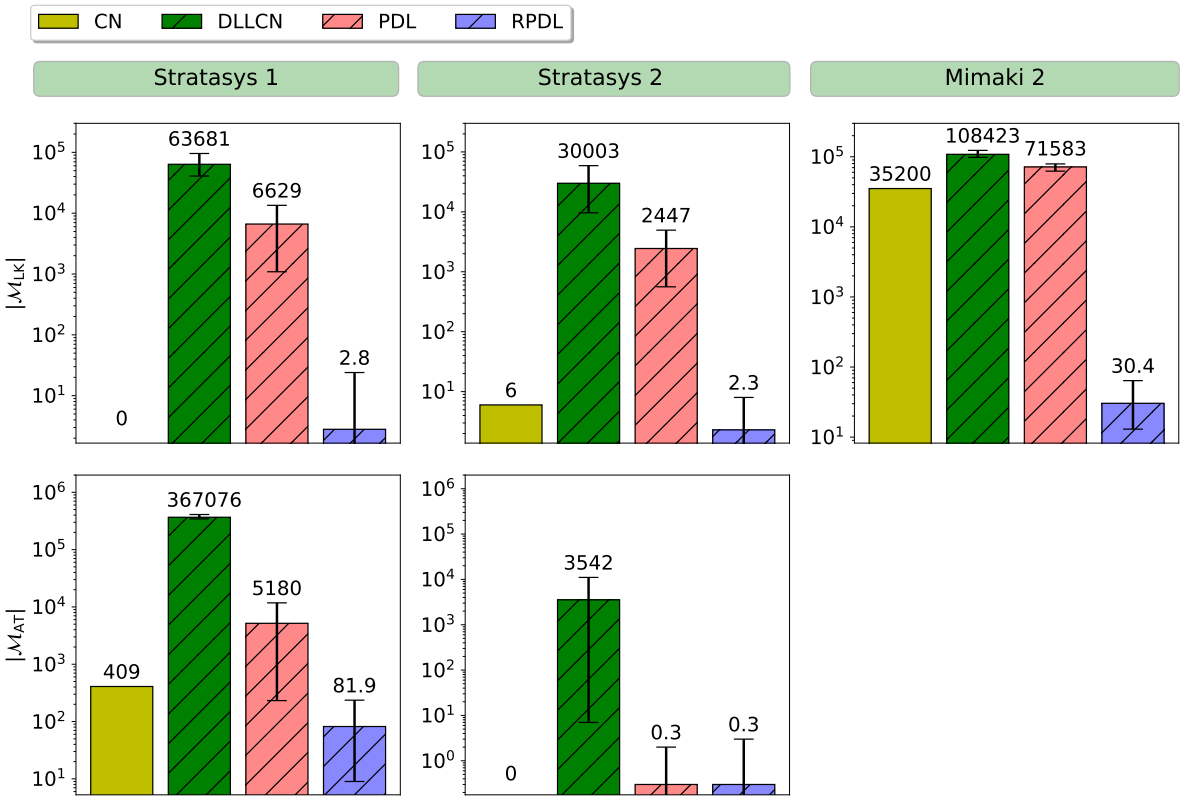


Figure B.20.: Bar plots of monotonicity violations (L^*-vs-K and α-vs-T). Each quantitative result shown above the corresponding bar is the averaged performance through 10 models, with a whisker to show the maximum and the minimum. Each row is set to have a unified y-axis display range for better comparison.

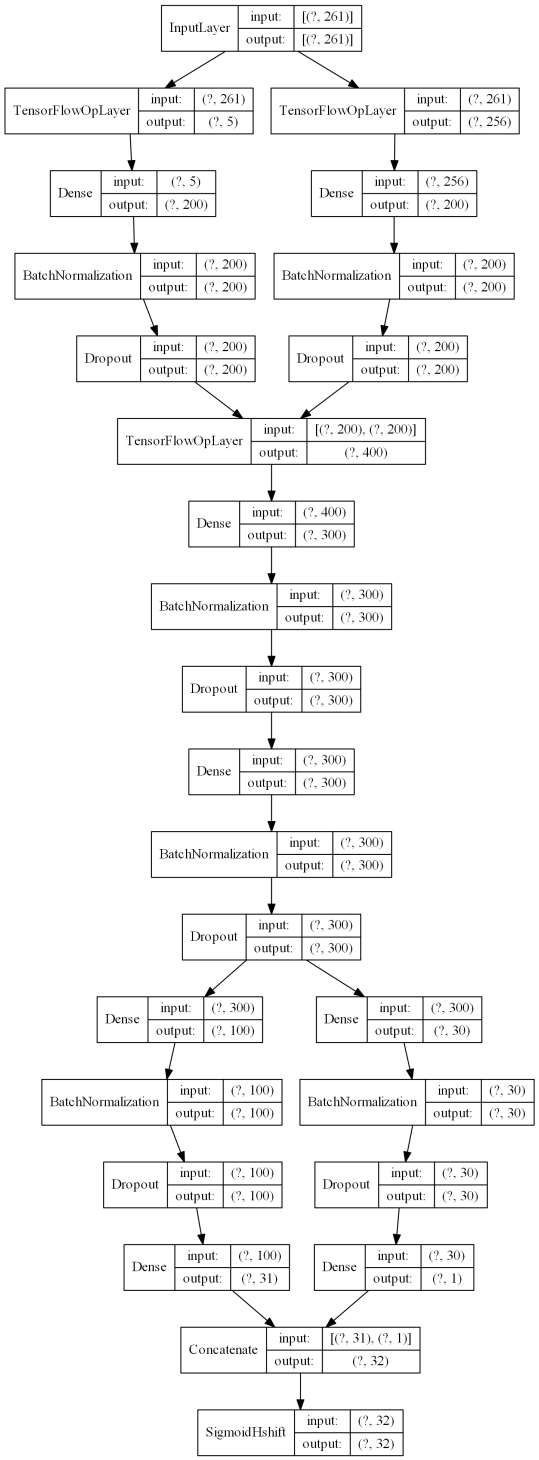


Figure B.21.: The detailed structure of the neural network used by the MPDL framework. Please zoom in for best view.

C. Publications

The thesis is partially based on the following first-authored publications:

C.1. Publications

1. CHEN D., URBAN P.: Deep learning models for optically characterizing 3d printers. *Optics Express 29*, 2 (Jan 2021), 615–631.
2. CHEN D., URBAN P.: Inducing robustness and plausibility in deep learning optical 3d printer models. *Optics Express 30*, 11 (May 2022), 18119–18133.
3. CHEN D., URBAN P.: Multi-printer learning framework for efficient optical printer characterization. *Optics Express 31*, 8 (Apr 2023), 13486–13502.

D. Curriculum Vitae

Personal Data

Name	Danwu Chen
Birth date & place	05.04.1988 in Zhanjiang
Family status	Married
Nationality	China
Email	danwu.chen@igd.fraunhofer.de

Education

2011 – 2013	M.Sc. in Telecommunications, Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan
2007 – 2011	B.Sc. in Information Engineering, South China University of Technology, Guangzhou, Guangdong, China

Work Experience

2019 – current Researcher, Fraunhofer Institute for Computer Graphics Research IGD, Darmstadt, Germany

- Early-Stage Researcher fellowship (2019 – 2022) under European Union's H2020 Marie Skłodowska-Curie Actions (Grant agreement ID: 813170). https://cordis.europa.eu/project/id/813170

2014 – 2019 Software Engineer, ASML, Shenzhen, China

Patent

1. MACHINE LEARNING BASED IMAGE GENERATION FOR MODEL BASE ALIGNMENTS, WO2021083608A1, Issued May 6, 2021, https://patents.google.com/patent/WO2021083608A1
2. Tonal Vector Determination for Printing Device Control, US20230236570A1, Publication July 27, 2023, https://patents.google.com/patent/US20230236570A1

Bibliography

[AA98] AGAR A. U., ALLEBACH J. P.: An iterative cellular ynsn method for color printer characterization. In *IS&T/SID* (Scottsdale Ariz., 1998), pp. 197–200. 21, 22, 25, 83

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 51

[ABTU15] ARIKAN C. A., BRUNTON A., TANKSALE T. M., URBAN P.: Color-managed 3d-printing with highly translucent printing materials. In *SPIE/IS&T Electronic Imaging Conference* (San Francisco, 2015), pp. 9398 – 9398 – 9. 5, 25

[AM94] ABET S., MARCU G.: A neural network approach for rgb to ymck color conversion. In *TENCON'94-1994 IEEE Region 10's 9th Annual International Conference on:'Frontiers of Computer Technology'* (1994), IEEE, pp. 6–9. 21, 23

[AMMIL12] ABU-MOSTAFA Y. S., MAGDON-ISMAIL M., LIN H.: *Learning from Data: A Short Course*. AMLBook.com, 2012. 52

[AOS*16] AMODEI D., OLAH C., STEINHARDT J., CHRISTIANO P., SCHULMAN J., MANÉ D.: Concrete problems in ai safety, 2016. 64

[AST11] ASTM: ASTM E 1767: Standard Practice for Specifying the Geometry of Observations and Measurements for Characterizing the Appearance of Materials, 2011. 5

[BATU18] BRUNTON A., ARIKAN C. A., TANKSALE T. M., URBAN P.: 3d printing spatially varying color and translucency. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 157:1–157:13. 22, 25, 33, 34, 92, 94, 95

[BAU15] BRUNTON A., ARIKAN C. A., URBAN P.: Pushing the limits of 3d color printing: Error diffusion with translucent materials. *ACM Transactions on Graphics (TOG) 35*, 1 (2015), 4. 5, 25

[BBC] Hackney man first to receive 3d-printed prosthetic eye. *BBC*. 1

[BBC*23] BROHAN A., BROWN N., CARBAJAL J., CHEBOTAR Y., CHEN X., CHOROMANSKI K., DING T., DRIESS D., DUBEY A., FINN C., FLORENCE P., FU C., ARE-

NAS M. G., GOPALAKRISHNAN K., HAN K., HAUSMAN K., HERZOG A., HSU J., ICHTER B., IRPAN A., JOSHI N., JULIAN R., KALASHNIKOV D., KUANG Y., LEAL I., LEE L., LEE T.-W. E., LEVINE S., LU Y., MICHALEWSKI H., MORDATCH I., PERTSCH K., RAO K., REYMANN K., RYOO M., SALAZAR G., SANKETI P., SERMANET P., SINGH J., SINGH A., SORICUT R., TRAN H., VANHOUCKE V., VUONG Q., WAHID A., WELKER S., WOHLHART P., WU J., XIA F., XIAO T., XU P., XU S., YU T., ZITKOVICH B.: Rt-2: Vision-language-action models transfer web knowledge to robotic control. 17

[beh] Behind the scenes at laika's wildly imaginative new stop motion movie, missing link. *TechCrunch*. 4

[Ber00] BERNS R. S.: *Billmeyer and Saltzman's: Principles of Color Technology*, 3 ed. John Wiley & Sons, Inc., New York, 2000. 7, 32

[BH16] BABAEI V., HERSCH R. D.: *n*-ink printer characterization with barycentric subdivision. *IEEE Transactions on Image Processing 25*, 7 (2016), 3023–3031. 21, 22

[BJTM22] BAHRI D., JIANG H., TAY Y., METZLER D.: Scarf: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations* (2022). 71, 72, 73

[Bot99] BOTTOU L.: *On-Line Learning and Stochastic Approximations*. Cambridge University Press, USA, 1999, p. 9–42. 18

[Car97] CARUANA R.: Multitask learning. *Machine learning 28*, 1 (1997), 41–75. 31

[CFGH20] CHEN X., FAN H., GIRSHICK R., HE K.: Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297* (2020). 73

[Cha] All you need to know about chatgpt, the a.i. chatbot that's got the world talking and tech giants clashing. *CNBC*. 20

[Cha60] CHANDRASEKHAR S.: *Radiative transfer*. Courier Dover Publications, 1960. 4

[Cho17] CHOLLET F.: *Deep Learning with Python*, 1st ed. Manning Publications Co., USA, 2017. 18

[CIE01] CIE PUBLICATION NO. 142: *Improvement to Industrial Colour-Difference Evaluation*. Tech. rep., Central Bureau of the CIE, Vienna, Austria, 2001. 22, 32

[CKNH20] CHEN T., KORNBLITH S., NOROUZI M., HINTON G.: A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), III H. D., Singh A., (Eds.), vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 1597–1607. 71, 72, 73

[CNN] British man given 3d printed eye in world first, hospital says. *CNN*. 1

[CU] CHEN D., URBAN P.: Python code for PDL and DLLCN models, figshare (2020), https://osapublishing.figshare.com/s/c0a91c79bfbb24b6736d. 33

[CU21] CHEN D., URBAN P.: Deep learning models for optically characterizing 3d printers. *Opt. Express 29*, 2 (Jan 2021), 615–631. 6, 8, 25, 45, 53, 54, 62, 63, 66, 74, 97

[CU22] CHEN D., URBAN P.: Inducing robustness and plausibility in deep learning optical 3d printer models. *Opt. Express 30*, 11 (May 2022), 18119–18133. 45, 63, 66, 74, 76, 81, 85

[CU23] CHEN D., URBAN P.: Multi-printer learning framework for efficient optical printer characterization. *Opt. Express 31*, 8 (Apr 2023), 13486–13502. 63

[CY53] CLAPPER F., YULE J.: The effect of multiple internal reflections on the densities of half-tone prints on paper. *JOSA 43*, 7 (1953), 600–603. 21, 22

[CY55] CLAPPER F., YULE J.: Reproduction of color with halftone images. In *Proc. Seventh Ann. Tech. Meet. TAGA* (1955), pp. 1–14. 21, 22

[DBK*21] DOSOVITSKIY A., BEYER L., KOLESNIKOV A., WEISSENBORN D., ZHAI X., UNTERTHINER T., DEHGHANI M., MINDERER M., HEIGOLD G., GELLY S., USZKOREIT J., HOULSBY N.: An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 20

[DCLT19] DEVLIN J., CHANG M.-W., LEE K., TOUTANOVA K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4171–4186. 17, 20

[DDS*09] DENG J., DONG W., SOCHER R., LI L.-J., LI K., FEI-FEI L.: Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255. 17, 19, 71, 72

[Dem24a] DEMICHEL E.: Le procédé. *26*, 3 (1924), 17–21. 22

[Dem24b] DEMICHEL E.: Le procédé. *26*, 4 (1924), 26–27. 22

[DWP*10a] DONG Y., WANG J., PELLACINI F., TONG X., GUO B.: Fabricating spatially-varying subsurface scattering. *ACM TOG (Proc. SIGGRAPH) 29*, 4 (2010). 6, 22

[DWP*10b] DONG Y., WANG J., PELLACINI F., TONG X., GUO B.: Fabricating spatially-varying subsurface scattering. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 62:1–62:10. 25

[FB05] FLEMING R. W., BÜLTHOFF H.: Low-level image cues in the perception of translucent materials. *ACM Transactions on Applied Perception (TAP) 2*, 3 (2005), 346–382. 6

[Fraa] Fraunhofer technology revolutionises 3d printing of prosthetic eyes. *Fraunhofer Institute for Computer Graphics Research*. 1, 49

[frab] U.s. movie relies on german 3d printing technology. *Fraunhofer Institute for Computer Graphics Research*. 4

[GB10] GLOROT X., BENGIO Y.: Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 249–256. 28, 31, 36

[GBC16] GOODFELLOW I., BENGIO Y., COURVILLE A.: *Deep learning*, vol. 1. MIT press Cambridge, 2016. 13, 14, 16, 18, 31, 32, 52, 55

[GPAM*14] GOODFELLOW I. J., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial networks, 2014. 17, 19

[Gro17] GRON A.: *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st ed. O'Reilly Media, Inc., 2017. 18

[HC05] HERSCH R., CRÉTÉ F.: Improving the yule-nielsen modified spectral neugebauer model by dot surface coverages depending on the ink superposition conditions. In *Proc. SPIE* (2005), vol. 5667, pp. 434–445. 21, 22

[HCE03] HERSCH R., COLLAUD F., EMMEL P.: Reproducing color images with embedded metallic patterns. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 427–434. 21, 22

[HECC05] HERSCH R., EMMEL P., COLLAUD F., CRÉTÉ F.: Spectral reflection and dot surface prediction models for color halftone prints. *Journal of Electronic Imaging 14* (2005), 33001–12. 21, 22

[Hera] Hero forge color print : Review! *DeadAussieGamer*. 3

[Herb] Hero forge kickstarter update. *Hero Forge*. 3

[HF13] HENSLEY B. D., FERWERDA J. A.: Colorimetric characterization of a 3d printer with a spectral model. In *Color and Imaging Conference* (2013), Society for Imaging Science and Technology, pp. 160–166. 21, 22

[HFM*10a] HAŠAN M., FUCHS M., MATUSIK W., PFISTER H., RUSINKIEWICZ S.: Physical reproduction of materials with specified subsurface scattering. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 61:1–61:9. 25

[HFM*10b] HAŠAN M., FUCHS M., MATUSIK W., PFISTER H., RUSINKIEWICZ S.: Physical reproduction of materials with specified subsurface scattering. *ACM TOG (Proc. SIGGRAPH) 29*, 3 (2010). 6, 22

[HFW*20] HE K., FAN H., WU Y., XIE S., GIRSHICK R.: Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020). 71, 72, 73

[HH15] HÉBERT M., HERSCH R. D.: Review of spectral reflectance models for halftone prints: principles, calibration, and prediction accuracy. *Color Research & Application 40*, 4 (2015), 383–397. 21, 22

[HS97] HOCHREITER S., SCHMIDHUBER J.: Long short-term memory. *Neural computation 9* (12 1997), 1735–80. 17, 19

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. 28

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778. 17, 19

[ICC10] ICC: *File Format for Color Profiles*, 4.3.0.0 ed. International Color Consortium, http://www.color.org, 2010. 32

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015). 31

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 4

[KB14] KINGMA D., BA J.: Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014). 18, 36

[KM31] KUBELKA P., MUNK F.: Ein Beitrag zur Optik der Farbanstriche. *Zeitschrift für Technische Physik 12* (1931), 593–601. 21, 22

[Kob18] KOBAYASHI S.: Homemade bookcorpus. `https://github.com/soskek/bookcorpus`, 2018. 17

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (2012), Pereira F., Burges C., Bottou L., Weinberger K., (Eds.), vol. 25, Curran Associates, Inc. 17, 19

[KW22] KINGMA D. P., WELLING M.: Auto-encoding variational bayes, 2022. 17, 19

[LA01] LAU D. L., ARCE G. R.: *Modern digital halftoning*. CRC Press, 2001. 78

[LDS02] LITTLEWOOD D., DRAKOPOULOS P., SUBBARAYAN G.: Pareto-optimal formulations for cost versus colorimetric accuracy trade-offs in printer color management. *ACM Transactions on Graphics (TOG) 21*, 2 (2002), 132–175. 21, 23

[LHZL20] LIU X., HAN X., ZHANG N., LIU Q.: Certified monotonic neural networks. In *Advances in Neural Information Processing Systems* (2020), Larochelle H., Ranzato M., Hadsell R., Balcan M. F., Lin H., (Eds.), vol. 33, Curran Associates, Inc., pp. 15427–15438. 50, 60, 61

[LLD*19] LU J., LIU A., DONG F., GU F., GAMA J., ZHANG G.: Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering 31*, 12 (2019), 2346–2363. 65

[LME*23] LI Z., MÜLLER T., EVANS A., TAYLOR R. H., UNBERATH M., LIU M.-Y., LIN C.-H.: Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2023). 19

[MAA*08] MOROVIČ J., ALBARRAN A., ARNABAT J., RICHARD Y., MARIA M.: Accuracy-preserving smoothing of color transformation luts. In *Color and Imaging Conference*

(2008), vol. 2008, Society for Imaging Science and Technology, pp. 243–246. 49

[Mal02] MALOUF R.: A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20* (USA, 2002), COLING-02, Association for Computational Linguistics, p. 1–7. 73

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph. 41*, 4 (July 2022), 102:1–102:15. 19

[MHN13] MAAS A. L., HANNUN A. Y., NG A. Y.: Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (2013), vol. 30, p. 3. 19, 26

[MKS*15] MNIH V., KAVUKCUOGLU K., SILVER D., RUSU A. A., VENESS J., BELLEMARE M. G., GRAVES A., RIEDMILLER M., FIDJELAND A. K., OSTROVSKI G., PETERSEN S., BEATTIE C., SADIK A., ANTONOGLOU I., KING H., KUMARAN D., WIERSTRA D., LEGG S., HASSABIS D.: Human-level control through deep reinforcement learning. *Nature 518*, 7540 (Feb. 2015), 529–533. 17

[Mot10] MOTOYOSHI I.: Highlight–shading relationship as a cue for the perception of translucent and transparent materials. *Journal of vision 10*, 9 (2010), 6–6. 6

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 19

[Mul85] MULLEN K. T.: The contrast sensitivity of human colour vision to red-green and blue-yellow chromatic gratings. *The Journal of Physiology 359*, 1 (1985), 381. 52

[Mur36] MURRAY A.: Monochrome reproduction in photoengraving. *Journal of the Franklin Institute 221* (1936), 721–744. 21

[Nes83] NESTEROV Y.: A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. 18

[Neu05] NEUGEBAUER H. E. J.: The Theoretical Basis of Multicolor Letterpress Printing (Translated D. Wyble and A. Kraushaar). *Color Research and Application 30* (2005), 322–331. 21, 22

[NW06] NOCEDAL J., WRIGHT S.: *Numerical Optimization*. Springer New York, 2006. 17

[OLV18] OORD A. V. D., LI Y., VINYALS O.: Representation learning with contrastive predictive coding, 2018. 73

[Ope23] OPENAI: Gpt-4 technical report. arXiv:2303.08774, 2023. 17, 20

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 19

[Pol64] POLYAK B.: Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics 4*, 5 (1964), 1–17. 18

[RB93] ROLLESTON R., BALASUBRAMANIAN R.: Accuracy of Various Types of Neugebauer Model. In *IS&T/SID* (Scottsdale Ariz., 1993), pp. 32–36. 21, 22

[RBL*22] ROMBACH R., BLATTMANN A., LORENZ D., ESSER P., OMMER B.: High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022). 17, 19

[RHW86] RUMELHART D. E., HINTON G. E., WILLIAMS R. J.: *Learning representations by back-propagating errors*, vol. 323. Nature, 1986. 18

[RN20] RUSSELL S., NORVIG P.: *Artificial intelligence : a modern approach*. Pearson, 2020. 13, 14, 16

[Rog00] ROGERS G.: A generalized clapper–yule model of halftone reflectance. *Color Research & Application 25*, 6 (2000), 402–407. 21, 22

[RR10] R. ROSSIER R. H.: Introducing ink spreading within the cellular Yule-Nielsen modified Neugebauer model. In *IS&T/SID, 18th Color Imaging Conference* (San Antonio, Texas, 2010), pp. 295–300. 21, 22, 25, 83

[Sau42] SAUNDERSON J.: Calculation of the color of pigmented plastics. *JOSA 32*, 12 (1942), 727–729. 21, 22

[SBK*19] SHI L., BABAEI V., KIM C., FOSHEY M., HU Y., SITTHI-AMORN P., RUSINKIEWICZ S., MATUSIK W.: Deep multispectral painting reproduction via multi-layer, custom-ink printing. *ACM Transactions on Graphics (TOG) 37*, 6 (2019), 271. 21, 23

[SHHM16] SIMONOT L., HERSCH R. D., HÉBERT M., MAZAURIC S.: Multilayer four-flux matrix model accounting for directional-diffuse light transfers. *Applied optics 55*, 1 (2016), 27–37. 21, 23

[SHK*14] SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R.: Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*, 1 (2014), 1929–1958. 31, 55

[SHM*16] SILVER D., HUANG A., MADDISON C. J., GUEZ A., SIFRE L., VAN DEN DRIESSCHE G., SCHRITTWIESER J., ANTONOGLOU I., PANNEERSHELVAM V., LANCTOT M., DIELEMAN S., GREWE D., NHAM J., KALCHBRENNER N., SUTSKEVER I., LILLICRAP T., LEACH M., KAVUKCUOGLU K., GRAEPEL T., HASSABIS D.: Mastering the game of go with deep neural networks and tree search. *Nature 529* (2016), 484–503. 17

[SJ86] SCHLIMMER J. C., JR. R. H. G.: Incremental learning from noisy data. *Machine Learning* (1986). 65

[SMB*20] SITZMANN V., MARTEL J. N., BERGMAN A. W., LINDELL D. B., WETZSTEIN G.: Implicit neural representations with periodic activation functions. In *Proc. NeurIPS* (2020). 19

[SRB*19] SUMIN D., RITTIG T., BABAEI V., NINDEL T., WILKIE A., DIDYK P., BICKEL B., KŘIVÁNEK J., MYSZKOWSKI K., WEYRICH T.: Geometry-aware scattering compensation for 3d printing. *ACM Transactions on Graphics 38*, 4 (2019). 4, 25

[SZ15] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition, 2015. 17, 19

[Tom96] TOMINAGA S.: Color control using neural networks and its application. In *Color Imaging: Device-Independent Color, Color Hard Copy, and Graphic Arts* (1996), vol. 2658, International Society for Optics and Photonics, pp. 253–260. 21, 23

[TRB06] TSUTSUMI S., ROSEN M., BERNS R.: Spectral Reproduction Using LabPQR: Inverting the Fractional-Area-Coverage-to-Spectra Relationship. In *ICIS* (Rochester, NY, 2006), IS&T, pp. 107–110. 49, 86

[TRB08] TSUTSUMI S., ROSEN M. R., BERNS R. S.: Spectral color management using interim connection spaces based on spectral decomposition. *Color Research & Application 33*, 4 (2008), 282–299. 33

[UG06] URBAN P., GRIGAT R.-R.: Spectral-Based Color Separation using Linear Regression Iteration. *Color Research and Application 31* (2006), 229–238. 49, 86

[UR07] URBAN P., ROSEN M. R.: Inverting the Cellular Yule-Nielsen modified Spectral Neugebauer Model. In *Ninth International Symposium on Multispectral Color Science and Application* (Taipei, Taiwan, 2007), pp. 29–35. 49, 86

[URB07] URBAN P., ROSEN M. R., BERNS R. S.: Accelerating Spectral-Based Color Separation within the Neugebauer Subspace. *Journal of Electronic Imaging 16* (2007), 043014. 49, 86

[USD14] URBAN P., STAHL S., DÖRSAM E.: Image display-printing (desktop, commercial). In *Academic Press Library in Signal Processing*, vol. 4. Elsevier, 2014, pp. 117–163. 7

[UTB*19] URBAN P., TANKSALE T. M., BRUNTON A., VU B. M., NAKAUCHI S.: Redefining A in RGBA: Towards a standard for graphical 3d printing. *ACM Transactions on Graphics (TOG) 38*, 3 (2019), 1–14. 6, 8, 33, 34, 46, 55, 78, 92, 93

[VB67] VAN NES F. L., BOUMAN M. A.: Spatial modulation transfer in the human eye. *JOSA 57*, 3 (1967), 401–406. 52

[Vig90] VIGGIANO J.: Modeling the Color of Multi-color Halftones. In *TAGA Proceedings* (1990), pp. 44–62. 21, 22

[VSAOS16a] VAN SONG T. P., ANDRAUD C., ORTIZ SEGOVIA M. V.: Implementation of the four-flux model for spectral and color prediction of 2.5 d prints. In *NIP & Digital Fabrication Conference* (2016), vol. 2016, IS&T, pp. 26–30. 21, 23

[VSAOS16b] VAN SONG T. P., ANDRAUD C., ORTIZ-SEGOVIA M. V.: Towards spectral prediction of 2.5 d prints for soft-proofing applications. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)* (2016), IEEE, pp. 1–6. 21, 23

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER L. U., POLOSUKHIN I.: Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), Guyon I., Luxburg U. V., Bengio S., Wallach H., Fergus R., Vishwanathan S., Garnett R., (Eds.), vol. 30, Curran Associates, Inc. 17, 19

[WB00] WYBLE D. R., BERNS R. S.: A Critical Review of Spectral Models Applied to Binary Color Printing. *Color Research and Application 25*, 1 (2000), 4–19. 8, 21, 66

[WXSL18] WU Z., XIONG Y., STELLA X. Y., LIN D.: Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018). 73

[XHL16] XU B., HUANG R., LI M.: Revise saturated activation functions. *arXiv preprint arXiv:1602.05980* (2016). 28

[YC51] YULE J. A. C., COLT R. S.: Colorimetric Investigations in Multicolor Printing. In *TAGA Proceedings* (1951), pp. 77–82. 21, 22

[YKOI11] YOSHIDA K., KOMEDA N., OJIMA N., IWATA K.: Simple and effective method for measuring translucency using edge loss: optimization of measurement conditions and applications for skin. *Journal of biomedical optics 16*, 11 (2011), 117003–1170038. 5, 6

[YN51] YULE J. A. C., NIELSEN W. J.: The penetration of light into paper and its effect on halftone reproduction. In *Tech. Assn. Graphic Arts* (1951), vol. 4, pp. 65–76. 21, 22

[YZLL22] YANG J., ZHOU K., LI Y., LIU Z.: Generalized out-of-distribution detection: A survey, 2022. 64

[ZC18] ZHENG A., CASARI A.: *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018. 28

[ZHF*18] ZOLLER C. J., HOHMANN A., FORSCHUM F., GEIGER S., GEIGER M., ERTL T. P., KIENLE A.: Parallelized monte carlo software to efficiently simulate the light propagation in arbitrarily shaped objects and aligned scattering media. *Journal of biomedical optics 23*, 6 (2018), 065004. 4, 21

[ZNS*12] ZHANG J., NACHLIELI H., SHAKED D., SHIFFMAN S., ALLEBACH J. P.: Psychophysical evaluation of banding visibility in the presence of print content. In *Image Quality and System Performance IX* (2012), vol. 8293, International Society for Optics and Photonics, p. 82930S. 49