Optics EXPRESS

Extrapolating from lens design databases using deep learning

GEOFFROI CÔTÉ,[1,2,*] JEAN-FRANÇOIS LALONDE,[2] AND SIMON THIBAULT[1]

[1] *Centre d'optique, photonique et laser, Université Laval, Quebec City, G1V 0A6, Canada*
[2] *Laboratoire de vision et systèmes numériques, Université Laval, Quebec City, G1V 0A6, Canada*
[*] *geoffroi.cote.1@ulaval.ca*

Abstract: We propose for the first time a deep learning approach in assisting lens designers to find a lens design starting point. Using machine learning, lens design databases can be expanded in a continuous way to produce high-quality starting points from various optical specifications. A deep neural network (DNN) is trained to reproduce known forms of design (supervised training) and to jointly optimize the optical performance (unsupervised training) for generalization. In this work, the DNN infers high-performance cemented and air-spaced doublets that are tailored to diverse desired specifications after being fed with reference designs from the literature. The framework can be extended to lens systems with more optical surfaces.

1. Introduction

Lens design is a formidable optimization problem. Optical performance is heavily non-linear with respect to variables such as surface curvatures, refractive indices and thicknesses, and is characterized by many local minima and steep ridges [1]. Hard physical constraints (e.g. feasible glass thicknesses and non-overlapping lens elements) add to the challenge of guiding the optimization process.

Local search approaches return a lens design in the neighborhood of a given initial solution, thus their potential is gated by the suitability of the initial lens design used as a starting point. In contrast, global optimization methods search a more global space and explore many potential solutions. Examples of global methods include global optimization with escape function [2], genetic algorithms [3,4], simulated annealing [5] and particle swarm optimization [6]. All local and many global search-based lens design methods require the lens designer to find an initial design, or starting point. The match between the starting point and the desired specifications will more often than not dictate the failure or success of the optimization process using lens design software. With damped least-squares optimization, very close starting points can lead to vastly different minima after optimization [7]. Finding the right starting point is therefore a crucial and time-consuming task.

To obtain a good starting point, the traditional method consists of searching in a lens design database for a design with specifications similar to the desired ones, then reoptimizing it to new specifications. Since this is a time-consuming trial-and-error process, lens designers would benefit from an automated method of retrieving high-quality starting points that are tailored to the desired specifications using machine learning, which is the motivation for our work.

A few learning-based approaches have been applied to lens design [8,9] by using training data generated using commercial lens design software optimization. However, these approaches are designed to learn from abundant data which cannot be easily generated other than for simple lens systems. It is unclear how they could cope with the scarcity of expertly designed systems found in the literature. Another learning-based approach is a database-querying tool [10] in which a shallow neural network was trained to return the best-matching lens design of the database according to the requirements given as input.

#373725 https://doi.org/10.1364/OE.27.028279
Journal © 2019 Received 31 Jul 2019; revised 30 Aug 2019; accepted 4 Sep 2019; published 19 Sep 2019

Using the function approximation capabilities of deep learning, abstract patterns of what makes a good design could be learned and used in the creation of new forms of design. Deep learning, which usually refers to the use of neural networks with at least one intermediate (hidden) layer, allows the learning of high-quality abstract features and tendencies directly from the data. Neural networks are highly flexible and their architecture can be adapted to express strong priors on data. The spatial structure of data can be taken by account by convolutional neural networks (CNN) [11], e.g. in vision-related tasks [12], whereas recurrent neural networks (RNN) [13] can capture sequential information, e.g. in natural language processing [14,15]. Readers may refer to [16] for a comprehensive review on deep learning.

In this work, by using known design forms from a lens design database, we show that we can train a deep neural network (DNN) to infer two-lens telescopic objectives with high flexibility and performance, tailored to the desired first-order specifications. Specifically, a DNN is trained to take first-order specifications as input and output high-quality lens designs. Our approach can be seen as a generalization of the database-querying tool [10] where we extrapolate from known designs, even if provided in very small numbers, in a continuous manner instead of returning an entry from the database. To our knowledge, our work represents the first time that deep learning is used in assisting lens designers and a first attempt at extrapolating from lens design databases.

In Section 2, we present an overview of our method. Specifications input to the DNN include the entrance pupil diameter (EPD) and half field of view (HFOV) as well as the allowed boundaries for every glass or air thickness. With our method, starting points are obtained simply by entering the desired specifications into the DNN, which is a fully automatic process for lens designers provided the DNN is already trained. This process is much faster than the traditional method and does not depend on the lens designer's know-how, thus it can be useful to both the experienced and novice lens designer. The trained model and framework could be accessible for inference through a toolbox for lens designers, e.g. as an online service or as a part of some software.

In Section 3, we explain how we train the DNN. Most machine learning applications distinguish between *supervised training* in which the target model outputs are known ahead of time, and *unsupervised training* in which the learning task does not require any manually-labeled data [17]. To successfully train the DNN, our key idea is to combine both training schemes. With supervised training, the DNN is fed with known lens designs from the literature [18] and learns to map the input specifications to the reference designs. However, the modest amount of data contained in lens design databases is insufficient to allow the learning of high-quality abstract features. Thus, we generalize to new specifications through unsupervised training in which we directly optimize the optical performance of the inferred lens designs for a large set of randomly-generated input specifications. By implementing a differentiable ray-tracing module, we optimize the RMS spot size at the image plane and design our loss function (used to train the DNN) to avoid ray-tracing failures and overlapping surfaces. A previous version of the unsupervised training scheme was briefly introduced in our conference paper [19].

In Section 4, we test our framework and compare the inferred designs to reference designs from the literature. We test our framework over two types of telescopic objectives, namely cemented doublets and air-spaced doublets. We show that the inferred designs are viable starting points with optical performance on par with the designs from the literature over a wide range of specifications.

In Section 5, we discuss how our framework could be extended to provide significant advantages to the user: (1) obtain diversified starting points for a single set of specifications, (2) model a wider variety of components such as aspherical surfaces, (3) learn designs with varying sequences of glass elements and air gaps with a single network, which could help generalize high-level features of what makes a good lens design, and (4) use the framework in an interactive manner in which the user enforces one or more parameters such as the glass variables and the model infers the missing parts. In short, we believe that our approach paves the way for promising future work.

2. Deep learning framework

Neural networks are function approximators that map an input vector to an output vector. Training a neural network is achieved by minimizing a loss function over the dataset, using some form of stochastic gradient descent (SGD) algorithm. Backpropagation is the process of computing the derivative of the loss with respect to every neural network parameter. To that end, neural networks are usually implemented using an automatic differentiation library such as PyTorch [20]. Thus, every operation defined in the forward pass is recorded so that the differentiation chain rules are automatically applied when calling the backward pass. In most cases, a training step (or iteration) consists of (1) batching randomly selected training samples together to form a mini-batch, (2) operating a forward pass on the mini-batch and averaging the loss function over the mini-batch, (3) operating a backward pass through backpropagation of the loss, and (4) updating the model parameters with a SGD step.

In this work, we consider for simplicity infinite conjugate optical systems composed of spherical surfaces, the first of which is the aperture stop. Figure 1 illustrates the variables that must be inferred by our deep learning model in order to fully define the lens system. The focal length of the lens system is always normalized to one by solving for the last curvature of the system. Thus, every parameter that is expressed in units of distance is normalized for a unitary focal length.

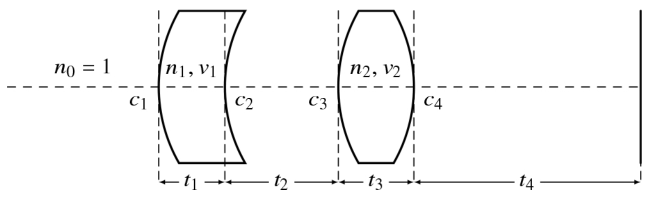

Fig. 1. Illustration of the variables used to describe an optical system in this work. Lens materials are modeled by the refractive index n and Abbe number v. Thicknesses t represent the distance between adjacent optical surfaces including the image plane. The surface curvatures are represented by c.

Our deep learning framework is represented in Fig. 2. Assuming the model is trained, the framework is used for inference by entering the input specifications into the model which directly produces a lens design as output. The number of inputs and outputs depends on the number of optical surfaces N and glass materials M. The input specifications include the entrance pupil diameter (EPD) and half field of view (HFOV) for which the lens system must be optimized. Since the focal length is normalized to 1, the EPD is simply the reciprocal of the f-number. The other input specifications are the minimum thicknesses $t_{1,\text{min}}, \ldots, t_{N,\text{min}}$ and thickness ranges $t_{1,\text{range}}, \ldots, t_{N,\text{range}}$, which serve to provide additional control over the inferred lens designs. For the air-spaced doublets represented in Fig. 1 ($N = 4$ and $M = 2$), the model takes 10 input specifications and infers 11 variables. For cemented doublets ($N = 3$), the model takes 8 input specifications and infers 9 variables.

The model is composed of two parts: a DNN with learnable parameters and a set of functions that convert the DNN outputs to valid, meaningful lens parameters. For this experiment we test a simple fully-connected DNN whose architecture is given in Appendix A. The DNN outputs are the curvatures $c_1, \ldots, c_{N-1}$, intermediate thickness variables $t_{1,raw}, \ldots, t_{N,\text{raw}}$ as well as the normalized glass variables $g_{1,1}, g_{1,2}, \ldots, g_{M,1}, g_{M,2}$. The thickness variables are then bounded within the desired range: $t_{i,\text{raw}}, t_{i,\text{min}}, t_{i,\text{range}} \longrightarrow t_i$, and the glass variables are converted as to approximate the behavior of real glasses: $g_{i,1}, g_{i,2} \longrightarrow n_i, v_i$. The exact procedure for the conversion is detailed in Appendix B.

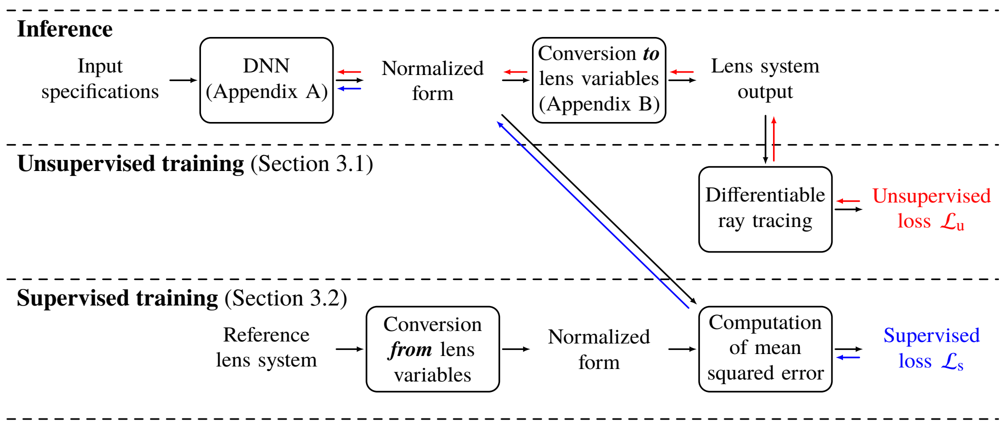
Inference
Input specifications
DNN (Appendix A)
Normalized form
Conversion to lens variables (Appendix B)
Lens system output
Unsupervised training (Section 3.1)
Differentiable ray tracing
Unsupervised loss $\mathcal{L}_u$
Supervised training (Section 3.2)
Reference lens system
Conversion from lens variables
Normalized form
Computation of mean squared error
Supervised loss $\mathcal{L}_s$

Fig. 2. Overview of the deep learning framework. Boxes refer to operators and plain text to data. When inferring lens systems, only the top part (labeled "inference") is necessary. To train the DNN, the unsupervised or supervised loss is computed, then backpropagated into the DNN through the corresponding colored arrows. In unsupervised training, the lens system performance is evaluated by computing the RMS spot size. In supervised training, the reference designs are converted to a normalized form then compared to the DNN outputs using the mean squared error.

3. Training procedure

Training the model is achieved by the careful mixture of two training schemes: unsupervised training to optimize the designs over varied specifications, and supervised training in which the model learns to reproduce known lens designs for appropriate specifications. In the unsupervised training pass, lens systems are inferred from randomly drawn sets of specifications, then the optical performance of the lens systems is estimated and optimized by computing the RMS spot size at the image plane. In the supervised training pass, the DNN operates on sets of specifications for which a reference design is available. The reference designs that correspond to those specifications must be normalized so they can be directly compared to the DNN outputs by computing the mean squared error (MSE).

3.1. Unsupervised loss function based on RMS spot size

The first goal of unsupervised training is to directly optimize the optical performance of the inferred lens systems. The second goal is to make sure that there are no overlapping surfaces and that the rays successfully pass through the system without experiencing total internal reflection (TIR) or missing a surface. The unsupervised loss function must output a single number representative of the optical performance of the lens system. RMS spot size is used as a default optimization target in most optical design software. Similarly, we use RMS spot size instead of other optical performance metrics, e.g. modulation transfer function (MTF) or Seidel aberrations, because it represents a good estimation of the optical performance for many non-diffraction-limited systems and it is straightforward to compute.

We implement a basic ray-tracing module using the automatic differentiation capabilities of PyTorch [20]. This allows the tracing of meridional rays through the entrance pupil which is located on the first surface of the system, and across subsequent spherical optical surfaces. Implementing our own ray-tracing module provides three clear advantages: (1) automatic differentiation to compute partial derivatives in a more precise and far quicker way than numerical differentiation, (2) parallel implementation that fully exploit GPU computing capabilities and (3) more knowledge and control over the ray paths which allow us to track and penalize both ray failures and overlapping surfaces. The ray-tracing formulas are included in Appendix C.

Ray-tracing failures occur if either the angle of incidence I or refraction I' cannot be computed. We can avoid rays that miss the surface and TIR by enforcing that $|\sin(I)|<1$ and $|\sin(I')|<1$, respectively. Overlapping surfaces can be accounted for by computing the longitudinal displacement Δz of the ray between two subsequent surfaces and by enforcing that $\Delta z>0$.

We compute the RMS spot size from the standard deviation of the tangential ray intersections y_{Hwp} at the image plane for a given field angle among 0, 0.7 and 1 times the HFOV, where we index the field angles with H, wavelengths with w and pupil heights with p. For each field angle, we propagate rays towards 17 equally spaced pupil heights for the standard C, d and F wavelengths, then we average the results over all field angles:

$$s = \frac{1}{N_H} \sum_H \sqrt{\frac{1}{N_w N_p} \sum_{w,p} (y_{Hwp} - \overline{y_H})^2} \, . \tag{1}$$

Next, we account for ray-tracing failures and overlapping surfaces by deriving a continuous everywhere and piecewise continuously differentiable penalty term q_{Hwp}. When overlapping surfaces or ray failures occur, a non-zero derivative is necessary to propagate the learning signal due to the penalty to the DNN. Our ray-wise penalty term is the following where we sum over all optical interfaces (k) encountered by the ray:

$$q_{Hwp} = \frac{1}{N_k} \sum_k \max\left(|\sin(I_{Hwpk})| - 1, 0\right) + \max\left(|\sin(I'_{Hwpk})| - 1, 0\right) + \max\left(\Delta z_{Hwpk}, 0\right) \, . \tag{2}$$

Note that the penalty term is equal to 0 if the ray hits the image plane while never traveling backward, and positive otherwise.

Our unsupervised loss function is the following combination of the mean spot size s and mean penalty term q:

$$\mathcal{L}_\mathrm{u} = \sqrt{s\left(1 + \lambda_\mathrm{q} \frac{1}{N_H N_w N_p} \sum_{H,w,p} q_{Hwp}\right)} \, , \tag{3}$$

where λ_p is empirically set to 10^3 to scale the penalty term. The square root is not strictly required but helps in weighing all training samples evenly during training. Otherwise, systems with larger aperture and field of view would generate more training signal due to their larger RMS spot size.

3.2. Supervised loss function

Given the reference design, that is, a set of curvatures, thicknesses, refractive indices and Abbe numbers, we first convert the thicknesses and glass materials to a normalized form by computing the reciprocal of the functions detailed in Appendix B: $t_i, t_{i,\mathrm{min}}, t_{i,\mathrm{range}} \longrightarrow t_{i,\mathrm{raw}}$ and $n_i, v_i \longrightarrow g_{i,1}, g_{i,2}$. The reference lens design is now in a normalized form that can be directly compared to the DNN outputs. Then, the supervised loss function is the MSE between the DNN outputs and the target lens variables, where the latter are decorated by primes:

$$\mathcal{L}_\mathrm{s} = \frac{\sum_{i=1}^{N-1} (c_i - c'_i)^2 + \sum_{i=1}^{N} (t_{i,\mathrm{raw}} - t'_{i,\mathrm{raw}})^2 + \sum_{i=1}^{M} (g_{i,1} - g'_{i,1})^2 + \sum_{i=1}^{M} (g_{i,2} - g'_{i,2})^2}{2N + 2M - 1} \, . \tag{4}$$

3.3. Combining unsupervised and supervised training

We define hybrid training as the combination of unsupervised and supervised training. A straightforward method considered was to train the DNN with our supervised scheme, make sure that the DNN learns to match the reference designs, then generalize to other specifications with the unsupervised scheme. Doing so, however, results in the DNN "forgetting" part of what it learned during supervised training. In practice, we apply both training schemes simultaneously so that the training signal given to the DNN during backpropagation comes in similar proportions from both the supervised and unsupervised tasks.

The process for a hybrid training step is as follows. We randomly sample and batch sets of specifications for both learning schemes, which provide us with an unsupervised mini-batch and a supervised mini-batch. We compute the model outputs for all training samples and apply the respective loss functions. We average both the unsupervised and supervised loss functions over the respective mini-batches, then we compute the hybrid loss with a weighted sum where we set λ_s to 10 to balance both terms :

$$\mathcal{L}_h = \overline{\mathcal{L}_u} + \lambda_s \overline{\mathcal{L}_s} , \tag{5}$$

We then process a backward pass of the loss through backpropagation, and we update the DNN parameters using a variant of the common stochastic gradient descent algorithm. More training details for our specific experiments are given in Appendix D.

4. Experiments

As a case study, we train our model to infer optimized two-lens telescopic objectives for any set of specifications in a given continuous interval. We consider two lens design structures: cemented doublets (glass-glass-air, or GGA designs) and air-spaced doublets (GAGA designs). For the supervision signal, we select from the literature [18] 5 GGA reference designs with a HFOV ranging from 0.75° to 1.5° and a f-number from 2.0 to 6.0, and 8 GAGA-type lens designs with a HFOV ranging from 0.5° to 2.0° and a f-number from 2.8 to 14.0.

4.1. Dataset generation

An unsupervised training sample corresponds to a set of input specifications. A supervised training sample corresponds to a target, which is a reference lens design in its normalized form, as well as an appropriate set of specifications for this target.

In training the DNN, the specifications that are given as input are drawn from a uniform probability distribution. In unsupervised training, the minimum and maximum values of the distributions must be chosen so that they represent the values used in practice. In supervised training, the specifications must be the ones for which the reference designs were optimized for.

In supervised training, the specifications must be drawn appropriately and on an individual basis for every reference design used. The EPD and HFOV are fixed to those that are given in each reference design's prescription. In choosing the distribution for the minimum thicknesses t_{min} both the lower and upper bounds must be lower than the real thickness for a given surface and no less than zero. Likewise, the lower bound for the range thickness t_{range} must be greater than the real thickness. Specific values are given in Table 1. The key point is that supervised training focuses on specific combinations of EPD and HFOV whereas unsupervised training is used to generalize to a wider variety of specifications.

4.2. Results and discussion

4.2.1. Training loss

In Fig. 3 we show the evolution of both the mean RMS spot size and mean squared error during training on the GGA and GAGA lens designs using our hybrid training scheme. Every training step processes 1024 random unsupervised samples and 1024 random supervised samples, where the supervised samples are distributed equally among all reference lens designs. For completion, we also compare these results to those obtained using the same parameters but omitting the supervision signal. All experiments are reproduced over 3 random DNN parameter initializations.

Table 1. Minimum and maximum values of the uniform distributions used to draw the input specifications when training the DNN. The last line (in bold) for each type of sequence gives the values used for generalization with unsupervised training. All dimensions are given for an effective focal length of 1 and the HFOV is given in radians.

sequence		EPD	HFOV	$t_{1,\mathrm{min}}$	$t_{1,\mathrm{range}}$	$t_{2,\mathrm{min}}$	$t_{2,\mathrm{range}}$	$t_{3,\mathrm{min}}$	$t_{3,\mathrm{range}}$	$t_{4,\mathrm{min}}$	$t_{4,\mathrm{range}}$
GGA (1)	max	0.167	0.026	0.000	1.045	0.000	1.041	0.957	2.007	–	–
	min	0.167	0.026	0.000	0.045	0.000	0.041	0.000	1.007	–	–
GGA (2)	max	0.333	0.017	0.045	1.095	0.000	1.050	0.924	1.974	–	–
	min	0.333	0.017	0.000	0.095	0.000	0.050	0.000	0.974	–	–
GGA (3)	max	0.167	0.013	0.000	1.047	0.000	1.041	0.957	2.007	–	–
	min	0.167	0.013	0.000	0.047	0.000	0.041	0.000	1.007	–	–
GGA (4)	max	0.200	0.017	0.010	1.060	0.000	1.040	0.949	1.999	–	–
	min	0.200	0.017	0.000	0.060	0.000	0.040	0.000	0.999	–	–
GGA (5)	max	0.400	0.017	0.075	1.125	0.000	1.045	0.907	1.957	–	–
	min	0.400	0.017	0.000	0.125	0.000	0.045	0.000	0.957	–	–
GGA (U)	max	0.400	0.026	0.075	1.125	0.020	1.050	0.957	2.007	–	–
	min	0.167	0.013	0.000	0.045	0.000	0.040	0.000	0.957	–	–
GAGA (1)	max	0.248	0.035	0.025	1.075	0.000	1.035	0.005	1.055	0.906	1.956
	min	0.248	0.035	0.000	0.075	0.000	0.035	0.000	0.055	0.000	0.956
GAGA (2)	max	0.156	0.026	0.000	1.041	0.000	1.026	0.000	1.041	0.957	2.007
	min	0.156	0.026	0.000	0.041	0.000	0.026	0.000	0.041	0.000	1.007
GAGA (3)	max	0.071	0.009	0.000	1.036	0.000	1.025	0.000	1.034	0.967	2.017
	min	0.071	0.009	0.000	0.036	0.000	0.025	0.000	0.034	0.000	1.017
GAGA (4)	max	0.143	0.017	0.000	1.037	0.000	1.035	0.000	1.044	0.960	2.010
	min	0.143	0.017	0.000	0.037	0.000	0.035	0.000	0.044	0.000	1.010
GAGA (5)	max	0.355	0.017	0.045	1.095	0.007	1.057	0.005	1.055	0.850	1.900
	min	0.355	0.017	0.000	0.095	0.000	0.057	0.000	0.055	0.000	0.900
GAGA (6)	max	0.125	0.013	0.000	1.040	0.000	1.026	0.000	1.037	0.961	2.011
	min	0.125	0.013	0.000	0.040	0.000	0.026	0.000	0.037	0.000	1.011
GAGA (7)	max	0.125	0.009	0.000	1.042	0.000	1.026	0.000	1.037	0.954	2.004
	min	0.125	0.009	0.000	0.042	0.000	0.026	0.000	0.037	0.000	1.004
GAGA (8)	max	0.200	0.017	0.010	1.060	0.000	1.034	0.000	1.040	0.927	1.977
	min	0.200	0.017	0.000	0.060	0.000	0.034	0.000	0.040	0.000	0.977
GAGA (U)	max	0.355	0.035	0.045	1.095	0.020	1.057	0.020	1.055	0.967	2.017
	min	0.071	0.009	0.000	0.036	0.000	0.025	0.000	0.034	0.000	0.900

The inclusion of the supervision signal leads to a mean RMS spot size significantly better for the GAGA lens designs. Surprisingly, in the case of GGA designs the supervision signal actually leads to the same final performance. We hypothesize that since GGA designs have a smaller parameter space than the GAGA designs, the loss function landscape is simpler and the unsupervised optimization process has no trouble matching the performance of the reference GGA systems. We suspect that for more complex lens design structures (e.g. double-Gauss systems), the supervision signal will be mandatory as the optimization process without supervision would likely fall in a bad local minimum.

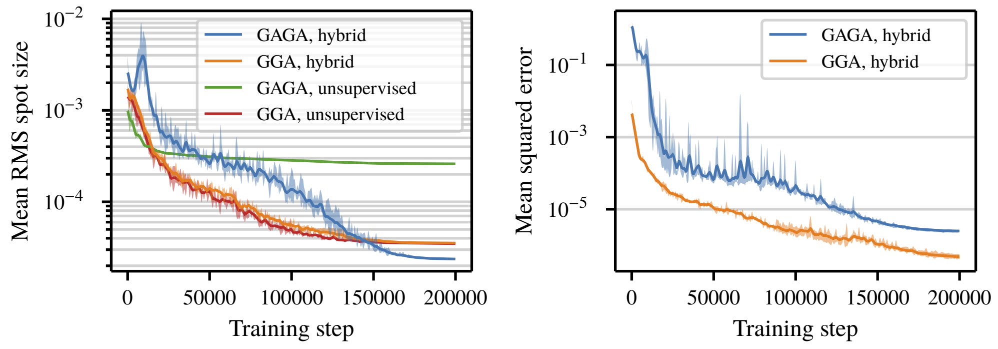

Fig. 3. Evolution of the mean normalized RMS spot size during training for the hybrid and unsupervised schemes and mean squared error for the hybrid scheme. For every curve we show the minimum and maximum over 3 experiments initialized with random DNN parameters. The mean RMS spot size cannot be directly compared between GGA and GAGA designs because the respective DNNs are trained on different input specifications.

4.2.2. Optical performance

Using the DNNs trained using our hybrid scheme for GGA and GAGA systems, we produce 100 000 additional sets of specifications based on the unsupervised dataset probability distribution (Table 1) and gather statistics on the lens designs inferred. Figure 4 shows the RMS spot size for varying EPD and HFOV. The color-encoded RMS spot size of the individual scatter points is Gaussian filtered with respect to the EPD and HFOV in producing this heat map. The RMS spot size is at least on par with that of the reference designs throughout most of the distribution of EPD and HFOV. The takeaway here is that the DNN learns to fill the gaps over specifications for which there is no reference design. For more complicated lens design structures such as double-Gauss designs, we believe that the DNN will learn to at least match the performance of the reference designs for the specifications close to the reference specifications, and perhaps for more varied specifications.

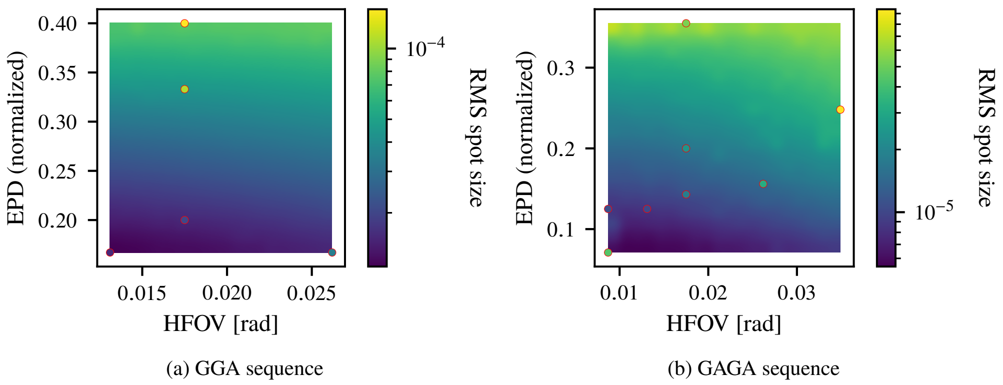

Fig. 4. Heat map of the average spot size for unitary focal length as a function of the entrance pupil diameter (EPD) and half field of view (HFOV) given as inputs to the DNN. The reference systems used for supervision are also shown as individual dots with their respective color-encoded RMS spot size. With the hybrid training scheme, the DNN learns to infer lens designs with an RMS spot size at least on par with that of the reference designs over most of the specifications considered.

4.2.3. Cross-validation experiment

To compare the optical performance of our inferred GAGA lens designs to the reference designs, we train the DNN with supervision from 7 of the 8 reference designs. Using the trained DNN, we infer a lens design using the same EPD and HFOV as the reference design that was not used for the training and we compare each other. We repeat this experiment over all reference designs. This is akin to the process referred to as leave-one-out cross-validation in the machine learning literature, which is used for testing the generalization performance of a model when there is a limited amount of training data. Figure 5 shows the 2D layouts and common aberration plots of both the inferred and reference designs.

As seen in the transverse error, astigmatism, distortion and lateral color plots, the optical performance of our designs is generally on par with that of the reference designs. Most of our designs fare better on the lateral color, but worse on the ray fan plots, which is most certainly the result of only optimizing for the RMS spot size. Moreover, we see from the 2D layouts that there are no overlapping surfaces or ray failures, which is attributed to our penalty term defined in Eq. (2). Not all inferred lens designs can beat their counterpart, but they are good starting points that are automatically obtained without manual searching or optimization. Interestingly, although there are reference designs of both Steinheil and Fraunhofer forms (which have similar residual aberrations [21]), the DNN favors the latter when extrapolating to new specifications.

5. Future work

The deep learning framework showed promising results on simple two-lens telescopic objectives which pave the way for exciting future work. Extending either the model, ray-tracing module or loss functions can lead to promising applications. Our approach can be extended straightforwardly to optical systems that feature only spherical surfaces, including many doublet, triplet, Tessar, Heliar, telephoto or double-Gauss lenses. Our ray-tracing module can be extended to account for conic, aspherical or other non-spherical surfaces.

A significant improvement to our work involves the use of a dynamic model such as a recurrent neural network (RNN) instead of the static DNN used in this work. A dynamic model with a recurrent structure could learn to infer lens variables in a sequential manner, which presents significant advantages over the current model. First, a single dynamic model can be trained to learn different lens design structures simultaneously whereas a single instance of our current model can only learn a specific structure, e.g. GGA or GAGA designs but not both. It remains an open question whether an RNN could learn high-level design practices from reference designs of different structures. Second, a dynamic model can capture the sequential structure of optical systems, which will presumably inject a favorable *a priori* into the training process and generalize better to new design forms. Third, one could use this framework interactively by fixing some lens design parameters, and the model could infer the remaining parameters. This could be helpful in many situations such as reusing stock lenses or fixing glass variables to real glasses.

In practice, the best lens design for a given application depends not only on the focal length, f-number and field of view, but also on several other variables that are not currently accounted for. Thus, another major future extension is the use of probabilistic models that account for the uncertainty of the data and training objective. With a probabilistic model, varied starting points could be inferred for a single set of specifications. Specifically for air-spaced doublets, both the Steinheil and Fraunhofer forms could be inferred for the same specifications.

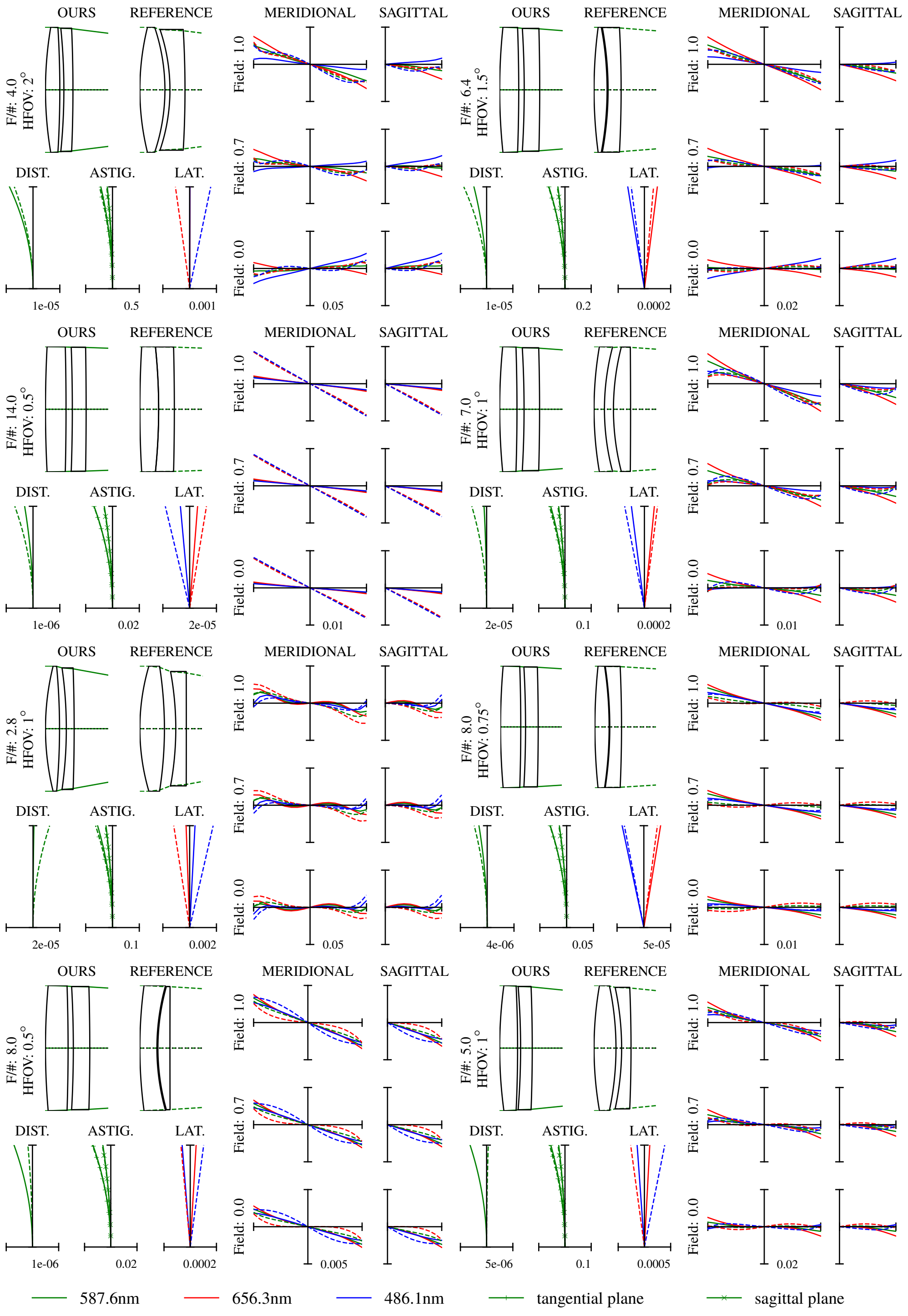

Fig. 5. Comparison of the lens designs inferred by our model to the reference designs from the literature. All systems are scaled to a 100 mm focal length. Dashed lines are used for the reference designs. In addition to each 2D layout, we show the distortion, astigmatism and lateral color w.r.t. the relative field angle, and the standard ray aberration plots. Aberration units are in mm except for the relative distortion. Best viewed by zooming in the electronic version.

6. Conclusions

In this work, we showed how a DNN model can learn to extrapolate from a lens design database and to infer optimized lens designs tailored to the desired specifications for two-lens telescopic objectives. We designed a supervised training approach that feeds on reference designs from the literature to provide a supervision signal for the DNN. We designed an unsupervised training method that generalizes this knowledge to other input specifications by optimizing the RMS spot size while avoiding ray failures and overlapping surfaces. With our experiments on cemented and air-spaced doublets, we showed that our hybrid training method can infer lens designs with an optical performance on par with the reference designs over a wide variety of input specifications.

Our approach can be used to learn other design structures that use spherical surfaces, and other designs with conic or aspherical surfaces can be accounted for by extending our ray-tracing module. In the future, our framework can be extended to provide exciting possibilities for the user such as interacting with the framework by enforcing specific lens variables, or obtaining a variety of starting points for a single set of specifications.

Appendix A: DNN architecture

For the DNN architecture, we use both the SELU activation layer and the weight initialization scheme described in [22]. Empirically, we find that self-normalizing neural networks (SNN) perform better on our problem than the popular ReLU activation coupled with batch normalization. Dropout layers are not included because our unsupervised loss function removes the need for regularization. The DNN architecture is illustrated in Fig. 6. For our application, we found experimentally that stacking the DNN, that is, instantiating the same network multiple times and averaging the results, leads to a better performance than simply increasing the dimensionality.

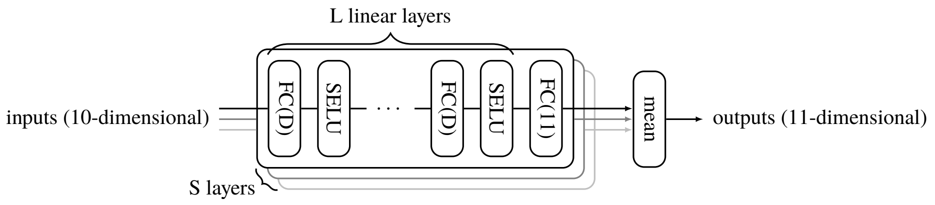

Fig. 6. Model architecture. The model is a stacked fully-connected (FC) DNN. The number of output units of each FC layer is given within parentheses. D is the dimensionality or number of neurons per layer, L is the number of hidden layers and S is the size of the stack. The input and output dimensions are given for lens designs with a glass-air-glass-air structure. Through a grid search, the best parameters found for the telescopic objectives studied in this work were S = 16, L = 7 and D = 32.

Appendix B: Conversion to valid lens variables

The DNN outputs can be any real number and must be further processed to produce valid, meaningful lens variables, namely glass and air thicknesses bounded within the desired range and glass parameters that closely match the behavior of real glasses.

Glass and dispersion model. We restrict our model to light in the visible range and we trace rays with identical weights at three wavelengths, namely the standard C, d and F lines at 656.3, 587.6 and 486.1 nm, respectively. Hence the refractive indices of every glass must be defined for these three wavelengths from the refractive index and Abbe number. As in Smith [21], we consider that the partial dispersion P varies linearly with the Abbe number v and use the glasses K7 (n_d = 1.51112, v = 60.41, P = 0.695) and F2 (n_d = 1.62004, v = 36.37, P = 0.706) as the anchors of our linear model.

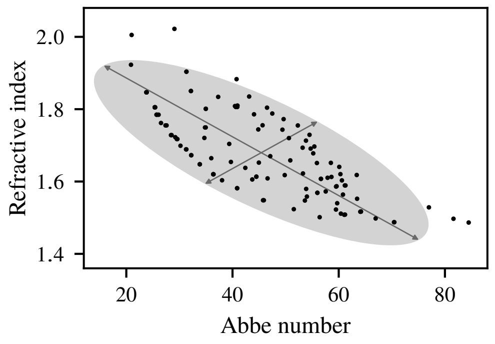

Fig. 7. Computation of valid glass variables. Data points were taken from the Schott glass catalog. The data is modeled as a multivariate normal distribution whose principal axes are shown by the arrows. We bound the glass variables in the gray area which encompasses 2.1 times the standard deviation.

Next, we must address the intimate relationship between n_d and v in real glass materials. Using data from Schott's glass catalog [23] (see Fig. 7), we model the probability distribution of n_d and v as a 2-dimensional normal distribution and find the two principal axes. We bound the glass variables inferred from the model within 2.1 times the standard deviation of the probability distribution. For every glass i, the two DNN outputs $g_{i,1}$ and $g_{i,2}$ are converted into n_d and v by positioning the glass material within the 2D space. Those DNN outputs are first bounded within in the]-1, 1[range using a tanh function, then linearly transformed along the principal axes. The glass model is illustrated in Fig. 7.

Thicknesses. The minimum thickness and thickness range allowed for every air gap or glass are given as inputs to the model. To guarantee that the predicted thicknesses are in the proper interval, the DNN raw outputs are converted with a sigmoid function $f : \mathbb{R} \longrightarrow (t_\mathrm{min}, t_\mathrm{min} + t_\mathrm{range})$. The function must have a non-zero derivative everywhere to make sure that the training signal does not vanish. We use the following sigmoid function with β set to 8:

$$f(t_\mathrm{raw}) = t_\mathrm{min} + \mathrm{softplus}(t_\mathrm{raw} - t_\mathrm{min}, \beta) - \mathrm{softplus}(t_\mathrm{raw} - (t_\mathrm{range} + t_\mathrm{min}), \beta)\ , \tag{6}$$

where $\mathrm{softplus}(\mathrm{x}, \beta) = \log(1 + \exp(\beta x))/\beta$. Regardless of the t_min and t_range parameters, instances from this specific form of sigmoid function all share a common linear region, which empirically performed better compared to other forms of sigmoid functions.

Appendix C: Meridional exact ray tracing

Exact ray tracing can be achieved by using basic geometry when tracing tangential rays through spherical surfaces, which involves successively applying Snell's law. The geometry and variables are illustrated in Fig. 8. Given Q and U at the first surface:

$$\sin I = Qc + \sin U\ , \tag{7a}$$

$$\sin I' = \frac{n \sin I}{n'}\ , \tag{7b}$$

$$U' = U - I + I'\ , \tag{7c}$$

$$Q' = \frac{1}{c}(\sin I' - \sin U')\ . \tag{7d}$$

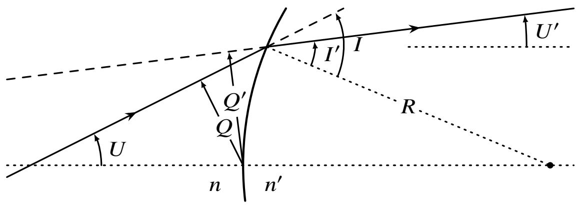

Fig. 8. Illustration of some variables involved in meridional ray tracing through spherical surfaces. Primes decorate variables relative to the refracted ray. I and U are the angle of the ray relative to the surface normal and to the optical axis, respectively. Q is the shortest distance from the surface vertex to the ray and c is the surface curvature.

Appendix D: Training settings

For both the unsupervised and supervised forward passes, we use a batch size of 1024, which means we process in parallel 1024 samples at a time. In the unsupervised pass, we draw 1024 sets of specifications from the unsupervised distribution, then we compute $\mathcal{L}_\text{u}$ from Eq. (3) for every sample. In the supervised pass, we draw 1024 sets of specifications evenly distributed between all the reference designs' probability distribution and compute $\mathcal{L}_\text{s}$ from Eq. (4), again for every sample. By automatic differentiation, the combined hybrid loss $\mathcal{L}_\text{h}$ from Eq. (5) is backpropagated into the DNN so that the derivative of every DNN parameter with respect to the loss is computed. Finally, the weights are updated using the Adam optimization algorithm [24] which is a variant of the SGD algorithm. The Adam parameters are set to the default values 0.9 and 0.999. We use gradient clipping with the threshold set to 8.0. We train over 200 000 training steps. The learning rate is linearly increased from 2×10^{-5} to 2×10^{-3} over 8000 training steps (warm-up phase), then is decayed back to 2×10^{-5} over the remaining duration of the training using a cosinus half-cycle. We found that the specific combination of hyperparameters doesn't matter much as the training schemes are quite robust to the choice of hyperparameters. Training over 200 000 training steps takes about 7 hours on an Nvidia GeForce GTX 1070 GPU. We hypothesize that the training time on more complex optical systems will be similar because of the supervision mechanism which enables the DNN to converge rapidly to a good local minimum. Intuitively, the DNN only explores systems similar to the reference designs, as opposed to the whole parameter space. About 60 000 lens designs can be inferred per second.

Funding

Natural Sciences and Engineering Research Council of Canada; Fonds de Recherche du Québec - Nature et Technologies.

References

1. D. Sturlesi and D. C. O'Shea, "Future of global optimization in optical design," in *1990 International Lens Design Conference*, vol. 1354 (International Society for Optics and Photonics, 1991), pp. 54–69
2. M. Isshiki, H. Ono, K. Hiraga, J. Ishikawa, and S. Nakadate, "Lens Design: Global Optimization with Escape Function," Opt. Rev. **2**(6), 463–470 (1995).
3. S. Thibault, C. Gagné, J. Beaulieu, and M. Parizeau, "Evolutionary algorithms applied to lens design: case study and analysis," in *Optical Design and Engineering II*, vol. 5962 (International Society for Optics and Photonics, 2005).
4. K. Höschel and V. Lakshminarayanan, "Genetic algorithms for lens design: a review," J. Opt. **2**, 463–470 (2018).
5. G. W. Forbes and A. E. W. Jones, "Towards global optimization with adaptive simulated annealing," in *1990 International Lens Design Conference*, vol. 1354 (International Society for Optics and Photonics, 1991), pp. 144–154.
6. C. Menke, "Application of particle swarm optimization to the automatic design of optical systems," in *Optical Design and Engineering VII*, vol. 10690 (International Society for Optics and Photonics, 2018).
7. M. van Turnhout and F. Bociort, "Instabilities and fractal basins of attraction in optical system optimization," Opt. Express **17**(1), 314–328 (2009).

8. D. Petković, N. T. Pavlović, S. Shamshirband, M. L. Mat Kiah, N. Badrul Anuar, and M. Y. Idna Idris, “Adaptive neuro-fuzzy estimation of optimal lens system parameters,” Opt. Lasers Eng. **55**, 84–93 (2014).
9. S. Shamshirband, D. Petković, N. T. Pavlović, S. Ch, T. A. Altameem, and A. Gani, “Support vector machine firefly algorithm based optimization of lens system,” Appl. Opt. **54**(1), 37–45 (2015).
10. S. W. Weller, “Neural network optimization, components, and design selection,” in *1990 International Lens Design Conference*, vol. 1354 (International Society for Optics and Photonics, 1991), pp. 371–379.
11. Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” in *The Handbook of Brain Theory and Neural Networks* (MIT Press, 1995), pp. 255–258.
12. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25*, (Curran Associates, Inc., 2012), pp. 1097–1105.
13. S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Comput. **9**(8), 1735–1780 (1997).
14. A. Graves, A. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE International Conference on Acoustics Speech and Signal Processing* (2013), pp. 6645–6649.
15. I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” in *Advances in Neural Information Processing Systems 27*, (Curran Associates, Inc., 2014), pp. 3104–3112.
16. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature **521**(7553), 436–444 (2015)..
17. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT University, 2016).
18. Zemax Development Corp., “Zebase 6 Optical Design Collection,” (2007).
19. G. Côté, J.-F. Lalonde, and S. Thibault, “Toward Training a Deep Neural Network to Optimize Lens Designs,” in *Frontiers in Optics / Laser Science*, (Optical Society of America, 2018), p. JW4A.28.
20. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, (2017).
21. W. J. Smith, *Modern Lens Design* (McGraw Hill Professional, 2004).
22. G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-Normalizing Neural Networks,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds. (Curran Associates, Inc., 2017) pp. 971–980.
23. Schott Corporation, “Optical Glass Catalog,” (2017).
24. D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *3rd International Conference on Learning Representations (ICLR 2015)*, (2015).