2021 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, OCT. 25–28, 2021, GOLD COAST, AUSTRALIA

A WEIGHTED LOSS FUNCTION TO PREDICT CONTROL PARAMETERS FOR SUPERCONTINUUM GENERATION VIA NEURAL NETWORKS

Diego Stucchi[†] *Andrea Corsini*[†⋆] *Goëry Genty*[⋆] *Giacomo Boracchi*[†] *Alessandro Foi*[⋆]

[†] Politecnico di Milano, Italy, firstname.lastname@polimi.it
[⋆] Tampere University, Finland, firstname.lastname@tuni.fi

ABSTRACT

Supercontinuum light is generated by a train of laser pulses propagating in an optical fiber. The parameters characterizing these pulses influence the spectrum of the light as it exits the fiber. While spectrum generation is a direct process governed by nonlinear equations that can be reproduced through numerical simulation, determining the parameters of the pulse generating a given spectrum is a difficult inverse problem. Solving this inverse problem has a relevant practical implication, as it allows generating beams with desired spectral properties. We solve this multidimensional parameter estimation problem by training a neural network and we introduce, as key technical contribution, a weighted loss function that improves the estimation accuracy. Most remarkably, this loss function is not specific to the considered supercontinuum scenario, but has the potential to improve solutions of similar inverse problems where the forward process can be reproduced via computationally demanding simulations. Our experiments demonstrate the effectiveness of the pursued approach and of our weighted loss function.

1. INTRODUCTION

Physical processes governed by nonlinear equations can often be accurately reproduced via numerical simulations. In most cases, the corresponding inverse problems admit no analytical solution that can be used for an efficient inversion. However, simulating the direct process enables generating vast training sets, which are in principle limited only by computational resources, and that can be used to train machine learning models to efficiently solve the corresponding inverse problems. This approach has been successfully pursued in diverse domains like medical imaging [1, 2], electromagnetic imaging [3] and geosteering [4], where deep learning models have proved to effectively solve signal processing problems provided abundant training data and no prior to invert the process.

In this paper, we adopt neural networks to solve an inverse problem in photonics, namely predicting the pulse parameters generating a target supercontinuum spectrum. Supercontinuum light results from the propagation of a high-power pulse in a nonlinear optical fiber [5], and achieves broader spectra than the traditional monochromatic laser sources. Over the past few years, supercontinuum generation has been an extensive subject of research, due to the wide range of applications in high precision metrology, high resolution imaging or remote sensing. Supercontinuum spectra are routinely generated in the laboratory and validated by simulations, as the process can be accurately modeled via the nonlinear Schrödinger equation or its generalized version [6], thus implemented in numerical software. Inverting the generation process is key to optimize the spectrum to match a target or to feature desired properties. Unfortunately, this is not straightforward as the nonlinearity characterizing the generation process prevents us from formulating an analytical solution. Moreover, the computational cost of simulating the supercontinuum generation limits the feasibility of traditional iterative inversion algorithms.

We frame supercontinuum generation as an inverse problem in a formal setting (Section 2), which we solve by means of neural networks. In particular, we present (Section 3) two different neural network architectures to solve this inverse problem: a fully connected (FC) model and a Convolutional Neural Network (CNN) designed to process one-dimensional signals. The major contribution of this paper is a weighted loss function specifically designed to guide the training process towards the minimization of a spectral error. Most remarkably, the proposed solution is based on a general principle, thus can potentially boost other machine learning models solving multivariate regression problems where the regression quality is measured in the signal domain and that requires generating signals via a computationally expensive simulation. In our experiments (Section 4), we train and test our models over a dataset produced by a generator implementing numerical simulations of the propagation process and we show that our neural networks yield very accurate parameters estimates on these spectra. Moreover, we show that the proposed weighted loss function is beneficial, as the networks trained to minimize it achieve a lower spectral error with respect to networks trained with a traditional parameter loss. Moreover, the proposed solution generalizes rather well to spectra that have not been generated via numerical simulations, but are mixtures of Gaussians locating peaks at different wavelengths.

Similar inverse problems on signals have been solved by training neural networks, but these adopt a standard loss in the parameter domain [1, 2], or prior information [3]. Particularly interesting to our work is [4], where the loss function measures the distance in the signal-domain by invoking multiple times the generator during training. Solutions to the specific inverse problem of supercontinuum generation are based on genetic algorithms [7, 8, 9, 10, 11], and perform an iterative search of the solution by repeatedly generating candidate spectra and then moving the prediction towards the parameters minimizing a spectral error metric. Our neural networks are trained using a pre-generated set of spectra, but without invoking further spectra generation during training, which would be prohibitively expensive. Our experiments show that the proposed weighted loss function effectively acts as a surrogate of the error over the spectra, leading to parameter estimates and spectra that better match the target characteristics.

♡ This work was supported by the Academy of Finland (project no. 310441).

978-1-7281-6338-3/21/$31.00

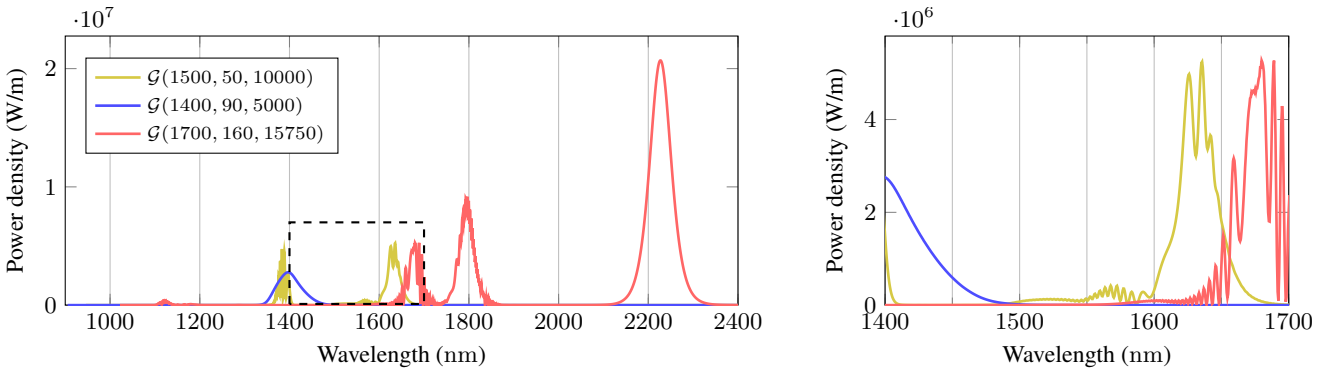


Fig. 1: Left: three spectra generated via the simulation $\mathcal{G}$ as in (1) for the values of $\boldsymbol{y}$ in the legend. These parameters modifies significantly the number of peaks, their magnitude and the wavelength range of these spectra. Right: a close-up view showing oscillations in the spectra.

2. PROBLEM FORMULATION

We denote the supercontinuum generation process as

$$\mathcal{G} : \mathcal{Y} \to \Sigma_{\mathcal{Y}} \subset \Sigma \tag{1}$$

where $\mathcal{Y} \subset \mathbb{R}^p$ is the set of parameters characterizing the pulse propagation, $\Sigma_{\mathcal{Y}}$ is the set of supecontinuum spectra that can be generated via $\mathcal{G}$, contained in the set $\Sigma \subset L^1(\mathbb{R})$ of arbitrary spectra. Our focus is on the *inverse problem* of predicting the parameters $\boldsymbol{y} \in \mathcal{Y}$ yielding a target spectrum $\boldsymbol{\sigma} \in \Sigma$, which has no analytical solution.

We consider $p = 3$ parameters of critical importance for the supercontinuum generation. Specifically $y^{(1)}$ is the pulse wavelength (λ_0), $y^{(2)}$ is its Full Width at Half Maximum (FWHM) and $y^{(3)}$ is its peak power (Pp), while any other generation parameter is fixed. We adopt a discrete representation of a spectrum $\boldsymbol{\sigma}$ as a vector of $N = 4096$ power density values $\boldsymbol{s}$ associated to an equal number of wavelengths $\boldsymbol{\lambda}$, i.e. each spectrum is described by a pair $\boldsymbol{\sigma} = (\boldsymbol{\lambda}, \boldsymbol{s}) \in \mathbb{R}^N \times \mathbb{R}^N$. Our task is to train a ML model $\mathcal{M} : \Sigma \to \mathcal{Y}$ such that

$$\mathcal{M}(\boldsymbol{\sigma}) = \underset{\boldsymbol{y} \in \mathcal{Y}}{\operatorname{argmin}} \left\| \mathcal{G}(\boldsymbol{y}) - \boldsymbol{\sigma} \right\|, \tag{2}$$

for some suitably defined norm $\|\cdot\|$ that captures the *main characteristics* of the spectra, while ignoring fine noise-like spurious features such as those visible in Figure 1. To this purpose, we assume that a training set Tr of K spectrum-parameters pairs is provided,

$$Tr = \{(\boldsymbol{\sigma}_k, \boldsymbol{y}_k) \in \Sigma_{\mathcal{Y}} \times \mathcal{Y} \mid \boldsymbol{\sigma}_k = \mathcal{G}(\boldsymbol{y}_k)\}_{k=1}^K. \tag{3}$$

3. PROPOSED SOLUTION

Our solution to the inverse problem (2), detailed in what follows, consists of training a model $\mathcal{M}$ over a training set Tr to minimize a weighted loss function $\mathcal{L}_W$ that approximates a spectral loss $\mathcal{L}_S$ (Section 3.2). Before being input to the model, spectra are preprocessed via binning and normalization operators (Section 3.1). We train two different architectures for $\mathcal{M}$ (Section 3.3), whose performance is tested in terms of a spectral loss $\mathcal{L}_S$. Figure 2 illustrates the operators and loss functions employed to train and test our models.

3.1. Preprocessing

3.1.1. Binning

To control the dimensionality of the data while also mitigating the spurious features of the power density profile shown in Figure 1, we resort to integral binning. More precisely, we integrate the spectra over n disjoint and adjacent intervals I_j of the same length, $I = \bigcup_j I_j$. We define the *binning* operator $\mathcal{B}_n^I : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^n$ as

$$\boldsymbol{a} = \mathcal{B}_n^I(\boldsymbol{\lambda}, \boldsymbol{s}) = \left[\int_{I_j} \boldsymbol{\sigma}(\omega) d\omega \right]_{j=1}^n \tag{4}$$

where I is the interval of integration and n is the number of subintervals considered. The choice of I and n depends on the adopted model, and is discussed in Section 3.3.

Controlling the dimensionality of the data is particularly important for the FC models, where the number of parameters and operations grow significantly with the input dimension. Moreover, by integrating the spectra over some fixed wavelength range R, the binning process defined above can be used to map spectra to a common reference interval, enabling a direct comparison between spectra potentially defined over different wavelength domains.

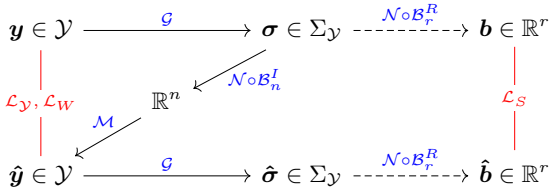


Fig. 2: Illustration of the operators and loss functions used in our solution. A model $\mathcal{M}$ takes as input spectra processed via binning (4) and normalization (5), and is trained minimizing a weighted loss $\mathcal{L}_W$ (8), which serves as a surrogate for a spectral loss $\mathcal{L}_S$ (7). The performance of our solution is measured in terms of $\mathcal{L}_S$, which requires invoking $\mathcal{G}$ to generate spectra. For generated spectra, one may also consider the parameter loss $\mathcal{L}_{\mathcal{Y}}$ (6), which is however not directly related to $\mathcal{L}_S$.

3.1.2. Normalization

We define a *normalization* operator $\mathcal{N} : \mathbb{R}^n \to [\Gamma, 0]$ to map the binned spectrum $\boldsymbol{a} = \mathcal{B}_n^I(\boldsymbol{\sigma})$ to a fixed range in dB scale:

$$\boldsymbol{b} = \mathcal{N}(\boldsymbol{a}) = \max\{\Gamma, 10 \log_{10}(\boldsymbol{a}) - 10 \log_{10}(\max(\boldsymbol{a}))\}, \tag{5}$$

where the clipping below Γ is applied as to ignore any spectral difference at insignificant power levels. We set $\Gamma = -40$.

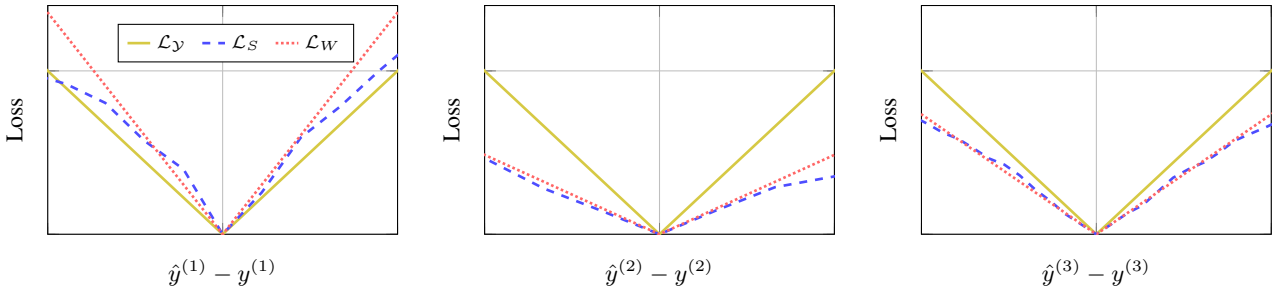


Fig. 3: Comparison among the losses $\mathcal{L}_\mathcal{Y}$ (6), $\mathcal{L}_S$ (7) and $\mathcal{L}_W$ (8) as functions of the error $\hat{y}^{(i)} - y^{(i)}$, $i = 1, 2, 3$, on the three parameters for a given target $\boldsymbol{y}$, assuming that the other two components are correctly predicted, i.e. $\hat{y}^{(j)} = y^{(j)}$ for $j \neq i$. First, we notice that the ideal $\mathcal{L}_S$ is impacted differently by errors over different parameters, while $\mathcal{L}_\mathcal{Y}$ is independent of the error direction. Moreover, the proposed $\mathcal{L}_W$ well approximates the evolution of $\mathcal{L}_S$ along the different error components. These plots illustrate a general behavior of these loss functions for any $\boldsymbol{y} \in \mathcal{Y}$.

3.2. Loss Functions

The inverse problem (2) is a multivariate regression problem, where we want to predict a parameter vector $\boldsymbol{y} \in \mathbb{R}^p$ from the 1-dimensional signal $\boldsymbol{\sigma}$. A straightforward choice for a regression loss function is the Mean Absolute Error (MAE)

$$\mathcal{L}_\mathcal{Y}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{p} \sum_{i=1}^{p} \left| \hat{y}^{(i)} - y^{(i)} \right|, \tag{6}$$

where $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are the true parameters and the regression estimates, respectively. However, when training our models minimizing $\mathcal{L}_\mathcal{Y}$, we ignore the corresponding spectral information, namely, how much error on the parameters estimate impacts the shape of the corresponding spectra. As we stated in Section 2, we want our models to predict parameters generating spectra matching the main characteristics of the target instead of focusing on the fine details of the power density profile. To this purpose, we adopt the binning and normalization operators defined in the previous sections to define a loss function capturing the spectra. We define the *ideal* loss function $\mathcal{L}_S$ as the MAE over the binned and normalized spectra:

$$\mathcal{L}_S(\boldsymbol{b}, \hat{\boldsymbol{b}}) = \frac{1}{r} \sum_{j=1}^{r} \left| b^{(j)} - \hat{b}^{(j)} \right|, \tag{7}$$

where $\boldsymbol{b} = \mathcal{N}(\mathcal{B}_r^R(\boldsymbol{\lambda}, \boldsymbol{s}))$ and $\hat{\boldsymbol{b}} = \mathcal{N}(\mathcal{B}_r^R(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{s}}))$ are the projections of target and estimated spectra to a common domain and $r = 60$. The wavelength range R is chosen to encompass nearly all of the energy of the spectra belonging to the dataset. The choices of R and r are further discussed in Section 4.1.

We observe that the target of the minimization of $\mathcal{L}_S$ and $\mathcal{L}_\mathcal{Y}$ is the same. In fact, for a given sample $(\boldsymbol{\sigma}, \boldsymbol{y}) \in \Sigma_\mathcal{Y} \times \mathcal{Y}$, the loss function $\mathcal{L}_S(\boldsymbol{b}, \cdot)$ achieves its minimum when $\hat{\boldsymbol{b}} = \boldsymbol{b}$, since $\mathcal{L}_S(\boldsymbol{b}, \boldsymbol{b}) = 0$, and it is attained also when $\hat{\boldsymbol{y}} = \boldsymbol{y}$, as this condition in turn realizes the minimum of $\mathcal{L}_\mathcal{Y}(\boldsymbol{y}, \cdot)$. Nevertheless, using a different loss function has an impact on the minimization path followed during the training procedure. In fact, as shown in Figure 3, the loss function $\mathcal{L}_\mathcal{Y}$ is isotropic with respect to the parameters, i.e. the impact of each parameter on the loss function is the same.Instead, $\mathcal{L}_S$ is anisotropic, as each parameter can have a different impact on the generated spectra. Therefore, we expect $\mathcal{L}_S$ to be more effective at estimating parameters corresponding to spectra that match the relevant characteristics of the target.

3.2.1. Proposed Weighted Loss as Nonparametric Local Polynomial Approximation

If we were to train a neural network minimizing $\mathcal{L}_S$, during each epoch of the training process we would have to invoke the generator $\mathcal{G}$ for each predicted parameter vector. Unfortunately, this is a very time-consuming operation, preventing us to adopt the ideal loss $\mathcal{L}_S$ as a training loss. As a surrogate for the ideal loss, we propose a weighted loss function $\mathcal{L}_W$ consisting of a nonparametric multivariate polynomial approximation of $\mathcal{L}_S$. Specifically, for each training sample $(\boldsymbol{\sigma}_k, \boldsymbol{y}_k) \in Tr$, $k = 1, \ldots, K$, we consider the binned and normalized $\boldsymbol{b}_k = \mathcal{N}(\mathcal{B}_r^R(\boldsymbol{\sigma}_k)) \in \mathbb{R}^r$ and we define the *weighted* loss function as

$$\mathcal{L}_W(\boldsymbol{y}_k, \hat{\boldsymbol{y}}_k) = \left[\frac{1}{2} \sum_{i=1}^{p} w_k^{(i)} \left(\hat{y}_k^{(i)} - y_k^{(i)} \right)^2 \right]^{1/2}, \tag{8}$$

where $\hat{\boldsymbol{y}}_k$ is the model prediction and $\boldsymbol{w}_k \in \mathbb{R}^p$ are local polynomial coefficients defined by solving the following optimization problem:

$$\boldsymbol{w}_k = \underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \sum_{l=1}^{K} e^{-\alpha \|\boldsymbol{y}_l - \boldsymbol{y}_k\|_2^2} \cdot \left[\mathcal{L}_S(\boldsymbol{b}_k, \boldsymbol{b}_l)^2 - \textstyle\sum_{i=1}^{p} w^{(i)} \left(y_l^{(i)} - y_k^{(i)} \right)^2 \right]^2 \right\}, \tag{9}$$

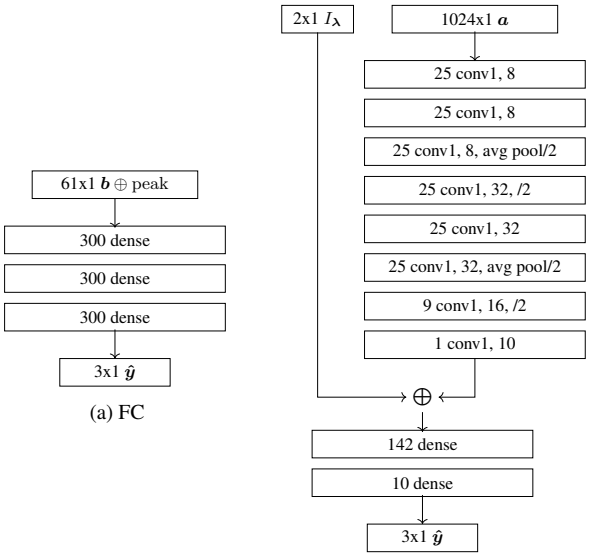
where $\alpha > 0$ controls the spread of a Gaussian window that localizes the fit to the ideal loss with respect to the deviation on the parameters. Figure 3 shows that $\mathcal{L}_W$ locally approximates $\mathcal{L}_S$ better than $\mathcal{L}_\mathcal{Y}$ in the neighborhood of the correct estimation.

3.3. Adopted models

This section presents the proposed models, detailing their architectures and the performed data pre-processing steps.

3.3.1. Fully Connected Model

The first model we present is the fully connected neural network. From each spectrum $(\boldsymbol{\lambda}, \boldsymbol{s})$, we extract a feature vector by computing its binned and normalized version over the unique wavelength range R defined above with $n = 60$ intervals, namely $\boldsymbol{b} = \mathcal{N}(\mathcal{B}_{60}^R(\boldsymbol{\lambda}, \boldsymbol{s})) \in \mathbb{R}^{60}$. Moreover, in order to provide relevant information to the network, we append the peak value of the binned spectra (before normalization) to the vector, yielding



(a) FC

(b) CNN

Fig. 4: Architecture of the (a) FC and (b) CNN models. Each layer uses a ReLU activation function as nonlinearity. The output layers are both characterized by linear activations.

a 61-dimensional feature vector. The adopted FC contains about 200000 trainable parameters and its architecture is reported in Figure 4(a).

3.3.2. Convolutional neural network

The second model consists of a 1D convolutional neural network (CNN) processing spectra as 1D signals. Since CNNs can process high-dimensional data with fewer trainable parameters than traditional fully connected neural networks, we employ a less coarse preprocessing step, where binning is performed with $n = 1024$ bins, and we do not apply any normalization. In particular, we preprocess each spectrum $\boldsymbol{\sigma}$ in its wavelength range $I_{\boldsymbol{\lambda}} = [\min(\boldsymbol{\lambda}), \max(\boldsymbol{\lambda})]$ and the binned spectrum is computed as $\boldsymbol{a} = \mathcal{B}^{I_{\boldsymbol{\lambda}}}_{1024}(\boldsymbol{\lambda}, \boldsymbol{s})$. Finally, since we use a different I for each spectrum, we also append the wavelength range $I_{\boldsymbol{\lambda}}$ to the latent representation returned after the convolutional block. The CNN contains about 96000 trainable parameters and its architecture is reported in Figure 4(b).

4. EXPERIMENTS

In this section, we describe our dataset (Section 4.1) and introduce the metrics adopted to evaluate our neural networks (Section 4.2). Then, in our experiments we investigate the impact of the training set size and of the proposed loss function when training our models. Finally, we show network performances when predicting spectra that have not been generated from $\mathcal{G}$.

4.1. Dataset

The employed dataset is generated using $\mathcal{G}$ with all the combination of the considered parameters varying in the ranges reported in Table 1, yielding $\mathbb{D} = \{(\mathcal{G}(\boldsymbol{y}); \boldsymbol{y}) = (\boldsymbol{\lambda}, \boldsymbol{s}; \boldsymbol{y})\}$. Following an analysis of the dataset, we define the wavelength interval R used in (7) as $[1043.7, 2675.4]$ nm, encompassing the wavelength region where most of the power density of the spectra in the dataset is contained. Moreover, we set $r = 60$, yielding a bin width of $|R|/r \approx 27.2$ nm.

	Minimum	Maximum	Step	#
λ_0	1400 nm	1700 nm	10 nm	31
FWHM	50 fs	250 fs	10 fs	21
Pp	500 W	20 000 W	250 W	79

Table 1: Ranges for the pulse parameters belonging to the dataset $\mathbb{D}$.

4.2. Metrics

We define the *spectrum error* as the ideal loss (7) accumulated over the test set Te :

$$E_S = \sum_{(\boldsymbol{\sigma}, \boldsymbol{y}) \in Te} \mathcal{L}_S(\boldsymbol{b}, \hat{\boldsymbol{b}}), \tag{10}$$

where Te is a set of parameters-spectra pairs, defined as (3). For performance assessment only, we generate the spectra associated to the model predictions $\{\hat{\boldsymbol{y}}\}$ over the test set Te. We also assess the parameter error, consisting of the MAE on the predicted parameters:

$$E_{\mathcal{Y}} = \sum_{(\boldsymbol{\sigma}, \boldsymbol{y}) \in Te} \mathcal{L}_{\mathcal{Y}}(\boldsymbol{y}, \hat{\boldsymbol{y}}), \qquad \hat{\boldsymbol{y}} = \mathcal{M}(\boldsymbol{\sigma}). \tag{11}$$

As we commented in Section 3.2, both the spectrum error (10) and the parameters error (11) realize the minimum when the predicted parameter vector (and consequently the generated spectrum) matches the target for every test sample.

In our study, we investigate how the performance of the considered models varies with progressively smaller training sets. To this purpose, we uniformly split the dataset $\mathbb{D}$ into F folds. Then, each neural network and baseline is exclusively trained on fold f_i and tested on fold $f_{(i \bmod F)+1}$, where $i = 1, \ldots, F$. Hence, increasing the number of folds F yields smaller training sets. Eventually, performance measures are averaged across all the F folds. In our experiments, we consider $F = 5, 10, 20$, corresponding to training sets having 10285, 5142 and 2571 samples respectively.

4.3. Employed Methods

We train both FC and the CNN to optimize the loss functions $\mathcal{L}_{\mathcal{Y}}$ and $\mathcal{L}_W$. In both cases we independently train 10 neural networks, and at inference time we consider the model ensemble, averaging predictions of the individual networks. Ensembles are typically used to reduce over-fitting, and consequently improve the scores over the metrics [12].

As a baseline model, we compare against the 1-Nearest Neighbor model (1NN) using $\mathcal{L}_S$ (7) as distance function, associating the preprocessed target spectrum to the parameter corresponding to the closest sample in the training set.

4.4. Results

Table 2 reports the averaged spectrum error E_S (10) and parameter error $E_{\mathcal{Y}}$ (11) achieved by our models in the cross-validation scenario presented above. Each value in the table is the result of averaging the performance of the model ensemble over F folds, with $F \in \{5, 10, 20\}$. As expected, both the parameter error $E_{\mathcal{Y}}$ and the spectrum error E_S grow when the training set size decreases, i.e. F increases. Also, the proposed models perform significantly better than the baseline 1NN, with the CNN outperforming the FC model

		$F=5$			$F=10$			$F=20$		
		$E_{\mathcal{Y}}$	E_S	ΔE_S	$E_{\mathcal{Y}}$	E_S	ΔE_S	$E_{\mathcal{Y}}$	E_S	ΔE_S
1NN		0.123	0.354	-	0.161	0.449	-	0.206	0.569	-
FC	$\mathcal{L}_{\mathcal{Y}}$	0.029	0.168	(-7.1%)	0.045	0.235	(-8.5%)	0.066	0.329	(-5.5%)
	$\mathcal{L}_W$	0.032	0.156		0.047	0.215		0.070	0.311	
CNN	$\mathcal{L}_{\mathcal{Y}}$	0.014	0.130	(-11.5%)	0.022	0.186	(-7.5%)	0.034	0.268	(+1.5%)
	$\mathcal{L}_W$	0.017	0.115		0.028	0.172		0.044	0.272	

Table 2: Performance of the proposed models and baseline in terms of parameter error $E_{\mathcal{Y}}$ (11) and spectrum error E_S (10) for different number of folds $F \in \{5, 10, 20\}$. We report the results achieved training the models both with $\mathcal{L}_{\mathcal{Y}}$ (6) and $\mathcal{L}_W$ (8). In parenthesis, we report the relative spectrum error variation ΔE_S resulting from using $\mathcal{L}_W$ instead of $\mathcal{L}_{\mathcal{Y}}$.

in every setting. We speculate that the heavy binning of spectra employed to define the input of the FC network (see Sec. 3.3) might discard some information useful for parameters estimation. In contrast, binning in the CNN is less severe and further dimensionality reduction is learned via convolutional blocks to maximise regression performance.

The proposed weighted loss $\mathcal{L}_W$ is mostly beneficial, as it decreases the spectrum error with respect to the comparable models trained with $\mathcal{L}_{\mathcal{Y}}$ in almost all settings. Nevertheless, when the training set is small, the estimated polynomial coefficients are less accurate, thus the weighted loss is less effective. This is particularly apparent with the CNNs, which are very effective even when using $\mathcal{L}_{\mathcal{Y}}$. In fact, when $F=20$, the weighted loss seems not beneficial for the CNNs. Finally, we remark that models trained with the weighted loss yield a larger parameter error $E_{\mathcal{Y}}$, suggesting that the training led to a lower point in the spectrum error surface, while not actually getting closer to the true parameters prediction in the Euclidean sense.

The first row in Figure 5(a) reports a pair of spectra generated via $\mathcal{G}$ and the corresponding FC and CNN predictions. We can notice that both our models have successfully predicted the parameters generating spectra, as they mostly overlap the target. On the second row, we show the corresponding binned and normalized versions of the spectra (which are used for computing the loss), for which the same considerations hold.

4.5. Application: non-generated spectra

Experiments in Section 4.4 demonstrate that our neural networks can successfully estimate the parameters $\hat{\boldsymbol{y}}$ of train pulses to provide a spectrum that is very close to a target $\boldsymbol{\sigma}$. However, those experiments involve only generated spectra $\Sigma_{\mathcal{Y}}$ which are therefore consistent with the generative process. Now we investigate the generalization capabilities of our neural networks, namely how good these are when predicting parameters corresponding to spectra that do not belong to $\Sigma_{\mathcal{Y}}$, thus that have not been produced by $\mathcal{G}$. This issue is particularly relevant for the adoption of our neural networks in practice, as generated spectra, like those in Figure 1, exhibit rapid fluctuations and detailed patterns that are very unlikely to emerge in a description of an arbitrary spectral profile provided by a user or dictated by system design or application requirements.

In this experiment, we provide a qualitative assessment of the parameters estimation performance over a dataset $\mathbb{H}$ containing spectra that have not been generated from $\mathcal{G}$. More specifically, $\mathbb{H}$ contains more than 45000 spectra with diverse macroscopic characteristics (number of peaks, wavelength range, peak intensity), that have been defined by fitting a Gaussian Mixture (using at most 5 mixture components) over a few spectra in $\Sigma_{\mathcal{Y}}$. These result in coarse approximations that are akin to a possible spectrum defined by a user. We use $\mathbb{H}$ only as a testing dataset; the ensemble of models $\mathcal{M}$ is trained over subsets of $\mathbb{D}$ as in the previous experiments but attempts now to predict parameters generating spectra as close as possible to the ones in $\mathbb{H}$.

In Figure 5(b), we report some spectra from $\mathbb{H}$ and the corresponding predictions obtained via both the FC model and the CNN ensembles. As expected, the predictions on $\mathbb{H}$ are not as accurate as those produced on $\mathbb{D}$ in Figure 5(a). This can be explained by the fact that spectra in $\mathbb{H}$ are much smoother than those generated by $\mathcal{G}$, which are characterized by fast oscillations. On the other hand, the normalized spectra in the second line indicate that after binning and normalization, the predictions are rather close to the target. This confirms that $\mathcal{L}_W$ (8) is a good surrogate for $\mathcal{L}_S$ (7), which involves binned and normalized spectra.

Both the employed models consist of small architectures and do not require GPU acceleration to be trained in reasonable time. While our main focus is on the weighted loss function rather than on optimizing the employed neural architectures, it is important to emphasize that both models allow inference of pulse parameters in a matter of seconds. This is dramatically faster than any alternative involving spectra generation through the simulator, as these have to generate many test spectra at run time, each requiring several minutes.

5. CONCLUSIONS

We present two neural networks solving the inverse problem of estimating pulse parameters for generating a target supercontinuum spectrum. These networks are particularly effective thanks to a weighted loss function that allows to pragmatically approximate a surrogate of a spectrum-wise error metric; this solution is also efficient, as it does not require generating spectra during training.

Future work goes towards adapting the proposed weighted loss function to similar inverse problems in the signal domain, as well as investigating new weighting schemes. One promising direction is to employ local anisotropic and adaptive polynomial estimators such as [13] to define adaptive localization window for the nonparametric local polynomial fit in (9). Moreover, we are going to extend the proposed solution to estimate more control parameters, and to integrate our trained models into an actual photonic system for controlling the laser pump and the train pulses in real time.

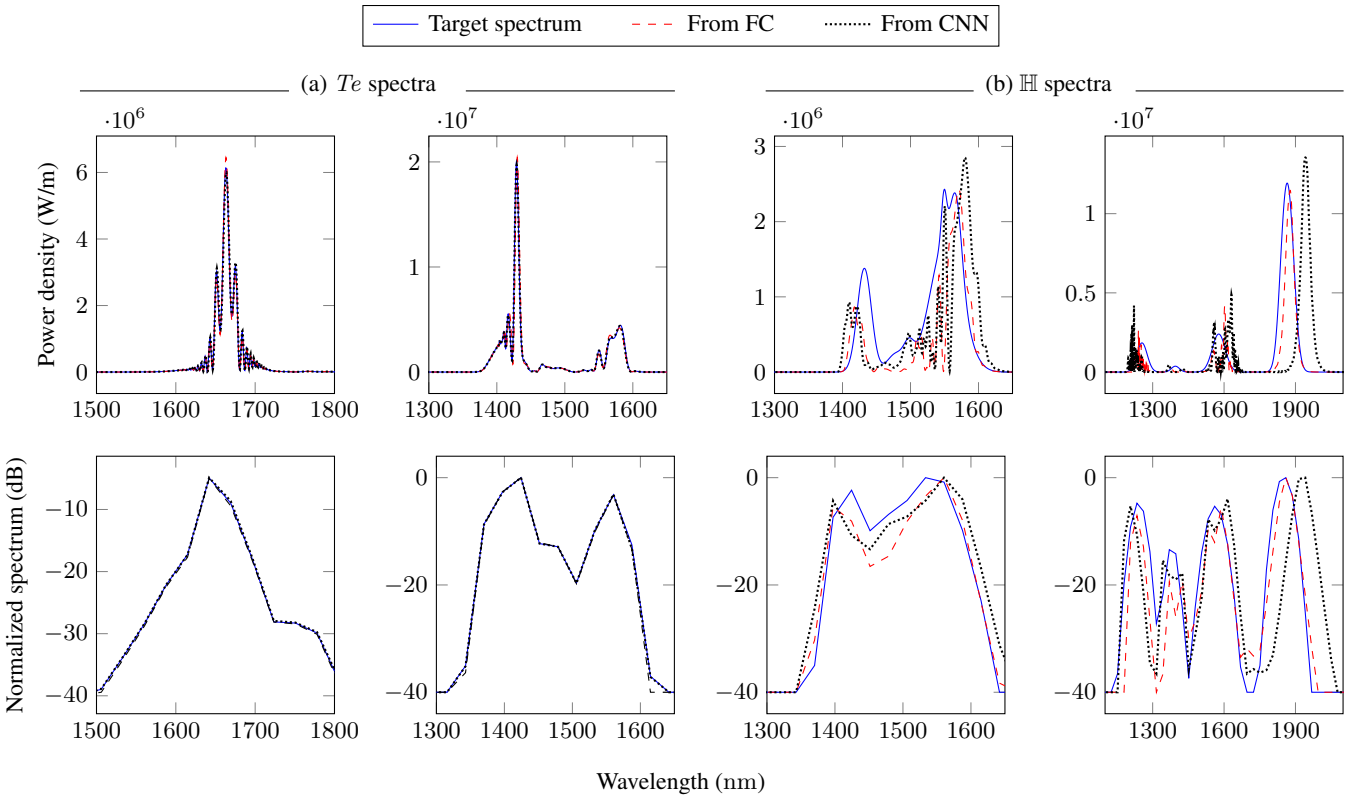


Fig. 5: In the first row, a target spectrum (blue) and the corresponding spectra generated by the pulse parameters predicted via the ensemble of FC model (red, dashed) and of CNN (black, dotted), both trained to minimize $\mathcal{L}_W$ (8) in the $F = 10$ settings. On the left, the target spectra belong to the test set Te. On the right, the target spectra belong to the dataset $\mathbb{H}$ of Gaussian Mixture approximations.

6. REFERENCES

[1] D. Patel, R. Tibrewala, A. Vega, L. Dong, N. Hugenberg, and A. A. Oberai, "Circumventing the solution of inverse problems in mechanics through deep learning: Application to elasticity imaging," *Computer Methods in Applied Mechanics and Engineering*, vol. 353, pp. 448–466, 2019.

[2] S. J. Hamilton and A. Hauptmann, "Deep d-bar: Real-time electrical impedance tomography imaging with deep neural networks," *IEEE Transactions on Medical Imaging*, vol. 37, no. 10, pp. 2367–2377, 2018.

[3] Z. Wei and X. Chen, "Deep-learning schemes for full-wave nonlinear inverse scattering problems," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 4, pp. 1849–1860, 2018.

[4] Y. Jin, Q. Shen, X. Wu, J. Chen, Y. Huang, et al., "A physics-driven deep-learning network for solving nonlinear inverse problems," *Petrophysics*, vol. 61, no. 01, pp. 86–98, 2020.

[5] J. M. Dudley, G. Genty, and S. Coen, "Supercontinuum generation in photonic crystal fiber," *Reviews of Modern Physics*, vol. 78, no. 4, pp. 1135, 2006.

[6] G. Agrawal, "Pulse propagation in fibers," in *Nonlinear Fiber Optics (Fifth Edition)*, G. Agrawal, Ed., Optics and Photonics, pp. 27–56. Academic Press, Boston, 2013.

[7] F. Arteaga-Sierra, C. Milián, I. Torres-Gómez, M. Torres-Cisneros, H. Plascencia-Mora, G. Moltó, and A. Ferrando, "Optimization for maximum Raman frequency conversion in supercontinuum sources using genetic algorithms," *Revista Mexicana de Física*, vol. 63, no. 2, pp. 111–116, 2017.

[8] E. Kerrinckx, L. Bigot, M. Douay, and Y. Quiquempois, "Photonic crystal fiber design by means of a genetic algorithm," *Optics Express*, vol. 12, no. 9, pp. 1990–1995, 2004.

[9] R. Musin and A. Zheltikov, "Designing dispersion-compensating photonic-crystal fibers using a genetic algorithm," *Optics Communications*, vol. 281, no. 4, pp. 567–572, 2008.

[10] G. Moltó, M. Arevalillo-Herráez, C. Milián, M. Zacarés, V. Hernández, and A. Ferrando, "Optimization of supercontinuum spectrum using genetic algorithms on service-oriented grids," in *Proceedings of the 3rd Iberian Grid Infrastructure Conference*, 2009, pp. 137–147.

[11] W. Q. Zhang, J. E. Sharping, R. T. White, T. M. Monro, and S. Afshar, "Design and optimization of fiber optical parametric oscillators for femtosecond pulse generation," *Optics Express*, vol. 18, no. 16, pp. 17294–17305, 2010.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*, chapter 14, Springer, 2006.

[13] A. Goldenshluger and A. Nemirovski, "On spatial adaptive estimation of nonparametric regression," *Mathematical Methods of Statistics*, vol. 6, no. 2, pp. 135–170, 1997.