Supplementary Materials for

Spectrally Programmable Optical Frequency Comb Generation

Hudi Liu[1], Yuhan Du[1], Xingfeng Li[1], Xingchen Ji[2], and Yikai Su[1*]

[1]State Key Laboratory of Advanced Optical Communication Systems and Networks, Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

[2]John Hopcroft Center for Computer Science, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

[*]Email: yikaisu@sjtu.edu.cn

Contents

S1. Neural Network Structure and Training

In this study, we used a 1D-CNN to process and analyze a dataset comprising 9500 1D sequences. Each sequence had a length of 5417, which was determined by the experimental setup, such as the resolution and span of the optical spectrum analyzer. The core architecture of the model was a 1D-ResNet specifically designed to adapt to 1D data and process spectral sequences. This architecture involved modifying traditional two-dimensional methods to accommodate 1D inputs. The central component of this architecture was the residual block, which contained two convolutional layers, each followed by batch normalization and a ReLU activation function. This structure not only stabilized the training process but also introduced the necessary nonlinearity into the network. Crucially, these blocks included a shortcut connection that facilitated the direct flow of input across the block, supporting the propagation of gradients during training, and allowing for the construction of deeper networks. If the input and output dimensions differ, the shortcut uses an additional convolutional layer to match them.

As shown in Fig. S1(a), the 1D-ResNet architecture began with an initial convolution layer designed to increase the data dimensionality while extracting preliminary features. Subsequently, the multiple residual structures were organized into four consecutive layers, each containing two residual blocks. Each residual block comprises of two convolutional layers. Following these layers, a global average pooling operation effectively summarized the features into a single vector for each channel, which was then processed by a fully connected layer designed to map the condensed features to a specified number of outputs. Finally, the sigmoid activation function normalized the output of each neuron. Table S1 shows the parametric table of 1D-ResNet used in our experiment.

After each training epoch, the model performance was evaluated using a separate test dataset. The test data were forward-propagated through the model to generate predictions, and the average test loss was calculated. The test loss for each epoch was stored to track the performance of the model on the unseen data. The learning rate was optimized and set to 3×10^{-4} to ensure stable and effective learning. The training and test losses of the 1D-ResNet are shown in Fig. S1(b). The loss of the validation set is lowest at the checkpoint, with a value of 0.0067.

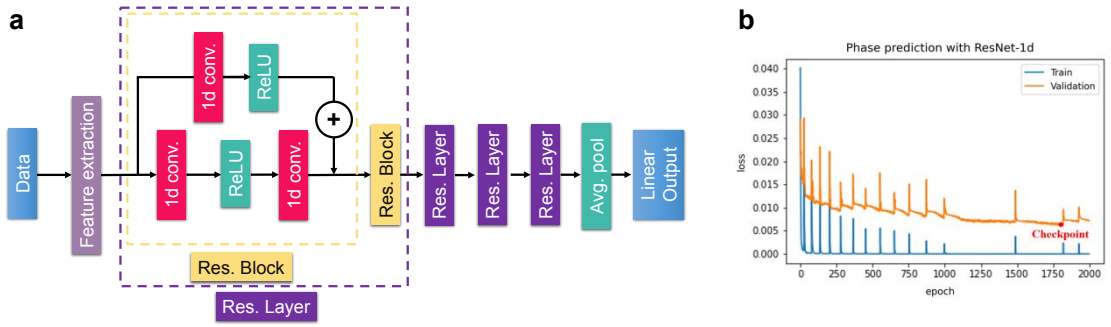

Fig. S1 | a 1D-ResNet model structure with parameters used in training. 1D conv., one-dimensional convolutional layer; ReLU, a kind of nonlinear activation function; Res. Block, residual blocks; Avg. pool, average pool. **b** Training and validation loss vs. training epoch of the 1D-ResNet model.

S2. Generating Neural Inputs from Target Envelopes

To use the trained 1D-ResNet model effectively for inference, it is crucial to generate appropriate neural inputs that align with the model-training data format. The 1D-ResNet was trained using full spectral sequences sampled by an OSA, rather than envelope sequences because the full spectrum captures the intricate spectral features generated by nonlinear processes such as self-phase modulation and four-wave mixing. These processes introduce interline components and subtle spectral variations that are essential for accurate phase adjustment predictions. Additionally, training on only peak powers would significantly reduce input dimensionality, leading to underfitting and insufficient learning of the complex relationships between spectral features. By utilizing the full spectral data, we ensure that all relevant spectral information is preserved. Therefore, to maintain consistency and ensure accurate predictions, the inference process required the generation of spectral sequences from the target envelopes.

To address this requirement, we proposed a method that used a long short-term memory (LSTM) network to establish a mapping relationship between the peaks and troughs of comb lines in the collected data. The LSTM network was implemented using the PyTorch 1.10.0 Python library in Python version 3.8.9. Through careful optimization of the hyperparameters, we determined that the optimal configuration for the LSTM network consisted of a hidden layer size of 128 and a depth of two hidden layers. The choice of the LSTM network was motivated by the observation that the positions of the troughs in the experimental spectral data correlated with adjacent peaks. Additionally, the characteristics of the OSA, such as its resolution and sensitivity, influence the positioning of the troughs. By training the LSTM network to learn these relationships, the corresponding troughs could be accurately inferred

for a given set of peaks in the target envelope. During the training process, we used the Adam optimization algorithm, which is based on the principles of stochastic gradient descent. The Adam optimizer adapts the learning rates for each parameter, enabling efficient convergence and minimizing the effects of noisy gradients. The loss function is defined using the MSE metric, which quantifies the difference between the predicted and actual trough positions. By minimizing the MSE loss, the LSTM network learned to accurately map the peaks to their corresponding troughs. The training and test losses of the 1D-ResNet are shown in Fig. S2.

Once the LSTM network was trained, it was used to infer troughs corresponding to the peaks in the target envelope. We used a Gaussian interpolation method to reconstruct the complete spectral sequence required as input for the 1D-ResNet model. This method interpolated the values between each peak and its corresponding trough, thereby generating a smooth continuous spectral sequence. By combining the LSTM network for peak-to-trough mapping and Gaussian interpolation for sequence reconstruction, we effectively generated the necessary neural inputs from the target envelopes. As shown in Fig. S3, this approach ensured that the 1D-ResNet model received input data that closely resembled the spectral sequences on which it was trained, enabling accurate and reliable predictions.

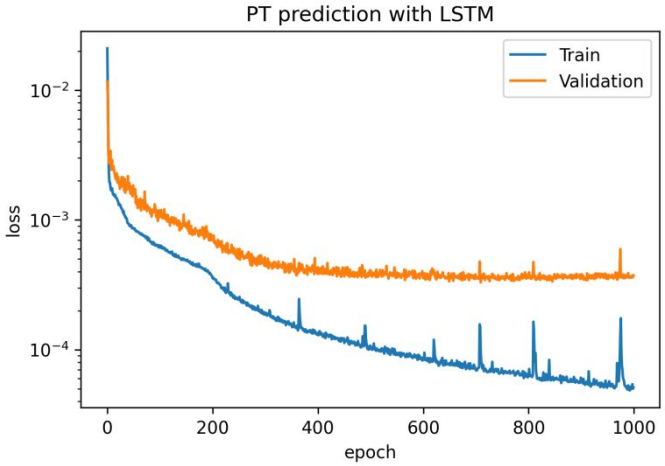

Fig. S2 | Training and validation loss vs. training epoch of the LSTM model.

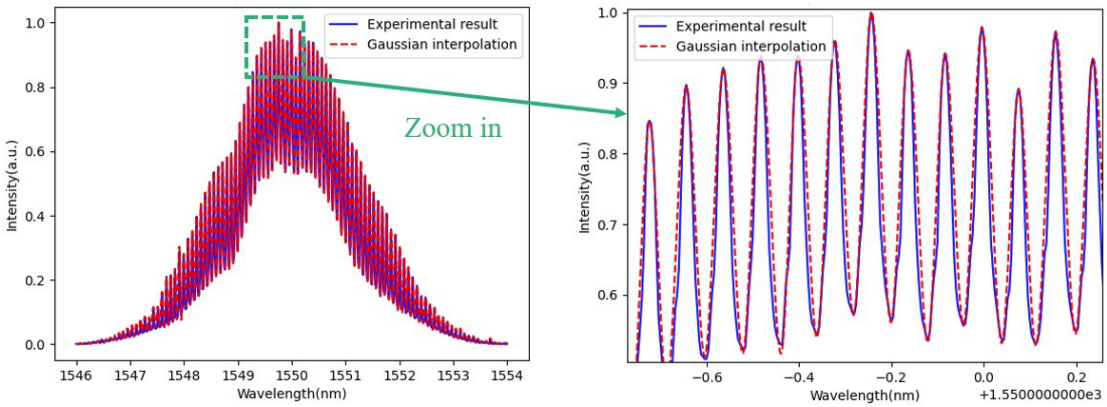

Fig. S3 | Spectrum reconstructed from peaks and troughs using Gaussian interpolation.

S3. The Programmable Range of the SPOFC Generation System

The programmable range of the SPOFC generation system was influenced by several factors, which can be broadly categorized into two main groups: physical limitations imposed by the experimental setup, and constraints arising from the neural network training process. Among the different experimental parameters, the input power of the HNLF plays a crucial role in determining the feasible range for broadened OFCs.

To investigate the effect of the input power on the broadening results, we conducted experiments under two different conditions. In the first scenario, labeled Dataset I, the output power of the EDFA was set to 23 dBm. Under these conditions, setting the parameter σ in Equation (2) to 0.4 failed to generate a broadened OFC whose envelope matched the target envelope. However, when the EDFA output power was increased to 27 dBm, as represented by Dataset II, the system was able to produce a broadening result with a higher similarity to the target envelope, as illustrated in Fig. S4. This observation highlighted the significance of HNLF input power in determining the achievable range of the SPOFC generation system.

The suboptimal performance observed at the lower input power of 23 dBm is primarily due to physical limitations in the experimental setup rather than any deficiency in the neural network's capacity to model complex nonlinear phenomena. At this power level, nonlinear effects in the fiber, such as self-phase modulation and four-wave mixing, are too weak to induce significant spectral broadening. Increasing the σ value of the Gaussian function demands a wider bandwidth for the broadened frequency comb. However, the experimental bandwidth cannot be expanded indefinitely under current conditions due to physical constraints. These limitations arise because nonlinear optical effects require a threshold of input

power to be effectively triggered, and below this threshold, no optimization technique can overcome these physical barriers to achieve the desired broadening. At 27 dBm input power, the nonlinear effects are more pronounced, leading to spectral broadening that closely matches the target spectrum. In this regime, the neural network can accurately predict and facilitate spectral shaping because the necessary nonlinear effects are present.

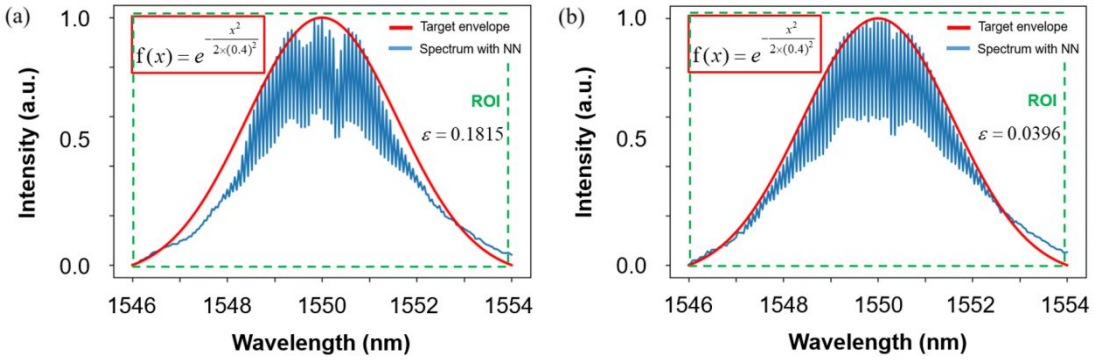

Fig. S4 | SPOFC generation with different input power of the HNLF. Input power is (a). 23 dBm and (b). 27 dBm.

In addition to the physical limitations, the neural network training process plays a vital role in constraining the spectral programmable range of the system. Deep learning algorithms are data-driven, and the content of the training dataset significantly influences the network's ability to fit and generalize unseen data. To mitigate the risk of overfitting and enhance the performance of the model on unknown data, we used several regularization techniques such as weight decay and batch normalization. These techniques helped improve the generalization ability of neural networks, allowing them to handle a wider range of target combs.

To expand the achievable target comb range further, we proposed three approaches. The first method focused on increasing the controllable feature parameters of the seed comb. This can be achieved by incorporating additional information into the training dataset, such as the number of comb teeth with phase perturbations or the amplitude of the seed comb at different frequencies. This approach necessitates the optimization of the network architecture, which requires an increase in the depth and width of the model to enhance its representational power and accommodate additional input features.

The second approach involves increasing the input power of the HNLF, which intensifies nonlinear effects and extends the feasible range of the target combs. However, it is important to note that high input powers may introduce other high-order nonlinear effects that are not considered in the model training

process. These unmodeled effects can adversely affect the prediction results. To address this issue, samples with different input powers can be added to the training dataset such that the model can adapt to different input power conditions. This approach allowed the model to learn and compensate for the effects of different input powers, thereby improving its robustness and performance.

The third approach leverages transfer learning to reduce the retraining time when adapting the model to different experimental setups, such as varying input power or comb characteristics. For instance, we conducted experiments under two conditions—Dataset I with an EDFA output power of 23 dBm and Dataset II with 27 dBm. The model trained on Dataset I was fine-tuned with Dataset II, and transfer learning reduced the retraining time by 75% while maintaining a high prediction accuracy for the broadened OFC. By reusing the learned features from the original model and fine-tuning them with new data, this method significantly shortens the retraining process, improving the model's efficiency and adaptability while maintaining high prediction accuracy across different conditions.

Table S1. Parameter table of 1D-ResNet used in the experiment

Layer	Type	Output Channel	Kernel Size	Stride	Padding	Activation Function
Input Layer	Conv1d	64	3	2	3	ReLU
Residual Layer1						
- Residual Block1	Conv1d	64	3	1	1	ReLU
	Conv1d	64	3	1	1	-
- Residual Block2	Conv1d	64	3	1	1	ReLU
	Conv1d	64	3	1	1	-
Residual Layer 2						
- Residual Block1	Conv1d	128	3	4	1	ReLU
	Conv1d	128	3	1	1	-
- Residual Block2	Conv1d	128	3	1	1	ReLU
	Conv1d	128	3	1	1	-
Residual Layer 3						
- Residual Block1	Conv1d	256	3	8	1	ReLU
	Conv1d	256	3	1	1	-
- Residual Block2	Conv1d	256	3	1	1	ReLU
	Conv1d	256	3	1	1	-
Residual Layer 4						
- Residual Block1	Conv1d	512	3	8	1	ReLU
	Conv1d	512	3	1	1	-
- Residual Block2	Conv1d	512	3	1	1	ReLU
	Conv1d	512	3	1	1	-
Pooling	AvgPool1d	-	10	-	-	-
Output Layer	Linear	5	-	-	-	Sigmoid