Optoelectronic Systems Trained With Backpropagation Through Time

Michiel Hermans, Joni Dambre, and Peter Bienstman

***Abstract*— Delay-coupled optoelectronic systems form promising candidates to act as powerful information processing devices. In this brief, we consider such a system that has been studied before in the context of reservoir computing (RC). Instead of viewing the system as a random dynamical system, we see it as a true machine-learning model, which can be fully optimized. We use a recently introduced extension of backpropagation through time, an optimization algorithm originally designed for recurrent neural networks, and use it to let the network perform a difficult phoneme recognition task. We show that full optimization of all system parameters of delay-coupled optoelectronics systems yields a significant improvement over the previously applied RC approach.**

***Index Terms*— Backpropagation through time (BPTT), delayed dynamic systems, optical computing, physical neural networks.**

I. Introduction

A rapidly growing body of research focuses on utilizing nonlinear photonic systems for information processing, using the so-called reservoir computing (RC) paradigm [1], [2]. RC is built on the concept of using a high-dimensional, dynamical system determined by randomly initialized parameters as a black-box information processing device. The dynamical system (reservoir) is driven with an input time series that needs to be processed. The high-dimensional reservoir state will, as a result, offer a broad set of nonlinear functions of the recent history of the input signal. Using linear regression, it is then possible to map the reservoir state to the desired output. Despite the apparent simplicity of the system, RC has, in certain cases, proven to be a powerful tool in information processing [3].

Due to the fact that RC uses randomly initialized parameters, it forms an ideal basis for photonic computation. There is no need for precisely engineering a particular photonic component for a particular type of data processing; a photonic reservoir simply needs to be able to take in information and internally generate a large set of nonlinear transformations. Research in this direction has lead to a number of photonic systems being considered as viable candidates for RC. The two most important ones are those based on optical networks [4], [5], and those based on delay-coupled systems [6]–[8]. In both cases, the ability to process data has been demonstrated on a number of tasks, such as channel equalization, speech recognition, and so forth.

The RC, however, has one important disadvantage. Due to the fact that the dynamics of the reservoir are determined by randomly chosen parameters, many types of data processing remain difficult or impractical to perform. Consider a simple example: suppose that one wishes to construct a reservoir system to act as a synchronous two-bit counter. In order for this to be possible, there must exist four distinct stable attractors somewhere within the space of the reservoir state, through which the system cycles each time a clock pulse arrives. The only way to obtain such a system in the RC approach is to sample a reservoir with these particular properties by luck. Even for such a simple device as a two-bit counter, the probability of sampling a dynamical system with these exact properties in a small-scale system is extremely small. If the desired behavior is even more complex, a suitable reservoir might never be found by random sampling. The only way to overcome this is to dramatically increase the dimensionality of the state space, yielding systems that are impractically large and certainly not competitive with alternative technologies.

The concept of RC originated (among others) as a new method to train recurrent neural networks (RNN). RNNs are abstract, discrete time dynamical systems used for processing time series. The time-honored method for optimizing (commonly called training in the machine learning community) RNNs is known as backpropagation through time (BPTT) [9], [10]. Essentially, one defines a cost function based on the desired network output, and using the chain rule, one determines the gradient of this cost function with respect to the internal parameters of the system. The term BPTT stems from the fact that, due to the recursion in RNNs, computing the gradient involves propagating an error signal backward through the system updates, i.e., backward in time.

In [11], we showed that BPTT can be extended to operate on systems that operate in continuous time, and which are governed by differential equations instead of a discrete update equation. It was shown that it can provide robust solutions to surprisingly complicated problems. Among others, it was successfully used to optimize the parameters of an integrated optical amplifier network, showing that it has potentially powerful applications in the domain of photonic computation.

In this paper, we consider a photonic delay-coupled system as proposed and studied in [7]. Rather than adhering to the RC paradigm, we consider the system as a parametric model, which can be optimized using gradient descent. We use BPTT to optimize most of the system parameters and test this approach on a challenging speech recognition task. We show that this approach can fully exploit the computational power of the photonic system, and yield significantly better results than the RC approach.

II. Delay-Coupled Photonics

We use the setup first described in [7]. The system consists of an electrically modulated Mach–Zehnder interferometer, which introduces nonlinearity into the system and modulates the intensity of a laser beam. This laser is fed into a very long optical fiber, which acts as a physical delay line. At the other end of the fiber, the light intensity is detected, converted to an electric signal, filtered and amplified, and added to an external input signal, before finally serving as the voltage signal that modulates the Mach–Zehnder interferometer. A schematic depiction is shown in Fig. 1. This particular system's state variable, which we denote by $a(t)$, can be described by the following equation (taken from [7], but manipulated to a more compact form):

$$\tau \dot{a}(t) = -a(t) + \sin(\mu a(t - D) + z(t) + \phi)/2. \quad (1)$$

Here, τ is the time scale of the low-pass filtering operation in the electronic amplifier, μ is the total amplification, D is the delay of

Manuscript received January 16, 2014; revised April 6, 2014, July 21, 2014, and July 23, 2014; accepted July 26, 2014. Date of publication August 15, 2014; date of current version June 16, 2015. This work was supported in part by the Interuniversity Attraction Pole Photonics@be, Belgian Science Policy Office, Paris, France, and in part by the European Research Council through NaResCo Project.

M. Hermans and P. Bienstman are with the Department of Information Technology, Ghent University, Gent 9000, Belgium (e-mail: michiel.hermans@ugent.be; peter.bienstman@ugent.be).

J. Dambre is with the Department of Electronics and Information Systems, Ghent University, Gent 9000, Belgium (e-mail: joni.dambre@ugent.be).

Digital Object Identifier 10.1109/TNNLS.2014.2344002

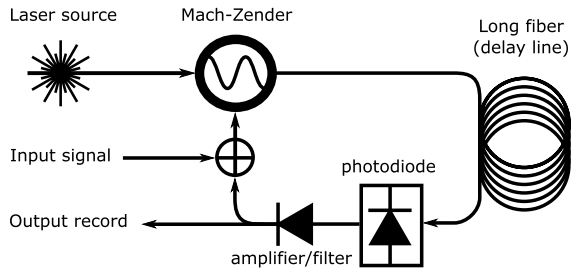

Fig. 1. Schematic depiction of a DCMZ interferometer.

the system, $z(t)$ is the external input signal, and ϕ is a constant offset. We will consider $\phi = 0$ for the remainder of this paper, as we will include an offset in the input signal. For ease of notation, we will denote the system with delay-coupled Mach–Zehnder (DCMZ). In this paper, all work is performed in simulation, and all parameters are considered to be dimensionless.

Delay-coupled systems of this sort can be used as reservoirs by applying time multiplexing, of which we provide a brief overview here, and refer the readers to [12] for more details. First of all, we consider a multivariate discrete input time series, which we will denote by $\mathbf{s}[i]$, for $i \in \{1, 2, \ldots, S\}$, S the total number of samples. This input time series is what we wish to process. We convert it to the continuous time signal $z(t)$ as follows:

$$z(t) = m_0(t) + \sum_{k=1}^{d_\mathbf{s}} m_k(t - i\,P_m) s_k[i] \quad \text{for } P_m < t < (i+1)P_m. \tag{2}$$

Here, $d_\mathbf{s}$ is the dimensionality of the input time series, and $m_k(t)$ are masking signals and P_M is the masking period. In other words, each sample of the input time series is converted into a continuous-time signal. Once this conversion has taken place, $z(t)$ is used to drive the system. The state variable $a(t)$ is recorded at a high sample frequency, and the time trace during one masking period then serves as the reservoir state. This means that a discrete-time system is essentially transformed into a continuous-time system, where each time step from the discrete-time interpretation, corresponds to a time segment of length P_M. The discrete-time input sequence now corresponds to a time concatenation of time segments of length P_m. The input signal $z(t)$ and the state variable $a(t)$, even though they are scalar, can show considerable time variation during one masking period, and this time trace itself then encodes high-dimensional information.

The sample vectors of $a(t)$ during one masking period are commonly interpreted as representing virtual nodes. Indeed, one interpretation of the DCMZ is that the delay line acts as a shift register, which stores values that can be considered node (neuron) activations. One after the other, they are propagated through the system nonlinearity where the low-pass filter operation and the delay coupling allows different nodes to interact with each other and the input signal.

Formally, we define $\mathbf{a}[i]$ as the reservoir state, corresponding with the ith input sample

$$\mathbf{a}[i+1] = [a\,(i\,P_m + \delta)\,; a\,(i\,P_m + 2\delta)\,; \cdots\,; a\,(i\,P_m + N_s\delta)\,; 1]$$

where $\delta = P_m/N_s$, the time-separation between virtual nodes, and N_s is the number of samples during one masking period. This is the vector, which is send to the reservoir output and, which contains information on the local history of the input. Note that the final element is a constant equal to one by definition, which is needed to provide any constant bias offset to the output. The output series is defined as

$$\mathbf{o}[i] = \mathbf{U}\mathbf{a}[i] \tag{3}$$

where N_s is the number of samples taken of $a(t)$ during each masking period, and U are the readout weights, a matrix of size $(N_s+1) \times d_\mathbf{o}$, with $d_\mathbf{o}$ the dimensionality of the output. All related publications up to this point have considered the masking signals $m_k(t)$ as piecewise constant, with a number of intervals equal to the number of samples per masking period. Accordingly, we will use the same approach. We can define a matrix $\mathbf{M}$ of size $(d_\mathbf{s}+1) \times N_s$, such that

$$m_{k-1}(t) = \mathbf{M}_{kl} \quad \text{if } (l-1)\frac{P_m}{N_s} < t < l\frac{P_m}{N_s}. \tag{4}$$

We have provided a schematic depiction of the masking process in Fig. 2(a). In this paper, the elements of $\mathbf{M}$ will always be chosen from a standard normal distribution, and next either scaled (in the RC approach), or trained (in the BPTT training approach).

In summary, the DCMZ is fully described by (1) and the parameters τ, D, μ, ϕ, $\mathbf{M}$, and $\mathbf{U}$. In the RC approach, all that is trained are the output weights $\mathbf{U}$. The elements of $\mathbf{M}$ are chosen randomly, and all that is optimized is its global scaling, and the feedback strength μ, the two parameters that determine in which dynamical regime the DCMZ operates. The upside of this approach is its simplicity: the output weights can be determined in a single shot by solving a linear system of equations, and their determination does not depend on the internal operation of the system. This has the added advantage that RC can be applied even when the underlying dynamics of the system are unknown or uncertain [for instance if other nonlinear effects appear in the system than those described by (1)]. All that is required for RC is to record the feature vectors $\mathbf{a}[i]$.

The downside is that a random input mask leads to a random encoding of the input. More informative components of the input time series are given the same weight as components that may contain only noise, or otherwise carry no useful information for the task at hand. In particular, if the dimensionality of $\mathbf{s}[i]$ is of the same order as the dimensionality of the feature vectors $\mathbf{a}[i]$, random input weights may cause obfuscation of the relevant information. Indeed, this is illustrated by attempts to apply RC on the phoneme recognition, where in order to obtain competitive performance, extremely large reservoirs were needed [13], with tens of thousands of neurons. In the case of delay-coupled systems, this would translate to using a very long delay D and a very large number of samples N_s. A long delay, on the other hand, would require a longer optical fiber, which would lead to more attenuation and signal degradation, and would impair one of the system's greatest advantages: its speed.

III. Backpropagation Through Time

In [11], we have introduced an optimization strategy that can be used to train the parameters of continuous-time dynamical systems. It is able to fully account for all transient effects that occur within nonlinear dynamical systems, and—more importantly—is able to exploit these effects. The algorithm computes the gradient of a cost function with respect to a set of trainable parameters, and then uses this to update them. The gradient itself is computed by means of a continuous-time variant of BPTT.

For the full derivation and proof of the general-case algorithm, we refer to [11] and its supplementary material. Here, we will limit ourselves to providing the key equations necessary for a delay-coupled system like the DCMZ. More precisely, let us write (1) in a more general form

$$\dot{a}(t) = f(a(t), a(t-D), z(t)). \tag{5}$$

We represent the set of internal parameters of the DCMZ that we wish to adapt (μ and $\mathbf{M}$) by $\mathbf{p}$. Note that the dependence on $\mathbf{M}$ is in this case through the driving signal $z(t)$. The next step is to define a cost function C_T, which is a function of the output of the system and

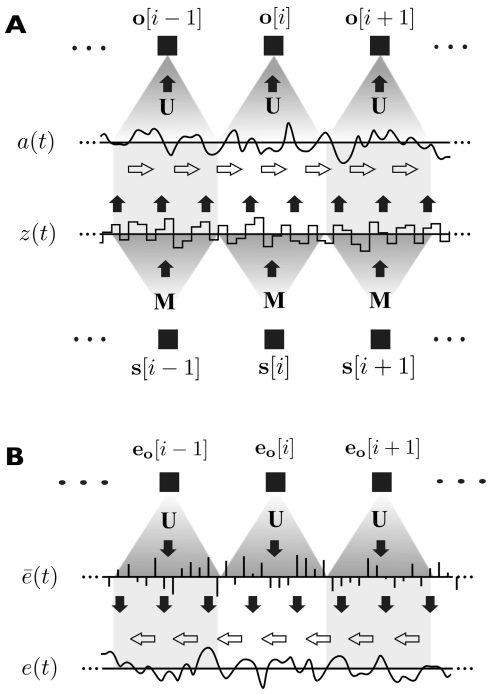

Fig. 2. A: Schematic depiction of the masking principle. Each instance of the time series $\mathbf{s}[i]$ is converted to a finite length signal through the masking weights $\mathbf{M}$. These are then concatenated in time to form $z(t)$, which in turn drives the dynamics that determine $a(t)$ (where the white arrow indicates the direction in which the differential equation progresses). Synchronously with the input segments, $a(t)$ is measured and multiplied with the output weights $\mathbf{U}$ to form the output time series $\mathbf{o}[i]$. B: Backpropagation process. Here, an external time series $\mathbf{e_o}[i]$ is converted into a continuous-time signal $\bar{e}(t)$, where now the masking weights come from $\mathbf{U}$ instead. As a consequence of the fact that the output is a function of samples of $a(t)$ at discrete moments in time, $\bar{e}$ is a series of scaled delta functions at the times of sampling. The variable $\bar{e}(t)$ in turn drives the differential equation that determines $e(t)$, which runs backward in time.

which we wish to minimize. In order to do this, we will compute the gradients $\mathbf{g_p}$ and $\mathbf{g_U}$ of the total cost with respect to the parameter sets $\mathbf{p}$ and $\mathbf{U}$, respectively

$$\mathbf{g_p} = \frac{dC_T}{d\mathbf{p}} \qquad \mathbf{g_U} = \frac{dC_T}{d\mathbf{U}}. \tag{6}$$

The gradient $\mathbf{g_U}$ can be computed straightforwardly from the analytic expression of the cost function. In [11], we show that

$$\mathbf{g_p} = \int_0^T e(t)\mathbf{k}(t) \tag{7}$$

with T the total time span in which the system state $a(t)$ is defined, and where

$$\mathbf{k}(t) = \frac{\partial f(a(t), a(t-D), z(t))}{\partial \mathbf{p}}$$

and the variable $e(t)$ is the error signal, which satisfies the following differential equation:

$$\dot{e}(s) = \bar{e}(s) + j(s)e(s) + j_D(s-D)e(s-D) \tag{8}$$

where $s = T - t$, which means the differential equation is solved backward in time, and

$$\bar{e}(t) = \frac{\partial C_T}{\partial a(t)}$$
$$j(t) = \frac{\partial f(a(t), a(t-D), z(t))}{\partial a(t)}$$
$$j_D(t) = \frac{\partial f(a(t), a(t-D), z(t))}{\partial a(t-D)}.$$

In this particular case, the signal $\bar{e}(t)$ is formed in a very similar way as the masking process described earlier. In this paper, we can define the cost function as the sum of the costs of each output instance $\mathbf{o}[i]$

$$C_T = \sum_{i=0}^{S} c(\mathbf{o}[i]).$$

The chain rule then leads to

$$\bar{e}(t) = \frac{\partial C_T}{\partial a(t)} = \sum_{i=0}^{S} \frac{\partial c(\mathbf{o}[i])}{\partial \mathbf{o}[i]} \frac{\partial \mathbf{o}[i])}{\partial \mathbf{a}[i]} \frac{\partial \mathbf{a}[i]}{\partial a(t)}.$$

The factor $\partial c(\mathbf{o}[i])/\partial \mathbf{o}[i]$ corresponds to a multivariate time series, which we will denote by $\mathbf{e_o}[i]$. The factor $\partial \mathbf{o}[i]/\partial \mathbf{a}[i]$ corresponds to the output weights $\mathbf{U}$ and the factor $\partial \mathbf{a}[i]/\partial a(t)$ corresponds to a set of delta functions, which are centered at the moments at which the samples of the feature vectors $\mathbf{a}[i]$ are drawn. This means that $\bar{e}(t)$ is formed in almost the same manner as the input signal $z(t)$ with the elements of $\mathbf{U}$ now serving as the masking weights. The only difference is that instead of a piecewise constant function, $\bar{e}(t)$ is a series of scaled delta functions. Note that in reality, a physical measurement will not be instantaneous (as we assume here for the samples that make up $\mathbf{a}[i]$), but rather take an average over a certain time span. If for instance, the measurement takes the average over one masking time step δ, $\partial \mathbf{a}[i]/\partial a(t)$ would lead to a piecewise constant function as well.

Each element of $\mathbf{k}(t)$ stands for a different parameter. Note that the partial derivative with respect to μ can be derived straightforwardly from (1). To compute the partial derivative with respect to the elements of $\mathbf{M}$, we need to use the chain rule

$$\frac{\partial f(a(t), a(t-D), z(t))}{\partial \mathbf{M}} = \frac{\partial f(a(t), a(t-D), z(t))}{\partial z(t)} \frac{\partial z(t)}{\partial \mathbf{M}}.$$

The first factor can be computed straightforwardly from (1). The second can be derived from the definition of $z(t)$ from (2) and (4).

The process of computing $e(t)$ is depicted in Fig. 2(b). Once the gradients are computed, they can be used to update the parameter set $\mathbf{p}$ as follows:

$$\mathbf{p} \leftarrow \mathbf{p} - \eta \mathbf{g_p} \qquad \mathbf{U} \leftarrow \mathbf{U} - \eta \mathbf{g_U}$$

with η a (small) learning rate.

In [11], a photonic application of BPTT was already demonstrated. It was shown that circuits of semiconductor optical amplifiers could be optimized to perform digital operations. However, the example was concerned with training the physical design parameters of the system, which requires an optimization phase in simulation, and next the actual fabrication of the device. In the DCMZ described in Section II, the parameters describing the output weights, masking weights, and feedback strength are set electronically. Optimizing them can be validated easily and quickly. In this paper, we explore the use of gradient descent to train the weights $\mathbf{M}$, $\mathbf{U}$ and the feedback strength μ all simultaneously.

IV. Phoneme Recognition

Previous work with delay-coupled reservoirs, including the DCMZ, has among others focused on a spoken digit recognition task. The goal was to classify spoken digits form zero to nine. The classification error, however, is close to zero in most papers on the subject, and it serves more as an example task to demonstrate the potential of the system than for comparative reasons. When we attempted this task using BPTT, we consistently got a zero classification error. Therefore, we used the far more challenging TIMIT data set, which is often used for machine learning benchmarking.

A. TIMIT Data Set

The goal is to classify phonemes, which are the smallest segmental unit of sound employed to form meaningful contrasts between utterances. We used the internationally renowned TIMIT speech corpus [14]. The speech is labeled by hand for each of the 61 existing phonemes, which was reduced to 39 symbols as proposed in [15]. The TIMIT corpus has a predefined train and test set with different speakers. The speech has been preprocessed using mel frequency cepstral coefficient analysis [16]. The data has a dimensionality of $d_{\mathbf{s}} = 39$.

For optimizing metaparameters, we chose a validation set using roughly one tenth of the data in the training set. In this paper, we only consider the frame error rate (FER), which is the fraction of frames (time steps) that have been misclassified. Note that in true speech recognition applications, one is more interested in the phoneme error rate, which can be measured after the individual frames have been clustered into a symbolic string of phonemes. In this paper, we are only interested in the core mechanism that processes the data, i.e., the DCMZ, so we omit this step here.

B. Model Details

We make a comparison between the RC approach and the full training approach as performed by BPTT. For both setups, we consider systems, which take $N_s = 500$ samples. In accordance to previous paper, we choose $\tau = 5P_m/N_s$, such that the low-pass filtering operation spans roughly the time span of five virtual nodes. In the case of the RC approach, the parameters μ, D, and the global scaling of $\mathbf{M}$ are optimized on the validation set by means of a random search, where we picked the best values found over 100 trials. In principle, when using BPTT, such a search is not necessary. Final performance, however, will still depend on the initial scalings of the trained parameters, so here too we did a (small) search for good initial scaling values for $\mathbf{M}$ and μ, based on only 20 trials. Note that in most work on the DCMZ, the delay D and the masking period P_m are considered equal, such that each virtual node receives its own previous activation value as input, and all mixing interaction between virtual nodes is due to the low-pass filter operation. Other work has used the DCMZ setup without any low-pass filtering [6], and here a mismatch between D and P_m is necessary to provide a mixing interaction between different nodes. Here, we found that the exact delay parameter mattered very little in either the RC approach or the fully trained network, as long it was close to the masking period. In principle, it is possible to train D as well. We found, however, that it always settled immediately to a value very near to its initial value (less than the length of a single mask step δ), such that it was of little use.

In [17], it was shown that the performance of DCMZs is limited by output noise. More precisely, by the fact that the readout happens by an analog-to-digital converter with a limited number of bits (quantization noise). In order to take this effect into account, we included 8-bit quantization (such that its values are quantized to 256 different levels) to the readout in all experiments, both during training and testing. This ensures that BPTT does not find a nonrobust solution, which would fail under realistic quantization levels. Due to the fact that the quantization operation is nondifferentiable, we did not include it in the error backpropagation phase (treating it as if it was not there).

As the task at hand is a classification task, we add a so-called softmax function to the output (see for instance [18, Ch. 4]), generating a probability for a certain output to belong to each class. The system is trained to minimize the cross-entropy between the output probabilities and the given probabilities (provided by the output labels: one for the correct class and zero for all the others).

TABLE I
RESULTS ON THE TIMIT PHONEME DATA SET, EACH AVERAGED OVER 10 EXPERIMENTS

Model	FER test	Computing time
Reservoir, linear regression	43.9 %	6 min
Reservoir, cross-entropy	41.1 %	8.2 h
BPTT, cross-entropy	31.1 %	18.8 h
BPTT, cross-entropy, $\mathbf{U}$ fixed	36.9 %	18.4 h
Continuous Density HMM [19]	25.0 %	

We performed two additional sanity checks. First of all, we tried out the more classic RC approach for training the output weights, where they were determined using linear regression, and the cost function is the squared error between the output and the labels. Even though this approach is suboptimal for classification tasks, linear regression has the advantage that the output weights can be determined in a single shot. The other test we did was to train the DCMZ with BPTT, but with random and fixed output weights (scaled using the average scaling of trained output weights). The reasoning behind this test is the fact that, when we compare BPTT to the RC approach, BPTT has the inherent advantage that we simply train more parameters, which should always lead to better performance. In this case, the number of parameters for the input weights $\mathbf{M}$ and the output weights $\mathbf{U}$ are roughly the same, such that we can measure directly which of the two is more useful to train.

C. Training Details

For all experiments (except the one where the outputs were trained using linear regression), we used 500 000 training iterations. As the training set is far too large to compute each single gradient instance on, each iteration we selected five sequences of 50 time steps, and used the sum of the gradients to perform a single update, such that we essentially used stochastic gradient descent. Given the fact that the training set contains a little over 10^6 time steps in total, this means that the training runs through the full training set roughly 100 times.

We chose the learning rate η at $5 \cdot 10^{-4}$ at the start of training, and decreased it with a fixed amount each training iteration until it reached zero after 500 000 iterations. The parameters that are trained are $\mathbf{U}$, $\mathbf{M}$, and μ, which have sizes 500×40, 39×501 and a scalar value, respectively.

V. RESULTS

The resulting performances on the test set for the three used approaches are shown in Table I. The DCMZ trained with BPTT outperforms both reservoir implementations by a relatively large amount. Interestingly, the approach with random and fixed output weights and trained masking weights also yields significantly better performance than the reservoir approach. This indicates that an optimized input encoding is highly desirable compared with a random one as used in the RC approach. Indeed, if input parameters are trained, they can engage in a useful interaction with the internal dynamics of the system, optimally extracting the required information from the input stream. Training output weights cannot adapt the dynamics of the DCMZ, and therefore optimizing them can only extract useful information, which happens to reside within the reservoir.

The bottom result in Table I is the lowest FER on the TIMIT data set mentioned in literature, shown for comparison. Note, however, that most literature does not mention FER for the reasons mentioned in Section IV-A, such that even better performances have very likely been attained in other work.

The computing times for each method are also provided in Table I. The RC approach, while having inferior performance, still has the

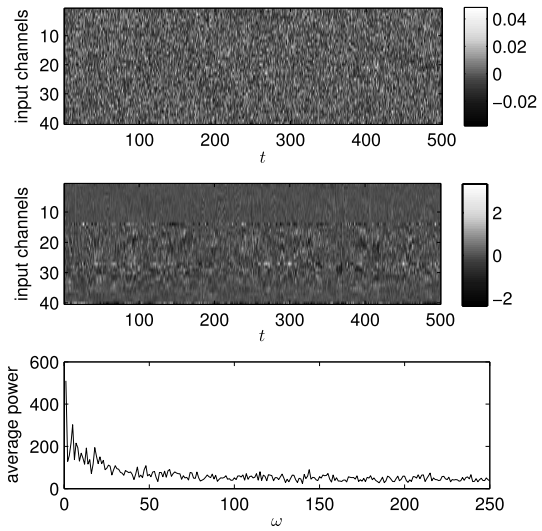

Fig. 3. Top panel and middle panel: input mask **M** before and after training, respectively. Vertical axis: different input channels of the 39-dimensional feature vector. Horizontal axis: time. Bottom panel: average power spectrum of the trained input masks $m_i(t)$.

advantage of very great speed compared with iterative gradient descent methods. The two BPTT approaches take roughly twice as long as the RC approach trained with cross-entropy. This makes sense as in BPTT, each simulation of the DCMZ is accompanied by a backward error propagation simulation, which has roughly the same computational requirements as the forward simulation (simulation of a differential equation of the same dimensionality and over the same time span).

In order to gain some understanding of what is obtained by training **M**, we visualized the elements of the input masks before and after training in the top and middle panel of Fig. 3, where the vertical axis stands for different input channels, and the horizontal axis is the time direction. We showed those of the experiment in which both input and output weights were trained. First of all, it turns out that BPTT significantly increases the overal scaling of the input weights, pushing the DCMZ in a strongly nonlinear regime. Furthermore, it appears that different input channels have been scaled differently, which indicates that BPTT is able to give different parts of the input a proper weighting factor depending on how informative they are. The bottom panel of Fig. 3 shows the average power spectrum of the input masking signals. Note that fully random weights would lead to a flat spectrum. It appears that lower frequency components in $m(t)$ are increased during training. This may partially be due to the low-pass filter operation within the network, which reduces the transfer of higher frequencies and therefore makes them less useful to encode information.

One final noteworthy observation is that during the training process, the amplification factor μ consistently grows and arrives at large values. In our model, the DCMZ is asymptotically stable with a single fixed point when $\mu \leq 2$ (the so-called edge of stability known from echo state networks [1]). Here, we found, however, that μ settled at values closer to 4, far beyond this boundary. Indeed, in the absence of input ($z(t) = 0$), the trained DCMZs show strong oscillatory behavior and do not settle to a stable fixed point.

VI. Conclusion

Optoelectronic delay-coupled systems offer a promising platform for serving as nonconventional data processing devices. We have shown that it is possible to use a machine learning algorithm known as BPTT to radically improve their performance on a challenging phoneme recognition task. Instead of considering the system as random, we were able to directly optimize the full set of system parameters. This lead to a far better input representation of the data, in turn leading to good performance.

The demonstrated success of the approach has profound implications for the research field of optoelectronic computation. Not only can BPTT lead to far more efficient and robust processing, combined with the DCMZs great speed and high degree of parallelism, it could also provide the boost that makes photonic systems competitive with more conventional processors. The presented optimization approach can be applied on any photonic component, which has a dynamical character (or even just internal delays), and this includes microscopic on-chip photonic circuits. This means that it can be used beyond only machine learning applications, but also to optimize the behavior of basic photonic components.

Future work in this direction will need to overcome several challenges. The system as was used in this paper has a total of 39 000 trainable parameters, yet this is very modest compared with common well-performing machine learning models, which often have hundreds of thousands or even several millions of trainable parameters [20]. There is little doubt that such a scale-up is necessary to tackle truly difficult problems in time series processing, such that one of the most pressing questions is how we can redesign the DCMZ, or similar setups, such that they too are able to embed such a large number of parameters.

References

[1] H. Jaeger, "Short term memory in echo state networks," German National Research Center for Information Technology, Sankt Augustin, Germany, Tech. Rep. GMD Report 152, May 2001.
[2] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Netw.*, vol. 20, no. 3, pp. 391–403, Apr. 2007.
[3] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication," *Science*, vol. 308, no. 5667, pp. 78–80, Apr. 2004.
[4] K. Vandoorne *et al.*, "Toward optical signal processing using photonic reservoir computing," *Opt. Exp.*, vol. 16, no. 15, pp. 11182–11192, Jul. 2008.
[5] K. Vandoorne, J. Dambre, D. Verstraeten, B. Schrauwen, and P. Bienstman, "Parallel reservoir computing using optical amplifiers," *IEEE Trans. Neural Netw.*, vol. 22, no. 9, pp. 1469–1481, Sep. 2011.
[6] Y. Paquot *et al.*, "Optoelectronic reservoir computing," *Sci. Rep.*, vol. 2, pp. 1–6, Feb. 2012.
[7] L. Larger *et al.*, "Photonic information processing beyond turing: An optoelectronic implementation of reservoir computing," *Opt. Exp.*, vol. 20, no. 3, pp. 3241–3249, Jan. 2012.
[8] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature Commun.*, vol. 4, p. 1364, Jan. 2013.
[9] D. Rumelhart, G. Hinton, and R. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986.
[10] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Netw.*, vol. 1, no. 4, pp. 339–356, 1988.
[11] M. Hermans, B. Schrauwen, P. Bienstman, and J. Dambre, "Automated design of complex dynamic systems," *PloS One*, vol. 9, no. 1, p. e86696, Jan. 2014.
[12] L. Appeltan *et al.*, "Information processing using a single dynamical node as complex system," *Nature Commun.*, vol. 2, p. 468, Sep. 2011.
[13] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, "Phoneme recognition with large hierarchical reservoirs," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2010, pp. 2307–2315.
[14] J. Garofolo *et al.*, *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. Philadelphia, PA, USA: Linguistic Data Consortium, 1993.

[15] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 37, no. 11, pp. 1641–1648, Nov. 1989.

[16] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 28, no. 4, pp. 357–366, Aug. 1980.

[17] M. C. Soriano *et al.*, "Optoelectronic reservoir computing: Tackling noise-induced performance degradation," *Opt. Exp.*, vol. 21, no. 1, pp. 12–20, Jan. 2013.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). New York, NY, USA: Springer-Verlag, 2006.

[19] C. C. Cheng, F. Sha, and L. Saul, "A fast online algorithm for large margin training of online continuous density hidden Markov models," in *Proc. Interspeech*, Brighton, U.K., 2009, pp. 668–671.

[20] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, Dec. 2010.