КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2022 Т. 14 № 2 С. 399–416
DOI: 10.20537/2076-7633-2022-14-2-399-416

Ки&М

МАТЕМАТИЧЕСКИЕ ОСНОВЫ И ЧИСЛЕННЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ

УДК: 519.856

Стохастическая оптимизация в задаче цифрового предыскажения сигнала

А. В. Алпатов[1,a], Е. А. Петерс[2,b], Д. А. Пасечнюк[3,c], А. М. Райгородский[3,d]

[1]ГАОУ СО «Самарский лицей информационных технологий (Базовая школа РАН)»,
Россия, 443096, г. Самара, ул. Больничная, д. 14а
[2]ЦДНИТТ «УникУм» при КузГТУ,
Россия, 650000, г. Кемерово, ул. Красноармейская, д. 117
[3]Московский физико-технический институт,
Россия, 141701, г. Долгопрудный, Институтский пер., д. 9

E-mail: [a] artemalpatov2@gmail.com, [b] cool47.cool@gmail.com, [c] pasechnyuk2004@gmail.com, [d] mraigor@yandex.ru

Получено 15.01.2022.
Принято к публикации 13.02.2022.

В данной статье осуществляется сравнение эффективности некоторых современных методов и практик стохастической оптимизации применительно к задаче цифрового предыскажения сигнала (DPD), которое является важной составляющей процесса обработки сигнала на базовых станциях, обеспечивающих беспроводную связь. В частности, рассматривается два круга вопросов о возможностях применения стохастических методов для обучения моделей класса Винера – Гаммерштейна в рамках подхода минимизации эмпирического риска: касательно улучшения глубины и скорости сходимости данного метода оптимизации и относительно близости самой постановки задачи (выбранной модели симуляции) к наблюдаемому в действительности поведению устройства. Так, в первой части этого исследования внимание будет сосредоточено на вопросе о нахождении наиболее эффективного метода оптимизации и дополнительных к нему модификаций. Во второй части предлагается новая квази-онлайн-постановка задачи и, соответственно, среда для тестирования эффективности методов, благодаря которым результаты численного моделирования удается привести в соответствие с поведением реального прототипа устройства DPD. В рамках этой новой постановки далее осуществляется повторное тестирование некоторых избранных практик, более подробно рассмотренных в первой части исследования, и также обнаруживаются и подчеркиваются преимущества нового лидирующего метода оптимизации, оказывающегося теперь также наиболее эффективным и в практических тестах. Для конкретной рассмотренной модели максимально достигнутое улучшение глубины сходимости составило 7 % в стандартном режиме и 5 % в онлайн-постановке (при том что метрика сама по себе имеет логарифмическую шкалу). Также благодаря дополнительным техникам оказывается возможным сократить время обучения модели DPD вдвое, сохранив улучшение глубины сходимости на 3 % и 6 % для стандартного и онлайн-режимов соответственно. Все сравнения производятся с методом оптимизации Adam, который был отмечен как лучший стохастический метод для задачи DPD из рассматриваемых в предшествующей работе [Pasechnyuk et al., 2021], и с методом оптимизации Adamax, который оказывается наиболее эффективным в предлагаемом онлайн-режиме.

Ключевые слова: цифровое предыскажение, обработка сигнала, стохастическая оптимизация, онлайн-обучение

Работа Д. А. Пасечнюка выполнена при финансовой поддержке гранта поддержки ведущих научных школ НШ-775.2022.1.1. Работа А. М. Райгородского поддержана грантом Российского научного фонда (проект № 21-71-30005).

COMPUTER RESEARCH AND MODELING
2022 VOL. 14 NO. 2 P. 399–416
DOI: 10.20537/2076-7633-2022-14-2-399-416

КиМ

MATHEMATICAL MODELING AND NUMERICAL SIMULATION

UDC: 519.856

Stochastic optimization in digital pre-distortion of the signal

A. V. Alpatov[1,a], E. A. Peters[2,b], D. A. Pasechnyuk[3,c], A. M. Raigorodskii[3,d]

[1]Samara Lyceum of information technologies (Basic school of the RAS),
14a Bol'nichnaya st., Samara, 443096, Russia
[2]CCSETC at KuzSTU «UnicUm»,
117 Krasnoarmeyskaya st., Kemerovo, 650000, Russia
[3]Moscow Institute of Physics and Technology,
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia

E-mail: [a] artemalpatov2@gmail.com, [b] cool47.cool@gmail.com, [c] pasechnyuk2004@gmail.com,
[d] mraigor@yandex.ru

Received 15.01.2022.
Accepted for publication 13.02.2022.

In this paper, we test the performance of some modern stochastic optimization methods and practices with respect to the digital pre-distortion problem, which is a valuable part of processing signal on base stations providing wireless communication. In the first part of our study, we focus on the search for the best performing method and its proper modifications. In the second part, we propose the new, quasi-online, testing framework that allows us to fit our modeling results with the behavior of real-life DPD prototype, retest some selected of practices considered in the previous section and approve the advantages of the method appearing to be the best under real-life conditions. For the used model, the maximum achieved improvement in depth is 7 % in the standard regime and 5 % in the online regime (metric itself is of logarithmic scale). We also achieve a halving of the working time preserving 3 % and 6 % improvement in depth for the standard and online regime, respectively. All comparisons are made to the Adam method, which was highlighted as the best stochastic method for DPD problem in [Pasechnyuk et al., 2021], and to the Adamax method, which is the best in the proposed online regime.

Keywords: digital pre-distortion, signal processing, stochastic optimization, online learning

Citation: *Computer Research and Modeling*, 2022, vol. 14, no. 2, pp. 399–416.

The research of D. Pasechnyuk was supported by the program «Leading Scientific Schools» (grant No. NSh-775.2022.1.1).
The research of A. Raigorodskii was supported by Russian Science Foundation grant (project No. 21-71-30005).

Introduction

Consider a base station providing some form of wireless communication. Let it be required to transmit with its help some given signal. The signal is represented as a sampled array $x \in \mathbb{C}^m$. One of the transformations that must be applied to the signal before transmission is amplification. To carry out this transformation, the PA device is most often used, whose action on the signal, ideally, has the form $\mathcal{PA}(x) = a \cdot x$, where $a \gg 1$. However, hardware imperfections lead to a deviation from this law; in particular, the $\mathcal{PA}(x)$ signal has out-of-band spurious harmonics, resulting in interference. This is illustrated in Fig. 1.

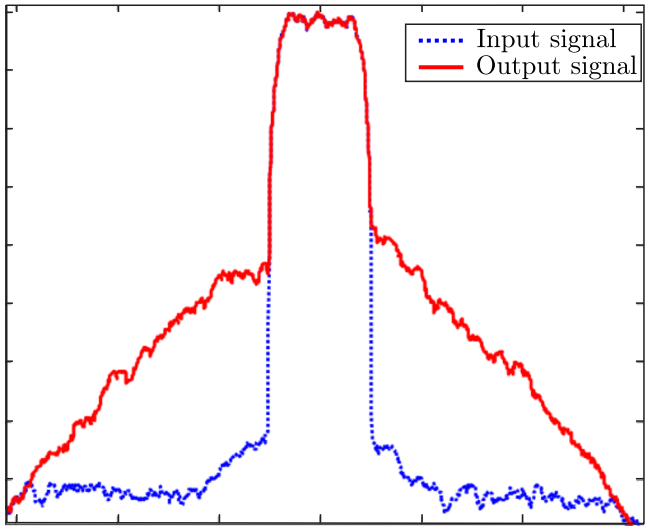


Figure 1. Power spectral density plot of the original signal (Input signal, blue dotted line) and the power amplified signal (Output signal, red line). Picture from *Wisell D. H.* Exploring the sampling rate requirements for behavioural amplifier modelling // XVIII IMEKO World Congress. — 2006. — p. 1–4

To prevent noise in the transmitted signal, there is a digital pre-distortion technology. Specifically, there is a DPD device acting on the signal just before the PA action, so that the result is the signal $\mathcal{PA}(\mathcal{DPD}(x))$. If we omit the factor a, which is easy to accomplish by scaling the signal, then we can see that for an ideal action it must hold that $\mathcal{DPD} = \mathcal{PA}^{-1}$ (Fig. 2). Thus, the task is reduced to the inversion of $\mathcal{PA}$ function. However, the $\mathcal{PA}$ function actually depends on the characteristics of the environment, the signal x itself, and on time, so it can only be inverted numerically. To do this, we choose a certain parametric family of functions $\{\mathcal{DPD}_\theta\}_{\theta\in\Theta}$ to describe DPD and choose such a parametrization θ in which $\mathcal{PA} \circ \mathcal{DPD}_\theta$ is closest to the identical transformation.

In this study, to model the behavior of DPD we use the Wiener – Hammerstein cascade models family [Ghannouchi, Hammi, Helaoui, 2015]. The structure of these models is very similar to the structure of a multilayer neural network, but instead of the neurons it consists of cells specific for signal processing. Figure 3 shows a schematic diagram of such a cell with polynomial nonlinearity within.

Now that we have defined the parametrization for DPD, we need to provide a more optimization-friendly formulation to the problem of inverting the $\mathcal{PA}$ function. The point is that to do this we need to have the perfect pre-distortion $\overline{y}$ for some signal x, to train the model to repeat this predistortion, i. e. to exactly invert the $\mathcal{PA}$ function. Using the standard empirical risk minimization framework, we then obtain some optimization problem with the sum-like function. The following formulation, proposed and described in detail in [Pasechnyuk et al., 2021], is the central one for this paper (we denote the kth

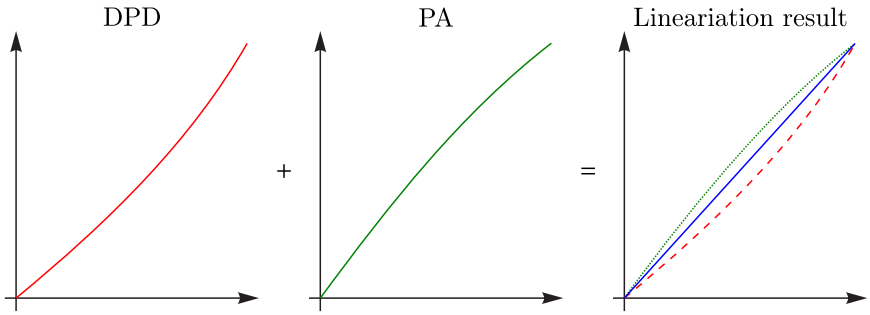


Figure 2. The intuitive principle of digital pre-distortion: DPD acts as an inverse of power amplifiers function, so that the resulting transformation is near linear (identical) Picture from [Pasechnyuk et al., 2021]

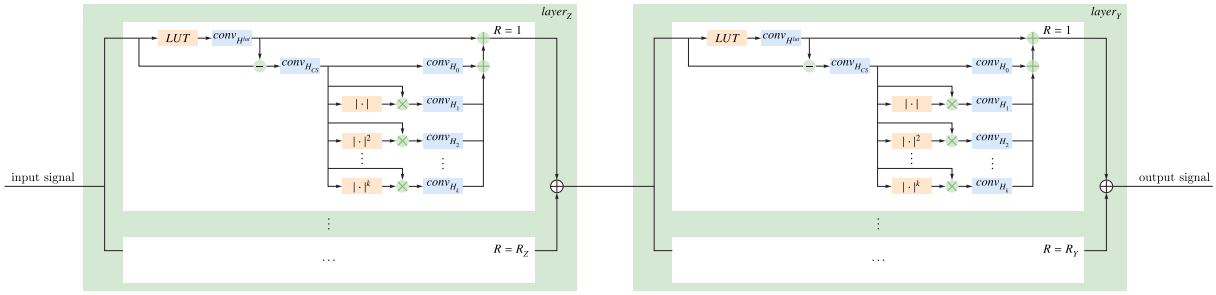


Figure 3. The structure of the Wiener – Hammerstein model cell with polynomial nonlinearity. Picture from [Pasechnyuk et al., 2021]

element in array x by $[x]_k$ and x_k).

$$\frac{1}{m}\sum_{k=1}^{m}\left|[\mathcal{DPD}_\theta(x)]_k - \overline{y}_k\right|^2 \to \min_\theta. \tag{1}$$

So, both problems of this study correspond to this problem formulation and concern the question of how to solve it properly, taking into account its original motivation.

RQ_1 Which stochastic optimization method is the most suitable to solve the stated problem with such a specific data and model behind?

In broad strokes, this question was considered in the course of study [Pasechnyuk et al., 2021] for different groups of numerical methods. Here we focus on the stochastic methods (indeed, (1) can be easily randomized by terms), and consider several recently proposed approaches in this field.

RQ_2 How should one organize the training procedure and the evaluation of the model to agree with the practical behavior of DPD device?

The operation of the full DPD technology stack is more complicated than it is represented in (1), and the main difference is its online nature. It is even more online than it is implied, for example, in the expectation minimization problems, because the source of the signal (and hence a joint distribution of $(x, \overline{y})$, continuing the analogy) also changes in time. We propose a way to simulate this specificity, and eliminate contradictions between our experimental results and some results of laboratory tests with DPD device.

Standard learning framework

In this section we consider the standard setup for the problem stated above, i. e., we train the model on some known data (it is important for the signal processing task that this subset of the whole data is not randomly chosen but is the prefix of the ordered signal array) and then test the model on the whole signal (NB: not the remaining part, as usual). In our experiments, the length of the training signal prefix is 75 % of the whole signal's length (albeit it is deliberately more than needed, see [Pasechnyuk et al., 2021] for details, to avoid any approximation issues that are out of the scope this time). We use the segment with $m = 2 \cdot 10^5$ ticks from the real-life 80 MHz signal as a dataset. As we shall see, this way to train/test the model is slightly naïve to represent the behavior of the real-live DPD device (being standard and widely used nonetheless), but thanks to it we can easily test many possible techniques under the simplest (and computationally inexpensive) conditions to select the few best to work with further.

To assess the results of evaluating the model on the test split of the dataset, we use the following logarithmic scaling analogue of Mean Squared Error metric with normalization, which is specific to the signal processing tasks:

$$\mathrm{NMSE}(y, \overline{y}) := 10 \cdot \log_{10} \frac{\sum\limits_{k=1}^{m} |y_k - \overline{y}_k|^2}{\sum\limits_{k=1}^{m} |x_k|^2} \text{ dB}.$$

The source code of the model used to obtain the following experimental results is available at https://github.com/dmivilensky/Sirius-2021-DPD-optimization (Python 3 + PyTorch). All the computations were performed on the standard personal computer without GPU acceleration.

Comparison of algorithms

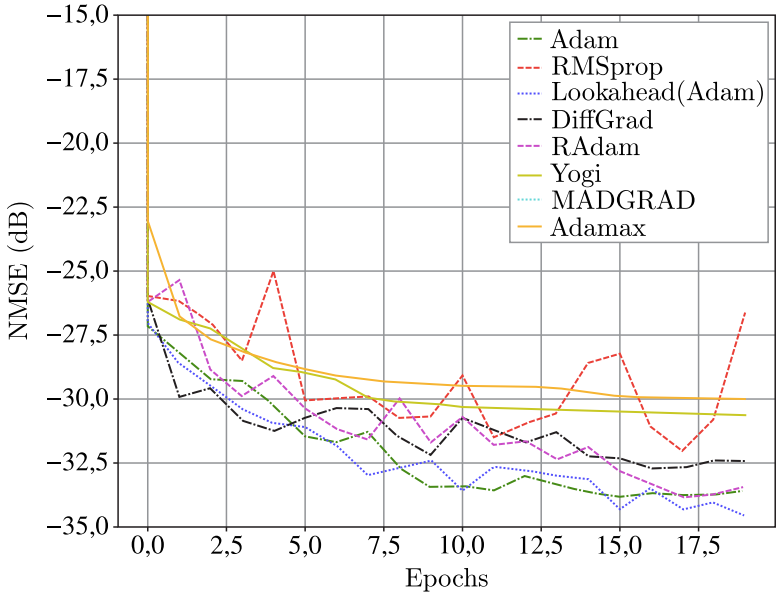


Figure 4. The convergence curves of different optimization methods in standard regime at initial epochs

Let us start by testing different optimization methods with respect to this standardly formulated learning problem. The convergence of some most known ones was examined in [Pasechnyuk et al., 2021] recently, so there is no need for us to present the same results for our model: compared to

the table shown in Fig. 5, they are just transposed a couple of decibels higher (in our research, we use the Wiener – Hammerstein model with significantly less parameters than for the model considered in [Pasechnyuk et al., 2021], so our score will be a little worse — it is just a specific feature of the particular used model).

Method	Setup	dB	
		$t = 0$ s	$t = 300$ s
ASGD	(128, 10,0)		−28,976
Adadelta	(2048, 10,0)		−32,697
Adagrad	(2048, 0,01)		−36,608
Adam	(2048, 0,001)	−15,616	−38,129
Adamax	(2048, 0,01)		−37,934
RMSprop	(2048, 0,001)		−36,499
SGD	(128, 10,0)		−34,061
FastAdaptive			−36,273

Figure 5. Summary of the result of training the DPD model with different optimization methods in the standard setting. Table from [Pasechnyuk et al., 2021]

Besides, in this section we focus on the starter methods, i. e., these whose convergence is the most efficient among all tested methods at the beginning of the training process. That's why we show the convergence of all the methods on the first 20 or 40 epochs. The first reason for this is that the long-term convergence in the standard training regime has already been considered in [Pasechnyuk et al., 2021], so we should clarify the short-term convergence, which has not been considered (it is important for the multi-method optimization schemes). The second is that the convergence on the late epochs and the depth of finally obtained minimum is the question of a late convergence rate of the method rather than the question of convergence depth, and we will see in the online learning framework that the methods presented as the best here are not well adopted to converge with a good rate at a late epoch, which is important in practice, because the optimal point we seek for is changing together with the data segment. So, let us avoid all these complicated questions in this section, and focus on the starting convergence, i. e., the behavior of methods far from the local minimum.

So, in Fig. 4, Fig. 6 and Fig. 7 we present the convergence curves of different tested optimization methods. Further, we describe our observations on it.

1. The starting convergence of Adamax is not very good, despite the fact that it is one of the most efficient methods in the long run. Some of the methods, like RMSprop, are just unstable for the model and the signal of used size.

2. The best among the methods from Fig. 4 are Adam, as expected, and Lookahead(Adam) (we also denote it by LaAdam), i. e., the Adam enveloped with Lookahead algorithm from [Zhang et al., 2019]. The latter demonstrates the faster convergence.

3. The only method that outperforms Adam is Shampoo [Gupta, Koren, Singer, 2018], it is the best starter method for DPD training for now.

4. Lookahead gives very notable results. It significantly improves and stabilizes the convergence of Adam, which is shown in Fig. 6. Unfortunately, there is no effect of Lookahead on Shampoo. Nevertheless, convergence of Shampoo is phenomenally stable without Lookahead too.

Another novel optimization method we tested is the Accelerated mini-batch SGD version from [Woodworth, Srebro, 2021] (we name it AccMbSGD). This method is specialized for the

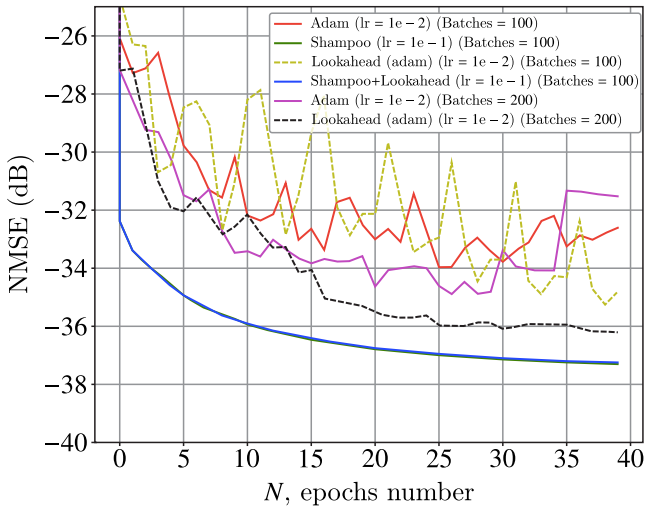


Figure 6. Comparison of Adam and Shampoo [Gupta, Koren, Singer, 2018] optimization methods, and the effect of Lookahead envelope on the training of the DPD model

overparametrized optimization problems. The point is that in our case the number of parameters in the model is significantly larger than the number of terms in sum in empirical risk, so it is reasonable to assume that value of NMSE in minimum is near zero (and it is). This is the characteristic property of overparametrization, and the above-mentioned method is theoretically optimal for this case.

5. Unfortunately, the convergence of AccMbSGD is not so good as expected. Moreover, this method is complicated in hyperparameter tuning. It seems that it is not adopted for DPD training. The corresponding curves are presented in Fig. 7.

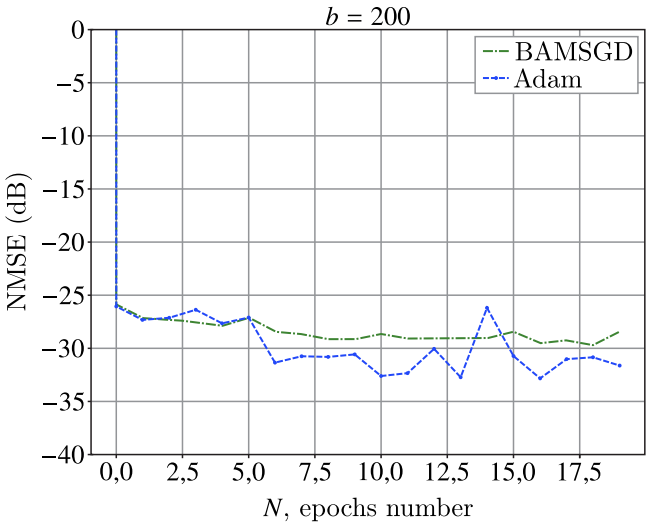


Figure 7. Comparison of Adam and AccMbSGD [Woodworth, Srebro, 2021] optimization methods as applied training of the DPD model

Dynamic batch size

It is well-known practice to use mini-batching in stochastic optimization methods to improve the convergence (due to the smaller variance of oracle) and speed up the calculations (it is typical

for training neural networks). Usually, the size of the batch is just a hyperparameter of the method and is the same at every iteration. Nevertheless, it is reasonable to strive to increase it if the latter is computationally efficient. On the other hand, this heuristic is quite legitimate: indeed, in the late steps of the method the current point is already close enough to the minimum and the step direction efficiency becomes more sensitive to the inaccuracies of the stochastic gradient approximation (which just can be reduced by increasing the size of the batch).

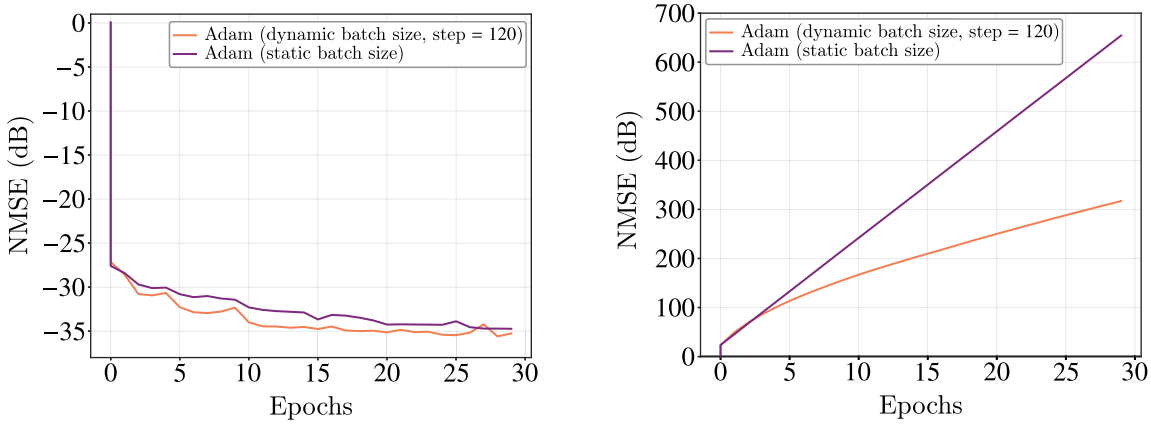
Some theoretical justifications on this technique are provided in [Zhao, Xie, Li, 2020]. The practical efficiency of batch size adaptation is also shown in [Devarakonda, Naumov, Garland, 2017] for some deep learning problems. So, further we check if the changing of the batch size affects the convergence of the Adam optimization method in our case. We considered both increasing and decreasing changing, as well as the different changing rates.

1. In our experiments, Adam with reducing the batch size demonstrated less stable and less deep convergence than the Adam with a fixed batch size (as expected).

2. The growth of the batch size following the exponential law leads to an improvement of convergence, but its effect is smaller than for the linear law.

3. The best formula for changing the batch size in our case is $b = 200 + 120 \cdot \text{epoch}$. Using it, we obtained a 3 % improvement in NMSE (Fig. 8, *a*) and reduced the training time to one half (Fig. 8, *b*). The results are summarized in Table 1.

4. The positive effect in training time increases with an increase in the coefficient in the latter linear formula (it is 120 there). But here we faced the trade-off, because the increase in the coefficient can also worsen the limit depth of convergence (value of NMSE). In the signal processing task the learning time is minor metric, so it is reasonable to stick around 120.



(a) The dependence of NMSE value (dB) on the number of training epochs

(b) The dependence of working time (s) on the number of training epoch

Figure 8. The effect of the batch size changing on the training of the DPD model. It leads to a deeper convergence of Adam (left plot) and a slower growth in the working time (right plot)

In the experiments above we used the Adam optimization method to demonstrate the efficiency of the dynamic batch size approach. The Adam was chosen as the leader among the methods for the standard learning framework [Pasechnyuk et al., 2021]. Nevertheless, these results are similar for all the tested optimization methods (they are listed in Table 4).

Table 1. Summary of the result of the training of the DPD model with a fixed and a changing batch size

	NMSE (dB)	time (s)
fixed	−34,90	660
changing	−35,92	320

Dynamic learning rate

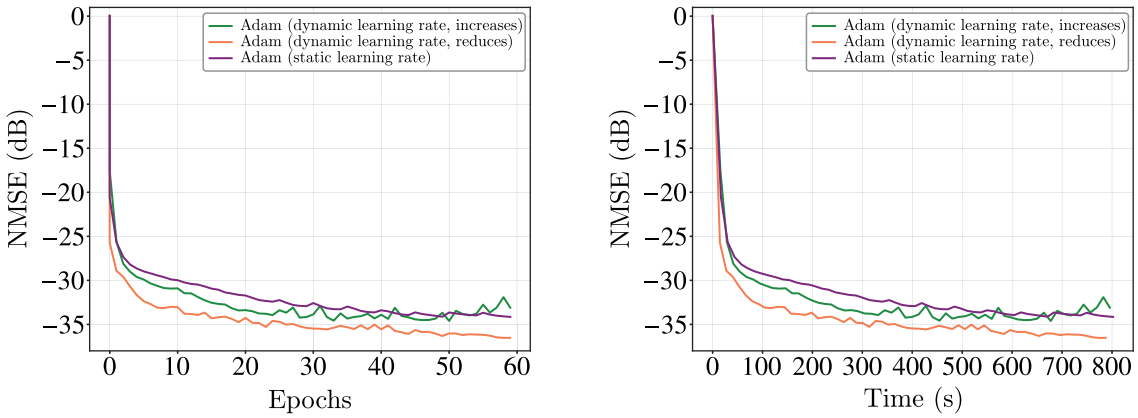
Another important hyperparameter of the optimization methods is its learning rate, i. e., some starting value of the step size. Further, we describe the results of our experiments on scheduling (changing) the value of the learning rate during the training. It is known that for some specific models it may be essential to use a proper learning rate scheduler to obtain a satisfactory accuracy [Xiong et al., 2020]. So, we tested some of the widely used schedulers in our case to check if it affects convergence significantly.

The first group of schedulers consists of the simple monotonically-decaying ones, and here we found the most suitable option for the growth rate and configuration of the scheduler. We tested both the increasing and decreasing changing of the learning rate, and obtained the following results:

1. The most proper formula to change the learning rate is a linear one (bounded below): lr = = $\max\{10^{-2}–10^{-4} \cdot \text{epoch}, 6 \cdot 10^{-3}\}$. The effect of scheduling is nearly the same, independent of the law it follows if the lower and upper bounds for the learning rate are $6 \cdot 10^{-3}$ and 10^{-2}, respectively.

2. The learning rate increasing strategy leads to a destabilization of convergence (as expected).

3. The reference score achieved by the Adam without learning rate scheduling is NMSE = = −34,29 dB. For the properly scheduled learning rate this score becomes NMSE = −36,56 dB. Thus, the improvement is 7 %. The convergence curves of some tested options are presented in Fig. 9.



(a) The dependence of NMSE value (dB) on the number of training epochs

(b) The dependence of NMSE value (dB) on the training time (s)

Figure 9. The effect of the learning rate changing on the training of the DPD model. When decreasing, it leads to the much deeper convergence of Adam

The results summarized above are related to only the monotone schedulers. On the other hand, it becomes more popular to use some cyclic strategies in some cases to improve the properties of the obtained solution. For instance, [Izmailov et al., 2018] proposed the scheduler called SWA (Stochastic

Weight Averaging), changing the learning rate in a cyclic manner and averaging the points after the smaller steps. In many practical deep learning problems it allows one to achieve around 1 % improvement in accuracy even without a precise tuning of the scheduler. The behavior of the cyclic learning rate scheduling without averaging is also considered in [Smith, 2017]. We tested both SWA and simple cycling strategies to compare their efficiency on our problem.

4. The SWA scheduler (with one epoch of the fixed learning rate, period of learning rate decreasing equal to one epoch, and the same lower and upper bounds on the learning rate as in previous experiment) worsens convergence as compared to the reference score obtained by Adam without any scheduling.

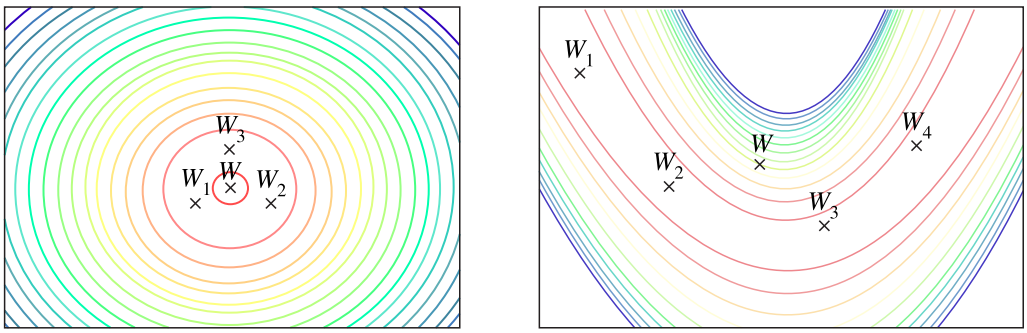
5. The same cyclic scheduling without averaging the points, on the contrary, allows a significant improvement to be obtained.

6. For the Adam optimization method, the improvement in depth given by the cyclic learning rate is 5 %. For the Adamax, it is 7 %. The results of the experiment are summarized in Table 2.

Table 2. Summary of the result of the training of the DPD model with a fixed and a cyclically changing learning rate

NMSE (dB)	Adam	Adamax
fixed	−33,30	−29,66
cyclic	−35,00	−31,86

7. We suppose that cyclic scheduling for our problem indeed behaves as it is described in [Garipov et al., 2018] (one of the main prerequisite papers for SWA). But averaging itself fails and it is not asymptotically equivalent to the FGE algorithm from [Garipov et al., 2018] in fact. It may be because the structure of local minima in the problem considered is not so good as in common deep learning tasks, i. e., instead of the situation shown in Fig. 10a we have something like Fig. 10b, so we see the loss in score after averaging.



(a) The behavior of SWA near the local minimum in many deep learning problems. The picture is similar to that from [Izmailov et al., 2018]

(b) The probable behavior of SWA in the case of a more complicated structured local minimum

Figure 10. The two probable cases of SWA [Izmailov et al., 2018] operation. We denote the result of averaging intermediate points w_1, w_2, w_3 by w. The left figure demonstrates a simple task for SWA, the right one may provide an explanation for the loss of SWA we observed

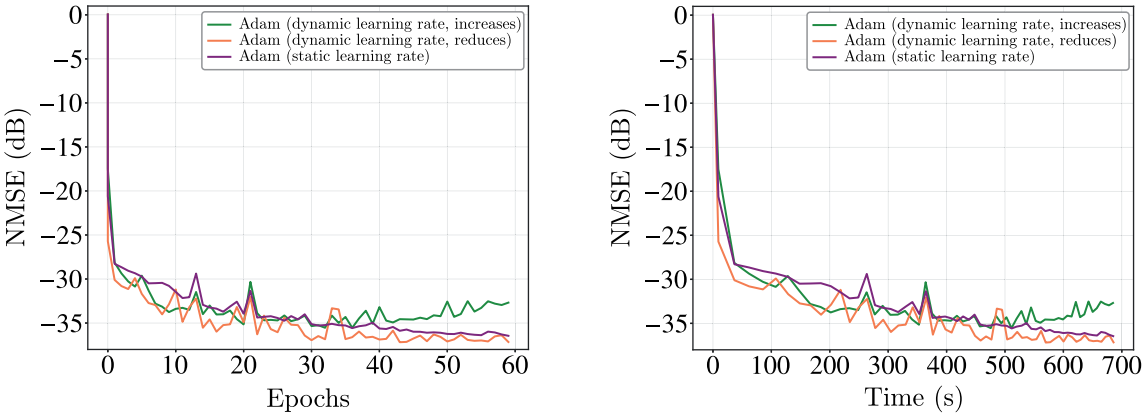
Finally, it is reasonable to assess the learning rate scheduling techniques we considered in conjunction with the previously described batch size changing, and check if we can combine them to obtain the better scores. We considered the testing of only a paired technique of increasing the batch

size and of monotonically decreasing the learning rate, since these two are efficient and simple enough to be combined without additional synchronizations.

8. Increasing the batch size with a decreasing learning rate is efficient, but the effect of the learning rate here is weaker than without batch size changing: it is only a 0,5 %, see Fig. 11. This is near the scale of score oscillations from epoch to epoch, so this improvement may be unstable.

9. For this compound regime the resulting score is NMSE = −35,46 dB. The reference value for this experiment is when the batch size increases, but the learning rate is fixed, it is NMSE = = −35,27 dB.

10. Increasing the learning rate still leads to a divergence, but this effect becomes more pronounced when the batch size increases.



(a) The dependence of NMSE value (dB) on the number of training epochs

(b) The dependence of NMSE value (dB) on the training time (s)

Figure 11. The effect of both learning rate changing and the increase in the batch size on the training of the DPD model. When the learning rate decreases, it leads to a little deeper convergence of Adam, while if it increases, it leads to a significant destabilization

These results are somewhat unexpected. One can think that if the batch size becomes bigger, the stability of the method becomes better, and at least with decreasing learning rate (which is very careful heuristic) it should have manifested itself in the improvement of scores, but it did not. Anyway, we can say that combining of batch size changing with the learning rate scheduling is not very efficient, so in any particular case it will be necessary to make a choice between big improvement in time and good improvement in depth. As mentioned above, the working time is minor metric for signal processing tasks. Nevertheless, it would be interesting to know that in online regime changing the batch size is just more efficient with respect to the depth of convergence than both of the learning rate scheduling policies.

Regularization

The last of the non-optimizational modifications considered in the course of our study is a regularization. It is very common to add the regularization term in the problem to control the properties of the solution, but the regularization may also be a good heuristic to improve the convergence, due to its impact on the (local) conditionality of the problem. This practical property of regularization seems to be underexamined and is bypassed in many new research papers on the application of optimization methods to learning problems.

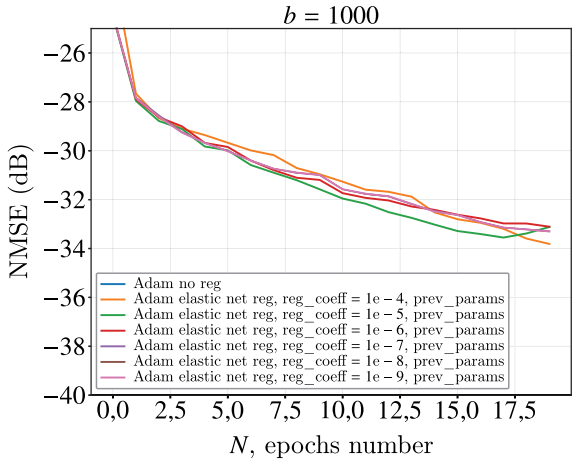
In out experiment, we considered only some widely known regularization terms, namely, ℓ_1 (lasso), ℓ_2 (ridge) and $\ell_1 + \ell_2$ (elastic net). To check their impact on convergence, we tested the effect of different optimization methods listed in Table 4, on the corresponding regularized problems. Below are the results it gave:

1. Unlike most of the results presented in the previous sections, the impact of regularization greatly depends on the chosen optimization method. So, Adam and Adamax are sensitive to it, while RAdam [Liu et al., 2019] is totally not (see Table 3).

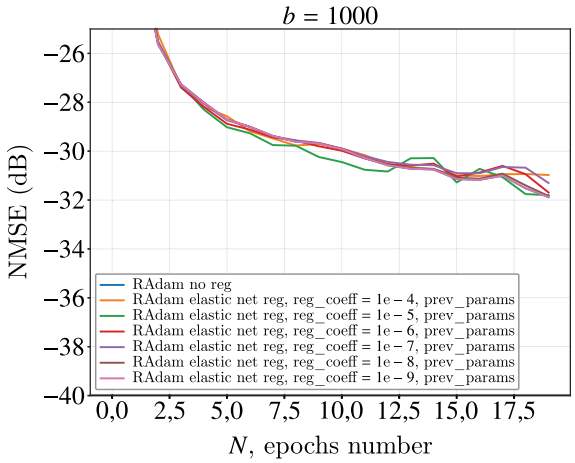
2. On the other hand, one can conclude from the left part of Table 3 that the standard regularization (one like $\lambda\|\theta\|_2^2$, we name it zero-centered; we set $\lambda = 10^{-4}$) does not affect the convergence. We present the minimum-aggregated values of NMSE, so we hide some probable destabilization effects here — they can appear in RAdam, but not in Adam or Adamax (see Fig. 12).

Table 3. Summary of the results of the training of the DPD model with different regularization terms

method	reference	min NMSE, zero-centered			min NMSE, prox-centered		
		ℓ_1	ℓ_2	$\ell_1 + \ell_2$	ℓ_1	ℓ_2	$\ell_1 + \ell_2$
Adam	−33,30	−33,30	−33,30	−33,30	−33,99	−33,31	−33,81
RAdam	−31,88	−31,88	−31,88	−31,88	−31,88	−31,88	−31,88



(a) The dependence of NMSE value (dB) on the number of training epochs for Adam



(b) The dependence of NMSE value (dB) on the number of training epochs for RAdam

Figure 12. The effect of prox-centered regularization on the training of the DPD model

The point is that it is more efficient here is to use more complicated regularization with moving center (one like $\lambda\|\theta - \overline{\theta}\|_2^2$, we name it prox-centered). We tested the version with a period of updates equal to one epoch and $\overline{\theta}$ set to the point obtained after the last iteration of the method. This configuration resembles the scheme of an inexact proximal gradient method a little.

3. So, this approach is efficient and gives 2 % improvement to the Adam optimization method. The results are summarized in Table 3.

4. It is unusual that the best result is obtained with ℓ_1-regularization (and also with elastic net, but the effect is smaller, so we can conclude that the ℓ_2 part of it only suppresses the effect of the ℓ_1 part).

Quasi-online learning framework

As mentioned above, the standard learning framework is not suitable for learning the DPD models due to some disagreements with the behaviour of practical devices. The point is that if we use some optimization methods, for example, Adam and Adamax, to fit the parameters of the real technical prototype, and then test it on some real life signal, we will find that Adamax (significantly) outperforms Adam, while our experiments in the previous section predict the opposite. The reason is that the operation of the technical prototype is different from the simulating scheme used before, due to the specifics of the signal processing task. In fact, the DPD model retrains periodically to be able to work with the new incoming signal. Unlike machine learning, the data in this case is updated with a big frequency (80 MHz or more) and, moreover, changes its properties, for example, due to the change of the interlocutor on the telephone line. A very simplified circuit diagram of DPD operation is presented in Fig. 13, and represents the looped process. So, if our results disagree with the experiment, we first should take into account this cyclicality as the most notable difference with the standard machine learning framework.

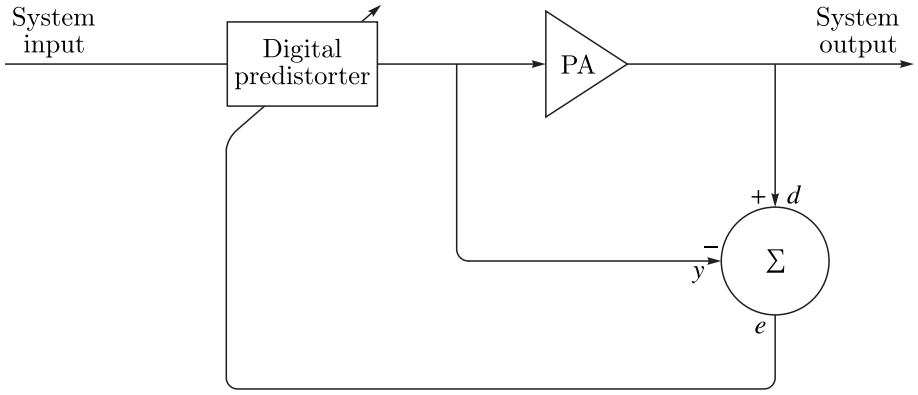


Figure 13. Schematic diagram of the operation and learning of the DPD device during signal processing

Further, we propose another framework to train/test DPD models, which includes the retraining procedures and more correct evaluation, and test some of the techniques proposed above under new conditions closer to real-life conditions.

New framework and comparison of algorithms

As was noted in the preamble of the section, the provided comparison of methods and testing of any practices under the previously postulated conditions have little in common with the behavior of these objects observed in reality. At least, it is difficult to find real analogies for training and test datasets, and yet the NMSE value we use is a function not only of the applied numerical method, but of these two parameters too. Of course, we are not able to avoid this functional dependence. We propose to replace these two variables with the most natural one — the whole given signal segment.

Let us turn to the operation of DPD device. Following the scheme from Fig. 13: DPD collects information about some small segment of signal, then it trains on it and is predicting distortion for some new small segment, while collecting information about it, then it repeats this procedure over and over again. The only thing we cannot wave away here is «over and over again», because we are limited by the given dataset, but we can exactly simulate all the other steps. Namely, let us split our dataset into some number of segments with the same length and train and test the model on the two consequent segments at every training *era*. This is shown in Fig. 14.

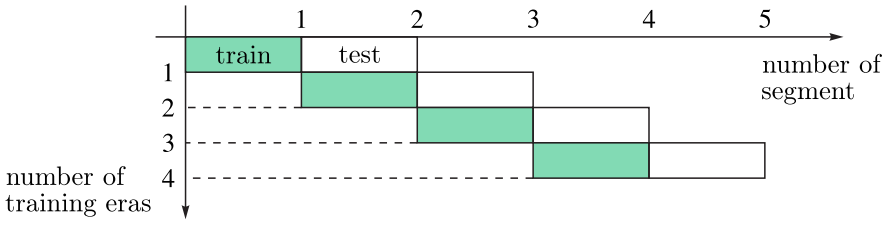


Figure 14. Description of an online learning framework: a diagram of data segments used at different moments in signal processing by the model

There are two options to assess the model in this framework. The most obvious one is the value of NMSE from the last era, i. e., evaluated on the last segment of the dataset. But this metric has two flaws: 1) it is evaluated only on the last segment of data, which may lead to approximation issues (for example, to an overestimation of error) if this segment is small; 2) it does not represent the performance of the method for the whole signal, and even if all segments except the last one are predicted absolutely incorrect, the score can be good enough. To deal with these problems, we also use the mean-aggregated version of NMSE:

$$\text{mean NMSE}(T, y, \overline{y}) := \frac{1}{T}\sum_{i=1}^{T} NMSE(y_i, \overline{y}_i) \text{ dB}.$$

It is in fact a very proper choice for assessment of the performance of online optimization methods, because it is usually used in their theoretical analysis [Hazan, Kale, 2014] (and is connected with the so-called regret metric).

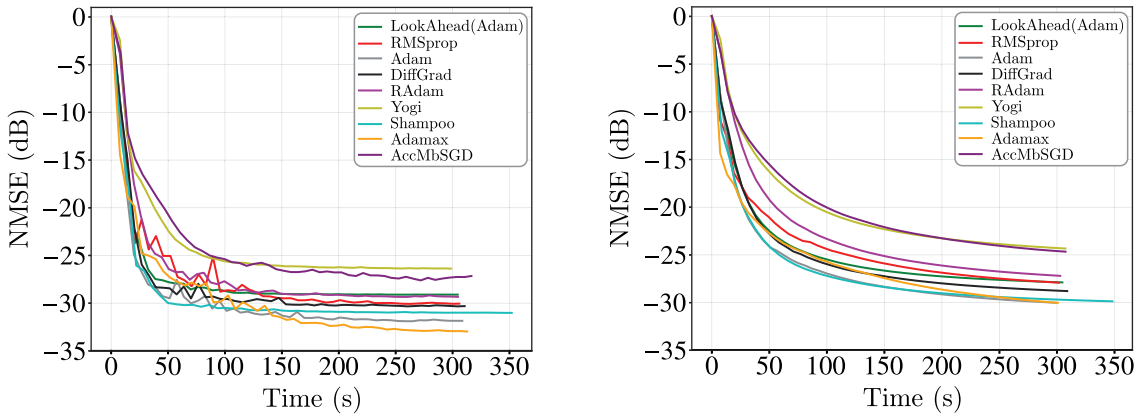
So, let us first reassess the optimization methods in a new setting without any additional modifications. In Table 4 we present the results for some of the methods with corresponding values of NMSE and mean NMSE. And here we see the key difference of the new approach with the previous one: the Adamax algorithm now is the best performing method and, by a large margin, it is 3 % better than Adam in NMSE, and also has a better rate of convergence at the late eras (see Fig. 15, *a*). The latter is the most important property of the optimization method in our task, because it determines the possible depth of solution given by this method. We see that despite the starting lag of Adamax it managed to overtake methods like Shampoo and Adam, which seems to be stuck in local minima or just to have declined in the convergence rate.

Table 4. Summary of the results of the training of the DPD model with different optimization methods in a new quasi-online setting

Method	NMSE (dB)	mean NMSE (dB)
Adamax	−33,04	−30,21
Adam	−31,94	−30,17
Shampoo	−31,06	−29,99
DiffGrad	−30,41	−28,76
RMSprop	−30,18	−27,96
LaAdam	−29,12	−27,90
RAdam	−29,35	−27,24
AccMbSGD	−27,18	−24,65
Yogi	−26,41	−24,29

For the other methods we see mostly the same results as those for the standard learning framework. Shampoo is still one of the best starter methods, and AccMbSGD does not demonstrate a good convergence rate. Figure 15, *b* shows the mean-aggregated version of the same plots, and we

can make similar conclusions from it, except that the leadership of Adamax here is not so pronounced, but it is just an effect of averaging.



(a) The dependence of NMSE value (dB) on the working time (s)

(b) The dependence of mean time NMSE value (dB) on the working time (s)

Figure 15. The convergence curves of different optimization methods in the quasi-online regime with evaluation after every era

Dynamic batch size

Now we should revisit the practices we proposed for the standard learning framework in view of the new one. Let us start by changing the batch size. Further, we consider only the most efficient policy we identified above (linear increase in the batch size), but at the same time, we test two modifications of this scheme, that are given by the more complicated structure of testing framework itself.

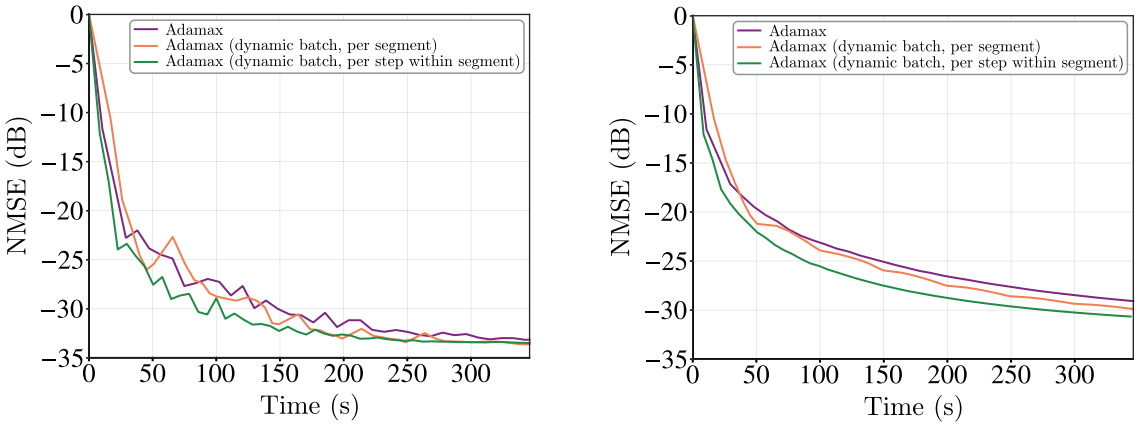
Namely, the first option is to change the batch size linearly depending on the current era (the number of the learning segment), and the second option is to link the batch size with the number of the epoch within every segment in every era. The choice of a policy here is in fact the trade-off between the depth of convergence in the «expected» score and the depth of convergence for every segment, respectively.

We tested both policies, and the result is that era changing (first option) and epoch changing (second option) are competitive depending on what metric we use. In the standard NMSE, era changing performs better, but after averaging the epoch changing becomes a better option. This means that it is more efficient to achieve a little improvement in convergence at every era (for every small piece of signal) instead of improving the outer loop convergence. It is quite an unexpected result.

The results of this experiment are summarized in Table 5. The use of the epoch changing batch size allows one to obtain 6 % improvement in mean NMSE metric for the considered setting. Figure 16 shows the convergence of Adamax with the different batch size changing policies in more detail for two metrics.

Table 5. Summary of the results of the training of the DPD model with a fixed and a changing batch size in the quasi-online regime

	NMSE (dB)	mean NMSE (dB)
fixed	−33,15	−28,96
era changing	−33,63	−29,85
epoch changing	−33,49	−30,66



(a) The dependence of NMSE value (dB) on the working time (s)

(b) The dependence of mean time NMSE value (dB) on the working time (s)

Figure 16. The effect of batch size changing on the training of the DPD model

Regularization

In the case of regularization, we do not need to take into account the difference between eras and epochs, so to test the performance of this modifications we just carried out one more experiment of learning the DPD model with the Adamax optimization method (it is the best one in the online setting) with ℓ_1-regularization (which was the best and the most stable option in the standard learning framework experiments). The result approves the positive impact of ℓ_1-regularization: compared to the reference value of mean NMSE = −30,72 dB, given by the model trained without any regularization, ℓ_1-regularization allows one to achieve mean NMSE = −30,81 dB, which is 0,3 % better. Unfortunately, this is near the scale of oscillations in convergence on the late iterations of the method, so we conclude that regularization gives no improvement in the online regime (but also does not impair the convergence depth).

We adhere to the explanation of the fact that the oscillations in the online regime are themselves really perceptible, and the effect of our modification is just suppressed by multiple switching between different data segments. Another reason is that after the last change of the segment, we again start to seek a new optimal point (corresponding to this segment) from some arbitrary point, and due to the small value of the regularization coefficient, we have a situation similar to that of the standard framework experiment, but within a small subset of data. In other words, regularization impacts convergence only on the last segment, and the impact is small because of smallness of this segment.

Cyclic learning rate

At last, we have reproduced the experiments with cyclic learning rate scheduling in the new online framework. Now we have applied this modification to the new best method, Adamax. We tested both options for the period of cycling: once in epoch or once in era, and chose the epoch — there is no qualitative difference between the results in these two configurations, but increasing the period leads to a decrease of this positive effect itself. So, we used learning rate cycling scheduling with a period equal to one epoch together with Adamax, for which the improvement was 1 % in mean NMSE, and with Adam, with a similar result — 1 % improvement in mean NMSE. The decrease in effect as compared to that for the standard learning framework is explained by the same reasons as presented in the previous section.

Conclusion

In the introduction, we posed two central questions of this research: which optimization method is the most suitable for training the DPD model, and what framework we need to use to simulate the DPD operation under conditions closer to real-life conditions and to match our simulation with the results observed in practice. In the course of our study, we carried out a large number of clarifying and detailing experiments to come to the two desired answers. In the first section, devoted to the standard learning framework, we summarize those that relate to the first question, presenting a comparison of many modern optimization methods and revealing the gist of the proposed modifications designed to improve their convergence. In the second section, we propose the new quasi-online learning framework and demonstrate that the ranking of the methods in our new experiments is mostly in line with that observed under close-to-real conditions, and also reassess some of the proposed modifications to approve their stability.

To express the achieved improvements in numbers: in the standard learning framework, the depth of convergence (the value of NMSE metric, which is logarithmic scaled itself) was improved in 7 % (thanks to steadily decreasing learning rate changing for Adam or cyclic scheduling for Adamax), while the maximum achieved improvement in the online regime is 6 % in mean NMSE (and given by the increase in the batch size at every epoch). In terms of working time, we have achieved a twofold improvement (with changing batch size), which preserves 3 % and 6 % improvement in NMSE and mean NMSE for the standard and online regime, respectively.

The interim results arising in the course of the study confirm that the considered DPD model is really specific and requires a separate study of the effectiveness of optimization methods as applied to its training: many approaches, common, for example, in the field of training neural networks, work here unexpectedly or do not work at all. The authors hopes that the proposed results will be useful for other researchers concerned with DPD models, and that the questions posed will stimulate a deeper study of related problems.

Acknowledgments

This research was supported and organized in cooperation with the educational center «Sirius» and the educational foundation «Talent and Success», Russia, under the Project science and technology program «Big Challenges». The assistance provided by the mentor of program, Anton Gusev, is greatly appreciated. The topic of this research is a part of a joint project of the Moscow Institute of Physics and Technology and the Huawei Russian Research Institute. We would like to thank Andrey Vorobyev, Eugeniy Yanitskiy and Anna Nikolaeva for openness to cooperation and interest in this study. We wish to extend out special thanks to Prof. Alexander Gasnikov for all the helpful discussions and his unwavering favor for this project. We also thank Liliana Baislanova and Mansur Zainullin for their participation in setting up some numerical experiments.

References

Devarakonda A., Naumov M., Garland M. Adabatch: Adaptive batch sizes for training deep neural networks // arXiv preprint. — 2017. — https://arxiv.org/pdf/1712.02029

Garipov T. et al. Loss surfaces, mode connectivity, and fast ensembling of dnns // Advances in neural information processing systems. — 2018. — Vol. 31.

Ghannouchi F. M., Hammi O., Helaoui M. Behavioral modeling and predistortion of wideband wireless transmitters. — John Wiley & Sons, 2015.

Gupta V., Koren T., Singer Y. Shampoo: Preconditioned stochastic tensor optimization // International Conference on Machine Learning. — PMLR, 2018. — P. 1842–1850.

Haykin S. S. Adaptive filter theory. — Pearson Education India, 2008.

Hazan E., Kale S. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization // The Journal of Machine Learning Research. — 2014. — Vol. 15, No. 1. — P. 2489–2512.

Izmailov P. et al. Averaging weights leads to wider optima and better generalization // arXiv preprint. — 2018. — https://arxiv.org/pdf/1803.05407

Kingma D. P., Ba J. Adam: A method for stochastic optimization // arXiv preprint. — 2014. — https://arxiv.org/pdf/1412.6980

Liu L. et al. On the variance of the adaptive learning rate and beyond // arXiv preprint. — 2019. — https://arxiv.org/pdf/1908.03265

Pasechnyuk D. et al. Non-convex optimization in digital pre-distortion of the signal // arXiv preprint. — 2021. — https://arxiv.org/pdf/2103.10552

Schreurs D. et al. RF power amplifier behavioral modeling. — New York, USA: Cambridge university press, 2008.

Smith L. N. Cyclical learning rates for training neural networks // 2017 IEEE winter conference on applications of computer vision (WACV). — IEEE, 2017. — P. 464–472.

Woodworth B., Srebro N. An Even More Optimal Stochastic Optimization Algorithm: Minibatching and Interpolation Learning // arXiv preprint. — 2021. — https://arxiv.org/pdf/2106.02720

Xiong R. et al. On layer normalization in the transformer architecture // International Conference on Machine Learning. — PMLR, 2020. — P. 10524–10533.

Zhang M. R. et al. Lookahead optimizer: k steps forward, 1 step back // arXiv preprint. — 2019. — https://arxiv.org/pdf/1907.08610

Zhao S. Y., Xie Y. P., Li W. J. Stagewise Enlargement of Batch Size for SGD-based Learning // arXiv preprint. — 2020. — https://arxiv.org/pdf/2002.11601