Advances in Artificial Intelligence and Machine Learning; Research 3 (1) 731-759 Received 19-01-2023; Accepted 22-02-2023; Published 01-03-2023

The Self-Learning AI Controller for Adaptive Power Beaming With Fiber-Array Laser Transmitter System

A.M. Vorontsov Artem.Vorontsov@kaspersky.com
Kaspersky Lab USA
500, Unicorn Park, Woburn, MA, 01801, USA

G.A. Filimonov gfilimonov1@udayton.edu
Intelligent Optics Laboratory
School of Engineering, University of Dayton
Dayton, Ohio, 45469, USA

Corresponding Author: A.M. Vorontsov

Abstract

In this study we consider adaptive power beaming with a fiber-array laser transmitter system in presence of atmospheric turbulence. For optimization of power transition through the atmosphere a fiber-array is traditionally controlled by stochastic parallel gradient descent (SPGD) algorithm where control feedback is provided via a radio frequency link by an optical-to-electrical power conversion sensor, attached to a cooperative target. The SPGD algorithm continuously and randomly perturbs voltages applied to fiber-array phase shifters and fiber tip positioners in order to maximize sensor signal, i.e. uses, the so-called, "blind" optimization principle.

By contrast to this approach a prospective artificially intelligent (AI) control systems for synthesis of optimal control can utilize various pupil- or target-plane data available for the analysis including wavefront sensor data, photo-voltaic array (PVA) data, other optical or atmospheric parameters, and potentially can eliminate well-known drawbacks of SPGD-based controllers. In this study an optimal control is synthesized by a deep neural network (DNN) using target-plane PVA sensor data as its input. A DNN training is occurred online in sync with control system operation and is performed by applying of small perturbations to DNN's outputs. This approach does not require initial DNN's pre-training as well as guarantees optimization of system performance in time. All theoretical results are verified by numerical experiments.

Keywords: Fiber-array, Power beaming, SPGD optimization, Reinforcement learning, Self-learning, AI controller

Citation: A.M. Vorontsov and G.A. Filimonov. The Self-Learning AI Controller for Adaptive Power Beaming With Fiber-Array Laser Transmitter System. Advances in Artificial Intelligence and Machine Learning. 2023;3(1):48.

1. INTRODUCTION

The interest in the development of fiber-array laser transmitter systems [1, 2], has been steadily growing over the past two decades due to its compact size and low cost of system components comparing with conventional systems having the same transmitting aperture size. The presence of a beam control system makes fiber-arrays capable for direct integration and digital implementation of exotic beam shaping [3], adaptive mitigation of the propagation-medium-induced phase aberrations [4], target tracking and beam pointing. All this allows one to consider fiber-array as a potential laser transmitter system for a wide class of optical applications – from directed energy to free-space optical communications [2, 4], additive manufacturing [5], and power beaming [6].

Wireless delivering energy to remote and hard-to-reach power consumers with laser transmitters is promising technology especially taking into account the explosive growth number of rechargeable portable devices and equipment. In recent power beaming experiments with single-aperture laser transmitters charging of remotely located mobile devices including drones [9], vehicles [7], and even cell phones [8], were performed where for optical-to-electrical power conversion photo-voltaic array (PVA) panels attached to a particular device were used. As far as all mentioned experiments occurred in lower Earth atmosphere authors especially reported significant dependence of laser energy transfer efficiency on atmospheric conditions along a beam propagation path. By this reason in [7], only 25%[1] of transmitted laser energy was delivered to a mini rover and in [8], for increasing of energy transfer efficiency authors optimized laser beam position trying to keep it near PVA center. Thus, in realistic conditions appropriate power beaming efficiency can be achieved using adaptively controlled laser systems only including fiber-arrays.

Consider the problem of optimal energy transfer through the atmosphere onto remote high-resolution PVA panel. Let for mitigation of the atmospheric-induced received power losses from a non-optimal match between photovoltaic conversion and the projected laser beam footprint, errors in laser beam pointing, turbulence-induced beam spread, wander and distortion controlled fiber-array laser transmitter system is used. Suppose that conformal laser beam outgoing from an array of optical collimators passes through the turbulent atmosphere and illuminates PVA (see FIGURE 1). The PVA performs optical-to-electrical power conversion by delivering the transferred energy to a power consumer and in addition has a radio-frequency (RF) link with multi-channel master oscillator/power amplifier (MOPA) fiber system. This link allows rapid (in comparison with turbulence changing time) transfer of PVA grid power values directly to the control system where this information can be used for optimization of phase shifts and fiber tip positions. The control efficiency in power beaming is traditionally measured using several well-known indicators (or metrics) including overall transferred energy magnitude and a parameter responsible for matching optimal power distribution on PVA.

The most common fiber-array optimization controller uses stochastic parallel gradient descent (SPDG) algorithm for syntheses of the control. Optimization of metric value by SPGD algorithm is continuously performed by generating and applying of small random perturbations to fiber-array actuators. If current perturbation increases (for metric maximization

[1] This estimation includes PVA transfer energy efficiency also.

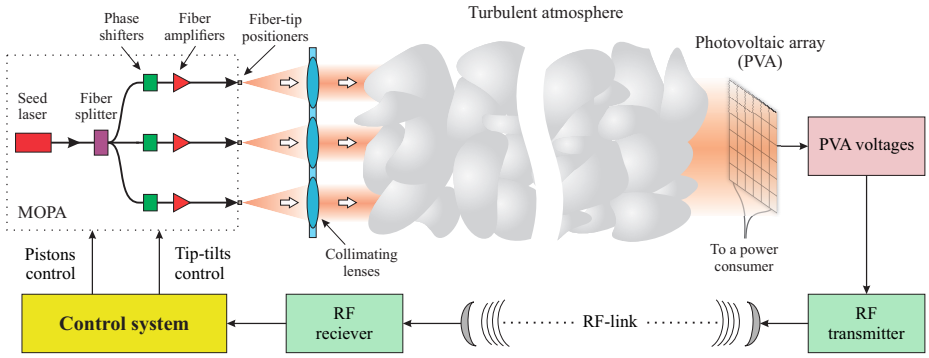

Figure 1: Notional schematics of the fiber-array transmitter system in the problem of power beaming through turbulent atmosphere.

problem) metric value, the optimization step is considered as completed, otherwise, depending on algorithm type, a new perturbation is generated or existing perturbation is applied with an opposite sign. Modern SPGD controllers used for fiber arrays coherent combining allow to perform up to 5×10^5 parallel optimization steps per second that provides desirable performance for a wide range of fiber-array configurations, propagation scenarios and atmospheric conditions. However this technique has several well-known drawbacks: (1) SPGD convergence is rapidly decreasing with increase of the number of subapertures above 50-100, (2) SPGD optimization algorithm is extremely sensitive to perturbation statistics and type of gain coefficient that in each particular case are chosen empirically, (3) SPGD realizes, so-called, "blind" optimization strategy and ignores any additional information available for the analysis and potentially useful for synthesis of the control, for example, PVA power distribution as in the considered case.

One of the most promising way to avoid aforementioned drawbacks is an enhancing of SPGD optimization algorithm using state-of-the-art deep learning (DL) methodology. The DL paradigm supposes to utilize deep neural networks (DNN) of various types and topologies for extraction and analyzing of information about interaction of the control system with environment in order to synthesize optimal control. Using of DNN in a control system supposes to have some training mechanism and here two baseline approaches can be proposed: (a) offline training when the controller's DNN is specially pretrained in offline regime and only than is applied for real-time control purposes and (b) self-learning when randomly initiated DNN is continuously interacting with changing environment is trained on-the-fly using performance metrics as stimulus for learning and observed environment characteristics as the input [10].

There are numerous papers (see, for example, [11, 12], and literature in these papers) dedicated to coherent beam combining using DNN with offline training strategy. Methods of DNN utilization for syntheses of optimal control are significantly varied for different authors and include direct generation of control by DNN using target-plane measurements [13], generation by DNN of some initial pupil-plane phase distributions [14], and more sophisticated cascaded schemes where DNN is operated in pair with SPGD [15]. However, this methodology has also several important drawbacks: (1) offline training technique supposes to manually collect or simulate huge training datasets that should cover all potential system application scenarios, (2) one should guarantee that optical system will always operate in the same environmental conditions as represented in this training set, so that any extension

of condition ranges or any system modification automatically requires to supplement (in the best case) or completely rebuild (in the worst case) of training set with the following DNN retraining. In other words, the offline learning scheme does not support any system adaptation to any changes and primary suitable for the systems which are interacted with the statistically constant environments.

In opposite, reinforcement learning (RL) approach [16], initially is designed for operation in unknown and changing environment. The major idea of this methodology is to place completely untrained intellectual agent (controller, in our terminology) in some environment, to provide their continuous mutual interaction and to define some reward function that should be maximized during these interactions. Here, the reward function plays a role of stimulus to agent's learning and its adaptation to environment changes, stimulus to forget unusual or outdated information about the environment or about interaction with the environment, stimulus to follow optimal trajectories. The agent-environment interaction in RL is considered in terms of "action-state" space where agent's action is typically represented in the form of multidimensional control vector and for describing of environment state available and currently observed information (f.e., instantaneous sensor measurements, snapshot images, etc.) is used.

The problem of fiber-array coherent beam combining can be classified in RL paradigm as "multidimensional continuous action space" problem that commonly considered using deep Q-learning [17], approach or more precisely deep deterministic policy gradient (DDPG) algorithm [18]. In this approach two separate DNN are considered – one a-DNN represents intellectual agent as well as other one q-DNN is responsible for simulation of environment response. Traditionally this response is chosen in the form of Bellman's Q-function [19], so that q-DNN will approximate it during agent-environment interaction. The training process is synchronized for both DNNs – an agent performs some (initially random) actions, explores environment's reaction and takes reward thereby training q-DNN that provides local approximation of environment response and allows to predict this response for future local actions. Built approximation of Q function allows to take (symbolic) gradient over q-DNN's inputs, then to use this gradient for correction of a-DNN trainable weights in accordance with policy gradient theorem and hence to improve agent future actions.

The proposed DDPG algorithm can be obviously applied to considered power beaming problem if [20] , for example, as the reward function one takes aforementioned system performance metric, as intellectual agent consider beam controller and as environment state – target-plane PVA intensity distribution and instantaneous metric magnitude. However, this straightforward approach has one important drawback – in this case one factually needs to build DNN approximation of metric response to all current and following controller's actions (i.e. Q-function for this case). For optical systems operated in realistic conditions performance metric depends on numerous factors, metric reaction on piston/tip-tilt control is non-linear and can be changed in time, number of control channels is big and these channels are strongly coupled. In these conditions there is no chance to build robust, accurate and static approximation of Q-function that will automatically lead to poor training of controller's DNN and completely avoid all advantages given by RL.

In this paper we introduce SPGD-based training procedure of DNN controller, where instead of consideration and differentiation of Bellman's Q-function we use SPGD-type estimation

of environment's response gradient. This technique is especially suitable for rapidly changing non-linear environments and multidimensional control space specific for the problem of energy transferring through the atmosphere. The idea is to continuously pass all available incoming information about environment state (PVA intensity distribution, metric) and controller's action (actuator voltages) to DNN input in order to synthesize optimal control in these particular conditions and time moment. At the same time DNN outputs are continuously perturbed using SPGD-type procedure that provides self-learning of DNN as well as guarantee minimization of performance metric in time. In this paper this type of control will be called "active AI control" meaning that here simultaneously two independent processes are taking place – "active" SPGD-type metric optimization with parallel DNN training and "passive" DNN inference with synthesis of the control.

In Section 2 the problem statement, theoretical background of SPGD-based optimal control and active AI control will be given. In Section 3 implementation details for the proposed in Section 2 approach will be stated. In Section 4 training capabilities and performance of metric optimization for the proposed AI control system is demonstrated via numerical experiments.

2. BASIC CONSIDERATIONS

In this section we introduce the power beaming problem as well as basic principles of SPGD and active AI control.

2.1 The Power Beaming Problem

Consider optical wave propagation along (parallel to) the axis Oz of Cartesian system $Oxyz$, assuming that the fiber-array transmitter is located at the coordinate origin O and PVA panel – at the point $(0, 0, L)$, where $L > 0$ is propagation distance, see FIGURE 2 on the right. Let $\boldsymbol{r} = (x, y, z)$ be a vector in $Oxyz$, $\boldsymbol{\rho} = (x, y)$ is a plane coordinates and $t \geq 0$ denotes time. Propagation of linear polarized monochromatic optical waves with complex amplitude $U(\boldsymbol{r}, t) = U(\boldsymbol{\rho}, z, t)$ through an optically inhomogeneous medium is commonly described by the following parabolic equation [21]:

$$2ik\frac{\partial U(\boldsymbol{\rho}, z, t)}{\partial z} + \nabla_{\perp}^2 U(\boldsymbol{\rho}, z, t) + 2k^2 n(\boldsymbol{\rho}, z, t) U(\boldsymbol{\rho}, z, t) = 0, \tag{1}$$

where k is the wave number, $\nabla_{\perp}^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$ and $n(\boldsymbol{\rho}, z, t) = n(\boldsymbol{r}, t)$ is a random function corresponding to the refractive index fluctuations having zero meanvalue $\langle n(\boldsymbol{r}, t)\rangle = 0$. The notation $\langle\cdot\rangle$ is used to describe statistical averaging over the ensemble of refractive index realizations. The boundary conditions for complex amplitude U at the plane $z = 0$ are defined as:

$$U(\boldsymbol{\rho}, z = 0, t) = U_0(\boldsymbol{\rho}, t),$$

where $U_0(\boldsymbol{\rho}, t)$ is the optical field complex amplitude at the transmitter plane.

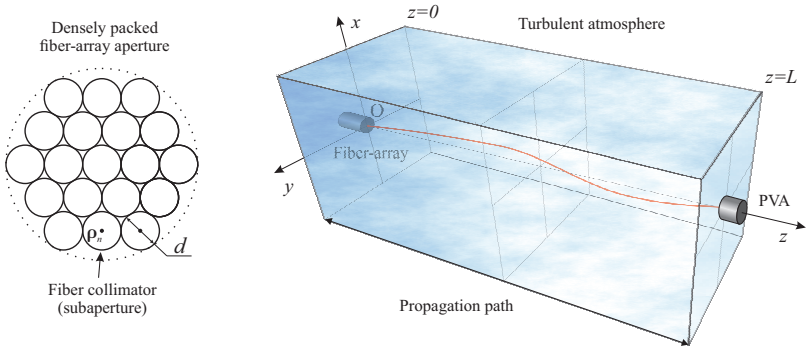

Figure 2: Fiber-array aperture with $N_{\text{sa}} = 19$ subapertures (left) and geometry of optical field propagation (right).

Assume that the fiber-array transmitter system has N_{sa} hexagonal subapertures (see FIGURE 2, left) and each particular subaperture is described by a stepwise function $H_n(\boldsymbol{\rho}) = H(\boldsymbol{\rho} - \boldsymbol{\rho}_n)$, $n \in \{1, ..., N_{\text{sa}}\}$, where $\boldsymbol{\rho}_n = (x_n, y_n)$ is coordinate of subaperture center and

$$H(\boldsymbol{\rho}) = \begin{cases} 1, & |\boldsymbol{\rho}| \le d/2, \\ 0, & \text{otherwice}, \end{cases}$$

where $d > 0$ is subaperture diameter. For the complex amplitude of optical field at the fiber-array output (pupil) plane $\{z = 0\}$ one has:

$$U_0(\boldsymbol{\rho}, t) = A_0 \sum_{n=1}^{N_{\text{sa}}} H_n(\boldsymbol{\rho})\, e^{-|\boldsymbol{\rho}-\boldsymbol{\rho}_n|^2/a_0^2}\, e^{ik\varphi_n(\boldsymbol{\rho},t)}, \tag{2}$$

where $A_0 \ge 0$ is the constant dependent on the power transmitted through the fiber-array and $\varphi_n(\boldsymbol{\rho}, t) = \mathbf{s}_n(t) \cdot (\boldsymbol{\rho} - \boldsymbol{\rho}_n) + c_n(t)$, $n \in \{1, ..., N_{\text{sa}}\}$ is the linear over $\boldsymbol{\rho}$ function representing phase component of the outgoing field in the n-th subaperture at time $t \ge 0$. Here $\mathbf{s}_n(t) = (s_{n,x}(t), s_{n,y}(t)) \in \mathbb{R}^2$ and $c_n(t) \in \mathbb{R}$ are, respectively, tip-tilt and piston components of the n-th beamlet and dot denotes scalar product of two vectors. The beamlet radius a_0 corresponds to e^{-1} fall-off in intensity.

Adaptive beam shaping and compensation of turbulence-induced aberrations can be performed using control of either solely piston $\{c_n(t)\}_{n=1}^{N_{\text{sa}}}$ or piston and tip-tilt $\{\mathbf{s}_n(t)\}_{n=1}^{N_{\text{sa}}}$ phase components. For future considerations denote as K the number of all control parameters so that $K = N_{\text{sa}}$ for single piston control (then $s_{n,x}(t)$ and $s_{n,y}(t)$ are constants for any $n \in \{1, ..., N_{\text{sa}}\}$ and $t \ge 0$) or $K = 3N_{\text{sa}}$ for both piston and tip-tilt control.

2.2 Performance Metric for Power Beaming Problem

Now suppose that considered fiber-array transmitter is equipped by a control system operating with aforementioned phase components. The control performance is traditionally estimated using several well-known metrics suitable for particular transfer energy task. For

example, overall fiber-array transmitted power is accounted using integral power-in-the-bucket (PIB) metric, for a beam shaping problem this metric should be accompanied with the term, accounting power distribution on PVA, for directed energy applications Strehl ratio dedicated to power peak-to-valley measurement is commonly used. The PIB and Strehl ratio can be measured directly using photo-voltaic sensors or computed using target-plane intensity distribution $I(\boldsymbol{\rho}, z = L, t) = |U(\boldsymbol{\rho}, z = L, t)|^2$ in case if PVA is attached to the target.

In this study we will suppose that at the target-plane $z = L$ there is a high-resolution square PVA centered at the point $(0, 0, L)$ with sides having the size $D > 0$ and parallel to x and y axes. As the performance metric we will consider, so-called, smooth Strehl ratio:

$$J_t = J(t) = \frac{1}{J_{\text{vac}}} \int I(\boldsymbol{\rho}, z = L, t)\, e^{-|\boldsymbol{\rho}|^2/\beta^2} d^2\boldsymbol{\rho}, \tag{3}$$

where $\beta > 0$ is smoothing parameter and J_{vac} is normalization factor for providing the condition $J_t \leq 1$ for any $t \geq 0$. This condition will be guaranteed if we consider free-space propagation of the initial complex field U_0 with $n(\boldsymbol{\rho}, z, t) \equiv 0$ in (1), take corresponding target-plane intensity distribution $I_{\text{vac}}(\boldsymbol{\rho}, z = L, t)$ and compute J_{vac} as:

$$J_{\text{vac}} = \max \left\{ \int I_{\text{vac}}(\boldsymbol{\rho}, z = L, t)\, e^{-|\boldsymbol{\rho}|^2/\beta^2} d^2\boldsymbol{\rho} \right\}, \tag{4}$$

where maximum is taken over all K control parameters. Note that all target-plane intensity distributions are measured on PVA (see FIGURE 1) so that integration in (3) and (4) is also limited to PVA area only.

2.3 Basic Principles of SPGD Control

Denote the entire set of control variables as $\mathbf{u} = \left(u^1, ..., u^K\right)$. Consider general case when performance metric depends on a vector of some pupil- or target-plane field characteristics $\mathbf{I}_t = \mathbf{I}_t(\boldsymbol{\rho}; \mathbf{u}) = \left(I^1(\boldsymbol{\rho}, t; \mathbf{u}), ..., I^M(\boldsymbol{\rho}, t; \mathbf{u})\right)$, $M \geq 1$, where each particular characteristic, in turn, depends on control parameters as well as can be considered as spatial and time dependent function. This notation covers metric dependence on one- or two-dimensional field characteristics such as time series, intensity and phase distributions. For example, for considered above smooth Strehl ratio one has $M = 1$ and $I^1(\boldsymbol{\rho}, t; \mathbf{u}) = I(\boldsymbol{\rho}, z = L, t)$ is time-dependent target-plane intensity image and for conventional Strehl ratio $I^1(t; \mathbf{u}) = I(\boldsymbol{\rho} = \mathbf{0}, z = L, t)$ is time series.

Then, for the performance metric one has:

$$J_t = J_t(\mathbf{u}) = J(\mathbf{I}_t(\boldsymbol{\rho}; \mathbf{u}))$$

and the problem of synthesis of optimal control can be formulated as follows:

$$J_t(\mathbf{u}) \to \max_{\mathbf{u}}. \tag{5}$$

The optimization problem (5) can be numerically solved using well-known gradient descent optimization algorithm. Let an optimal control trajectory be $\mathbf{u}_t = \mathbf{u}(t) = \left(u_t^1, ..., u_t^K\right) = \left(u^1(t), ..., u^K(t)\right)$. Taking as the time increment the unity step the iteration process will have the form:

$$u_{t+1}^k = u_t^k + \gamma_t \frac{\partial J_t}{\partial u^k} du^k, \;\; k \in \{1, ..., K\}, \tag{6}$$

where $\gamma_t = \gamma(J_t, t)$ is time-dependent gain coefficient and $\left\{du^k\right\}_{k=1}^K$ are positive scaling factors corresponding to different control sensitivities. In case when metric gradient over control parameters can be derived analytically or numerically the algorithm (6) (or its numerous variations), as the rule, provides rapid and robust computation of optimal trajectory's approximation. However, for real-world physical systems when $J_t(\mathbf{u})$ is measured in real-time regime and computation of metric gradient is performed via applying component-wise perturbations to control variables the iteration process (6) has important drawback. The point is that with increasing of the number of control parameters and under assumption of sufficiently fast changing of physical conditions (f.e., atmospheric turbulence is changing in approximately 1 ms) the calculation of gradient in (6) will be lag behind environmental changes and as the result the gradient approximation will be inaccurate.

The SPGD algorithm [22, 23], is dedicated to avoid this problem. The major idea of SPGD optimization is in fast estimation of metric gradient using simultaneous perturbations of all control parameters. Let $\mathbf{u}_{t+1} = \mathbf{u}_t + \delta\mathbf{u}_t$, where $\delta\mathbf{u}_t = \left(\delta u^1, ..., \delta u^K\right)$ is a small increment that can be applied to controls in each time step. Then $J_{t+1} = J\left(\mathbf{I}_{t+1}(\boldsymbol{\rho}; \mathbf{u}_{t+1})\right) = J_t + \delta J_t$ and

$$\delta J_t = J_{t+1} - J_t \approx \sum_{k=1}^{K} \frac{\partial J_t}{\partial u^k} \delta u_t^k + \sum_{m=1}^{M} \frac{\partial J_t}{\partial I^m} \frac{\partial I_t^m}{\partial t}. \tag{7}$$

Under assumption of relatively small changing of environment characteristics $\mathbf{I}_t$ per one time step in comparison with their reaction on the control perturbation, i.e. if:

$$\left|\frac{\partial I_t^m}{\partial t}\right| \ll \left|\sum_{k=1}^{K} \delta u_t^k \frac{\partial I_t^m}{\partial u^k}\right|, \;\; m \in \{1, ..., M\},$$

the second term in (7) can be omitted so that one has:

$$\delta J_t \approx \sum_{k=1}^{K} \frac{\partial J_t}{\partial u^k} \delta u_t^k.$$

Suppose that applied perturbations have stochastic nature and $\delta\mathbf{u}_t$ is a random vector with the following statistical properties:

$$\begin{gathered}\left\langle \delta u_t^k \right\rangle = 0, \;\; k \in \{1,...,K\}, t \geq 0, \\ \left\langle \delta u_t^k \delta u_t^l \right\rangle = B_t^{kl}, \;\; k,l \in \{1,...,K\}, t \geq 0, \\ \left\langle \delta u_t^k \delta u_{t'}^l \right\rangle = 0, \;\; \text{for any} \;\; k,l \in \{1,...,K\} \;\; \text{and} \;\; t \neq t', \, t,t' \geq 0,\end{gathered} \tag{8}$$

where $\langle\cdot\rangle$, as before, denotes statistical averaging under ensemble of realizations, the diagonal correlation matrix $B_t = \underset{k=1,...,K}{\text{diag}} \left\{ \left(\sigma_t^k\right)^2 \right\}$ has the size $K \times K$ and $\sigma_t^k > 0$, $k \in \{1,...,K\}$ are standard deviations which will be specified below. Then, taking into account (8), one has for $l \in \{1,...,K\}$:

$$\left\langle \delta J_t \delta u_t^l \right\rangle \approx \left\langle \sum_{k=1}^{K} \frac{\partial J_t}{\partial u^k} \delta u_t^k \delta u_t^l \right\rangle = \frac{\partial J_t}{\partial u^l} \left\langle \left(\delta u_t^l\right)^2 \right\rangle = \frac{\partial J_t}{\partial u^l} \left(\sigma_t^l\right)^2.$$

Changing in the last formula statistical averaging $\langle\cdot\rangle$ by averaging over time and taking into account aforementioned assumption about contribution of control and environmental factors one can rewrite the iteration process (6) in the form:

$$u_{t+1}^k = u_t^k + \frac{\gamma_t}{\sigma_t^k} \delta J_t \delta u_t^k, \;\; k \in \{1,...,K\}. \tag{9}$$

The formula (9) represents canonical SPGD optimization (maximization) algorithm which implementation will be discussed in details in Section 3.1.

The SPGD control algorithm provides appropriate optimization for a wide range of fiber-array configurations, performance metrics and atmospheric conditions. The computation of metric gradient estimation by SPGD algorithm is simple and can be performed extremely fast so that novel SPGD controllers can achieve the performance up to 5×10^5 iterations per second. All these facts motivate us try to adapt SPGD gradient estimation not solely for metric optimization but for DNN training purposes too.

2.4 Basic Principles of Active AI control

From the consideration above it is clear that SPGD controller uses “blind” optimization principle when metric maximization is performed using solely metric perturbations and any additional and potentially useful information for solving the optimization problem is ignored. In opposite to this approach an AI controller have to be designed in such a way that it could collect and analyze all available information about physical process in order to synthesize optimal control potentially faster and more robust than SPGD.

Recall that in our considerations this additional information is represented via the vector $\mathbf{I}_t = \mathbf{I}_t(\boldsymbol{\rho}; \mathbf{u})$ of observed physical characteristics as well as their historical values. Hence, in order to account this information for the synthesis of optimal control the vector $\mathbf{u}_t$ now should be represented as a function dependent on instantaneous and retrospective values of $\mathbf{I}_\tau$ and J_τ taken for the time $\tau \leq t$ and $\mathbf{u}_\tau$ taken for $\tau < t$. On the other hand, in order to make the controller capable for learning, i.e. capable for analyzing and correct utilizing the incoming information one has to include into this functional dependence a set of trainable parameters for their adjusting during controller operation. So, AI controller should have the form:

$$\mathbf{u}_t = \mathbf{U}\left(\mathbf{I}_{\tau \leq t}, \mathbf{u}_{\tau < t}, J_{\tau \leq t}; \boldsymbol{\alpha}\right),$$

where $\mathbf{U}_t(\boldsymbol{\alpha}) = \mathbf{U}\left(\mathbf{I}_{\tau \leq t}, \mathbf{u}_{\tau < t}, J_{\tau \leq t}; \boldsymbol{\alpha}\right) = \left\{U^k\left(\mathbf{I}_{\tau \leq t}, \mathbf{u}_{\tau < t}, J_{\tau \leq t}; \boldsymbol{\alpha}\right)\right\}_{k=1}^K$, is some unknown vector function and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_P)$, $P \geq 1$ is the vector of optimization (trainable) parameters. For the future considerations denote $U_t^k = U_t^k(\boldsymbol{\alpha}) = U^k\left(\mathbf{I}_{\tau \leq t}, \mathbf{u}_{\tau < t}, J_{\tau \leq t}; \boldsymbol{\alpha}\right)$. Then, the optimization problem (5) can be rewritten as:

$$\begin{gathered} J_t = J_t(\boldsymbol{\alpha}) = J\left(\mathbf{I}_t\left(\boldsymbol{\rho}; \mathbf{U}_t(\boldsymbol{\alpha})\right)\right), \\ J_t(\boldsymbol{\alpha}) \to \max_{\boldsymbol{\alpha}}. \end{gathered} \tag{10}$$

As before, the gradient descent optimization algorithm immediately gives us the following iteration process for the parameters $\boldsymbol{\alpha}$:

$$\alpha_{t+1}^p = \alpha_t^p + \gamma_t \sum_{k=1}^K \frac{\partial J_t}{\partial u^k} \frac{\partial U_t^k}{\partial \alpha^p} d\alpha^p, \;\; p \in \{1, ..., P\},$$

where $\gamma_t = \gamma(J_t, t)$ is the gain coefficient and $\{d\alpha^p\}_{p=1}^P$ are positive scaling factors. The estimation of gradients $\left\{\partial J_t / \partial u^k\right\}_{k=1}^K$ in the last formula can be performed as before using random perturbations. Then:

$$\alpha_{t+1}^p = \alpha_t^p + \gamma_t \delta J_t \sum_{k=1}^K \frac{\delta u_t^k}{\sigma_t^k} \frac{\partial U_t^k}{\partial \alpha^p} d\alpha^p, \;\; p \in \{1, ..., P\}. \tag{11}$$

In the most frequently used case one has $\sigma_t^k = \sigma_t$ for any $k \in \{1, ..., K\}$ and $d\alpha^p = d\alpha$ for any $p \in \{1, ..., P\}$ so that the formula (11) can be rewritten as:

$$\alpha_{t+1}^p = \alpha_t^p + \frac{c_t}{\sigma_t} \delta J_t \sum_{k=1}^K \delta u_t^k \frac{\partial U_t^k}{\partial \alpha^p}, \;\; p \in \{1, ..., P\},$$

where $c_t = d\alpha \gamma_t$ is, so-called, learning rate.

Now suppose that $\mathbf{U}\left(\mathbf{I}_{\tau\leq t}, \mathbf{u}_{\tau\leq t}, J_{\tau<t}; \boldsymbol{\alpha}\right)$ is taken in the form of DNN with trainable parameters $\boldsymbol{\alpha}$. In this case formula (11) factually represents the analogue of well-known back-propagation algorithm for training of DNN where instead of metric gradient its estimation via random perturbations is taken. The DNN gradients $\left\{\partial U_t^k/\partial\alpha^p\right\}_{k=1,p=1}^{K,P}$ can be computed directly using any symbolic derivative framework for neural networks (f.e., Tensorflow [31]).

Formula (11) represents the training rule for DNN $\mathbf{U}_t\left(\boldsymbol{\alpha}\right)$ and at the same time provides SPGD-type optimization of the performance metric $J_t\left(\boldsymbol{\alpha}\right)$. In the next sections we will give important implementation details of DNN training using formula (11) as well as conduct numerical analysis of the proposed training mechanism.

3. IMPLEMENTATION DETAILS

Application of the proposed above methodology to power beaming problem requires specifying of the mathematical formulations. For a future considerations let us suppose that the vector $\mathbf{I}_t$ of target-plane field characteristics consists of only target-plane intensity distribution so that $M = 1$ and $\mathbf{I}_t = \mathbf{I}_t\left(\boldsymbol{\rho}; \mathbf{u}\right) = \left\{I\left(\boldsymbol{\rho}, z = L, t\right)\right\}$. In addition, we will suppose that the metric values are normalized to the unity interval so that $J_t \in [0, 1]$.

3.1 The SPGD Controller

Practical implementation of SPGD algorithm based on formula (9) shows that choosing of appropriate perturbation strength (variances $\sigma_t^k,\ k \in \{1, ..., K\}$) and gain coefficient γ_t are extremely important for the algorithm convergence. The perturbation strength is typically chosen so that the relation $\delta J_t \sim 0.01\ (1 - J_t)$ for metric disturbance is held. This relation can be usually achieved using the following parametric representation for perturbation deviation $\sigma_t^k = a_\sigma^k(1 - J_t)^\mu + b_\sigma^k,\ k \in \{1, ..., K\}$, where $a_\sigma^k \geq 0$, $b_\sigma^k \geq 0$ and $\mu \geq 0$ some parameters to adjust. The gain coefficient is commonly taken in the form $\gamma_t^k = a_\gamma^k\left(1 - J_t\right) + b_\gamma^k, k \in \{1, ..., K\}$, where $a_\gamma^k, b_\gamma^k \geq 0$ are also parameters for adjustment. In the most frequently used case one has $\sigma_t^k = \sigma_t$, $\gamma_t^k = \gamma_t$ for all $k \in \{1, ..., K\}$ and the SPGD algorithm can be formulated as follows:

(0) Initialization: $t = 0$, $\mathbf{u}_0 = 0$;

(1) Either measure $J_t = J\left(I_t\right)$ or measure $I_t = I\left(\boldsymbol{\rho}, t; \mathbf{u}_t\right)$ and compute J_t;

(2) Draw random, uniformly distributed, zero-centered, vector $\delta\mathbf{u}_t$ with uncorrelated components having variance σ_t each;

(3) Either measure $J_{t+1/2} = J\left(I_{t+1/2}\right)$ or measure $I_{t+1/2} = I\left(\boldsymbol{\rho}, t + 1/2; \mathbf{u}_t + \delta\mathbf{u}_t\right)$ and compute $J_{t+1/2}$;

(4) Compute metric increment $\delta J_t = J_{t+1/2} - J_t$ and $\gamma_t = a_\gamma\left(1 - J_t\right) + b_\gamma$;

(5) Update control parameters $\mathbf{u}_{t+1} = \mathbf{u}_t + \gamma_t\delta J_t\delta\mathbf{u}_t$.

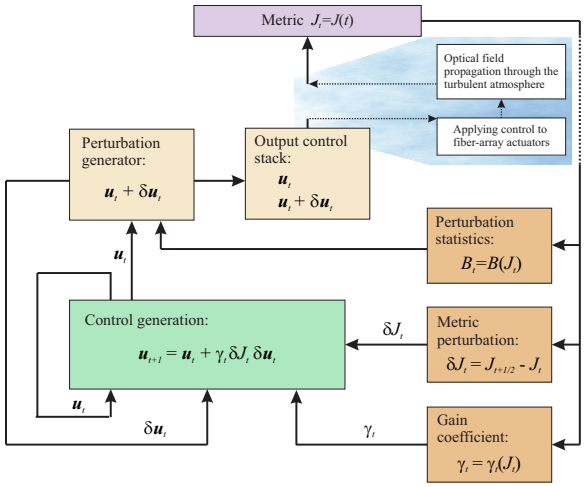

Figure 3: The schematic representation of SPGD controller operational scenario.

This is conventional two-step (see explanations below) SPGD algorithm where, in order to decouple perturbation and gain factors, the multiplicative term $(\sigma_t)^{-1}$ in the formula (9) is added to the gain coefficient. The operational scenario of SPGD controller is schematically shown in the FIGURE 3.

3.2 The AI Controller - DNN Topology

As it is usual for machine learning applications, design of AI controller begins from the specification of DNN topology that will represent the vector parametric function $\mathbf{U}_t(\boldsymbol{\alpha})$. Recall that AI controller for the power beaming problem will be continuously provided by measurements from PVA panel having the form of two-dimensional greyscale square image map $I_t = I(\boldsymbol{\rho}, t; \mathbf{u}_t)$. Supposing that this intensity distribution may contain some information useful for generation of optimal control the input DNN layers should be capable to extract this information from the image. It is well-known that the basic network structure which can provide such kind of analysis is convolutional neural network (CNN) [24]. The CNN can be interpreted as a set of trainable digital filters dedicated to extract features from the incoming images and structurally consists of several convolution and downsampling (or pooling) layers. As far as we deal with time-dependent image flow I_t the feature vector outcoming from the CNN will also keep dependency over time so that for the next and core DNN structure it is reasonable to take recurrent neural network (RNN) [25], especially designed for temporal data processing. The recurrent layers allow to include into analysis short- and long-term temporal relations in incoming data as well as synthesize complex and temporally correlated high-order feature vector. The conversion of feature vector outcoming from the RNN into controls is traditionally performed using several time-distributed fully-connected layers providing final analysis occurring in high-order feature space. In accordance from this design for DNN of AI controller one can propose the following topology represented in the FIGURE 4.

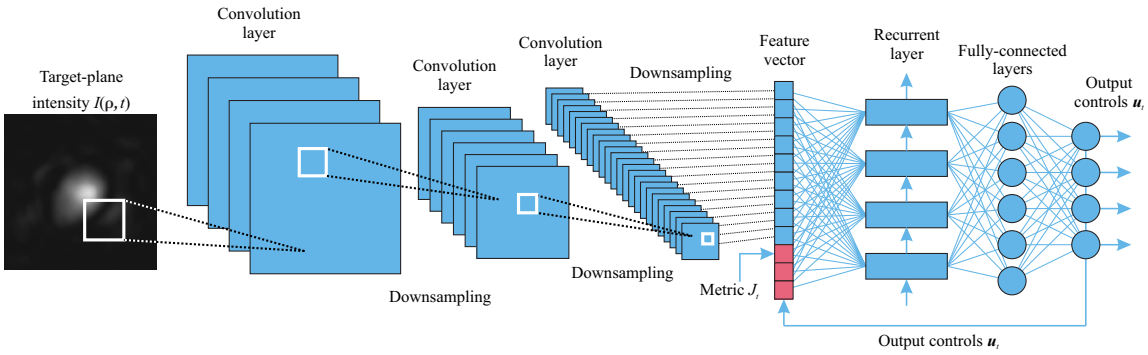

Figure 4: The schematic representation of DNN topology for AI controller.

The DNN in the FIGURE 4 has an input having the form of 4-dimensional matrix (also called as tensor in machine learning applications) of the shape $(1, N_{\text{ws}}, N_x, N_y)$, where the first, so-called, batch dimension equals to 1 due to controller's operation in real-time regime, $N_{\text{ws}} \geq 1$ is the time dimension, especially introduced for efficient training of recurrent layers, and the last two feature dimensions equal to the image size (N_x, N_y). Later on, the DNN input is directed into CNN having three sequentially placed time-distributed convolutional and max-pooling layers. The CNN output, in turn, is converted into time-distributed flat feature vector. This vector is supplemented with metric value J_t and DNN control outputs at the previous time step $\mathbf{u}_{t-1}$ and, after that, is directed into stateful gated recurrent unit (GRU) [26], containing $10K$ recurrent neurons. The GRU output is fully connected to two sequentially placed time-distributed dense (perceptron) layers with "*tanh*"-type activation function and $6K$ neurons for the internal layers and "*linear*" activation function and K neurons for the third dense output layer. It is easy to see that DNN output will have appropriate shape $(1, N_{\text{ws}}, K)$ for generation of K-dimensional control vector. Note that DNN of such particular architecture for $K = 3 \cdot 19 = 57$ (fiber array with $N_{\text{sa}} = 19$ subapertures and piston/tip-tilts control) and $N_x = N_y = 256$ (PVA with 256×256 resolution) has approximately $P \sim 6 \times 10^4$ trainable parameters $\boldsymbol{\alpha}$.

3.3 The AI controller – Training and inference

After choosing of DNN topology for the AI controller it is necessary to specify its training and inference algorithm. Here we present two variants of this algorithm differ from each other by SPGD approximation of metric gradient – first algorithm (two-step algorithm) uses conventional perturbation strategy as well as the second one fuses perturbations with control trajectory (one-step algorithm).

The two-step algorithm for inference and training of the AI controller can be formulated as follows:

(0) Initialize $t = 0$, $\mathbf{u}_0 = 0$ and DNN trainable parameters $\boldsymbol{\alpha}_0$;

(1) Measure $I_t = I(\boldsymbol{\rho}, t; \mathbf{u}_t)$ and compute $J_t = J(I_t)$;

(2) *Training phase (optional)*:

(2.1) Draw random, uniformly distributed, zero-centered, vector $\delta\mathbf{u}_t$ with uncorrelated components having variance σ_t each;

(2.2) Measure $I_{t+1/2} = I\left(\boldsymbol{\rho}, t+1/2; \mathbf{u}_t + \delta\mathbf{u}_t\right)$ and compute $J_{t+1/2} = J\left(I_{t+1/2}\right)$;

(2.3) Compute metric increment $\delta J_t = J_{t+1/2} - J_t$ and $\gamma_t = a_\gamma\left(1 - J_t\right) + b_\gamma$;

(2.4) Update trainable parameters:

$$\alpha_{t+1}^p = \alpha_t^p + \frac{\gamma_t}{a_\sigma}\delta J_t \sum_{k=1}^{K} \delta u_t^k \frac{\partial U^k\left(I_t, \mathbf{u}_t, J_t; \boldsymbol{\alpha}_t\right)}{\partial \alpha^p}, \;\; p \in \{1, ..., P\}; \tag{12}$$

(3) *Inference*: compute $\mathbf{u}_{t+1} = \mathbf{U}\left(I_t, \mathbf{u}_t, J_t; \tilde{\boldsymbol{\alpha}}_{t+1}\right)$, where $\tilde{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}_{t+1}$ if training step was performed or $\tilde{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}_t$ if not.

The corresponding one-step algorithm can be written as:

(0) Initialize $t = 0$, $\mathbf{u}_0 = 0$, $I_0 = 0$, $J_0 = 0$ and trainable parameters $\boldsymbol{\alpha}_0$;

(1) *Inference*: compute $\mathbf{u}_{t+1} = \mathbf{U}\left(I_t, \mathbf{u}_t, J_t; \boldsymbol{\alpha}_t\right)$;

(2) *Training phase (optional)*: draw random, uniformly distributed, zero-centered, vector $\delta\mathbf{u}_{t+1}$ with uncorrelated components having variance σ_{t+1} each;

(3) Set $\mathbf{w}_{t+1} = \mathbf{u}_{t+1} + \delta\tilde{\mathbf{u}}_{t+1}$, where $\delta\tilde{\mathbf{u}}_{t+1} = \delta\mathbf{u}_{t+1}$ if training phase is turned on or $\delta\tilde{\mathbf{u}}_{t+1} = 0$ if not.

(4) Measure $I_{t+1} = I\left(\boldsymbol{\rho}, t+1; \mathbf{w}_{t+1}\right)$ and compute $J_{t+1} = J\left(I_{t+1}\right)$;

(5) *Training phase (optional)*:

(5.1) Compute metric increment $\delta J_t = J_{t+1} - J_t$ and $\gamma_{t+1} = a_\gamma\left(1 - J_{t+1}\right) + b_\gamma$;

(5.2) Update trainable parameters:

$$\alpha_{t+1}^p = \alpha_t^p + \frac{\gamma_{t+1}}{a_\sigma}\delta J_t \sum_{k=1}^{K} \left(w_{t+1}^k - u_t^k\right) \frac{\partial U^k\left(I_{t+1}, \mathbf{w}_{t+1}, J_{t+1}; \boldsymbol{\alpha}_t\right)}{\partial \alpha^p}, \;\; p \in \{1, ..., P\}. \tag{13}$$

Here, as in section 3.1, we made several simplifications with coefficients and gains.

It is clear that one-step algorithm supposes to pass controls through the media just once in opposite to two-step version where it should be done twice per iteration. Implementation shows that in rapidly changing media this advantage can be avoid by instability of one-step iteration process and corresponding control trajectories.

The proposed algorithms structurally combine two regimes: DNN inference for synthesizing optimal control as well as DNN weights updating for training. If the controller operates in pure inference mode, when DNN weights updating is off, the controller uses previously collected knowledge to control the system. In training mode control synthesis as before is provided via DNN inference however DNN weights updating mechanism additionally

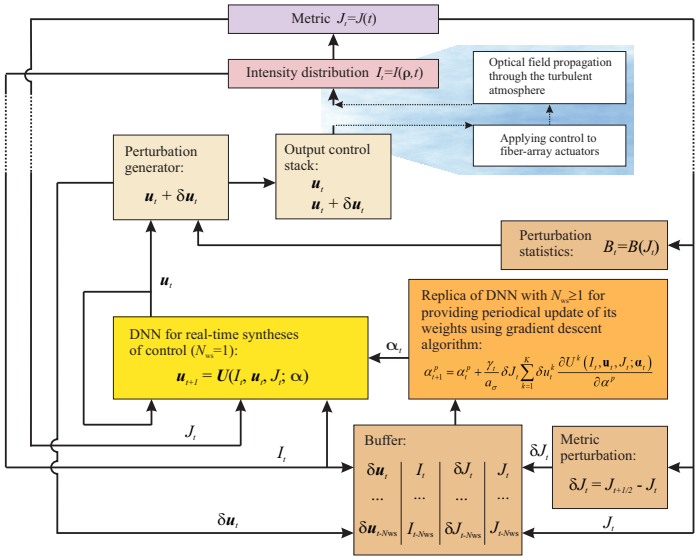

Figure 5: The schematic representation of AI controller operating in training mode.

provides SPGD-type optimization occurring now in training parameters $\boldsymbol{\alpha}$ space instead of initial control parameters $\mathbf{u}$ space as in conventional SPGD algorithm. As we will see below this combination of optimization and training in one process leads to a quite interesting numerical effects.

Utilizing of RNN layers in DNN topology as well as providing stability of training process require to have two replicas of DNN (represented as computational graphs) sharing one set of trainable parameters: one copy for inference mode, where the optimal control is synthesized, with $N_{\text{ws}} = 1$ and another copy for training mode with $N_{\text{ws}} \geq 1$. Note that, in case $N_{\text{ws}} > 1$ weights updating increment in formulas (12) and (13) should be slightly modified by adding extra summation over sequential N_{ws} time steps. Updating of the training parameters $\boldsymbol{\alpha}_t$ in both algorithms can be occurred by any reasonable strategy (periodically, by achieving some metric value and so on) and supposes to use a buffer of the size N_{ws} containing all DNN inputs as well as estimations of metric gradient with the corresponding perturbations.

The detailed illustration of AI controller operational scenario is given in the FIGURE 5.

3.4 Combined SPGD and AI Control

In the consideration above it was implicitly supposed that the controller has all necessary information for synthesis of correct control in any time. However, real-world physical systems, as a rule, operate in conditions when information available for the analysis is not full and the control based on this information can be non-unique. In addition, realistic systems usually operate in presence of stochastic and unpredictable factors that can not be accounted using deterministic control generators. In this case the proposed approach has to be supplemented by adding some mechanism that can compensate such unpredictable stochastic factors as well as the lack in input information.

Consider slightly modified formula for the control in the form:

$$\mathbf{u}_t = \mathbf{w} + \mathbf{U}_t\left(\boldsymbol{\alpha}, \mathbf{w}\right),$$

where $\mathbf{w} = \left(w^1, ..., w^K\right)$ is the part of the control responsible for compensation of aforementioned factors. Then, the optimization problem (10) can be rewritten as:

$$J_t\left(\mathbf{w}, \boldsymbol{\alpha}\right) = J\left(\mathbf{I}_t\left(\boldsymbol{\rho}; \mathbf{w} + \mathbf{U}_t\left(\boldsymbol{\alpha}, \mathbf{w}\right)\right)\right),$$
$$J_t\left(\mathbf{w}, \boldsymbol{\alpha}\right) \to \max_{\boldsymbol{\alpha}, \mathbf{w}}.$$

Under condition of uniform boundedness of DNN derivative over $\mathbf{w}$ variables, i.e. $\left|\partial U^k / \partial w^l\right| \leq C$, for some constant $C > 0$ and any $k, l \in \{1, ..., K\}$, the optimal control trajectory can be approximated as:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\gamma_t}{\sigma_t}\delta J_t \delta \mathbf{u}_t,$$
$$\mathbf{v}_{t+1} = \mathbf{U}\left(I_t, \mathbf{v}_t, \mathbf{w}_{t+1}, J_t; \boldsymbol{\alpha}_t\right),$$
$$\mathbf{u}_{t+1} = \mathbf{w}_{t+1} + \mathbf{v}_{t+1},$$

so that we factually have combination of DNN-based and SPGD control. The corresponding training and inference algorithm in this case can be written as:

(0) Initialize $t = 0$, $\mathbf{w}_0 = 0$, $\mathbf{v}_0 = 0$ and DNN trainable parameters $\boldsymbol{\alpha}_0$;

(1) Calculate $\mathbf{u}_t = \mathbf{w}_t + \mathbf{v}_t$;

(2) Draw random, uniformly distributed, zero-centered, vector $\delta\mathbf{u}_t$ with uncorrelated components having variance σ_t each;

(3) Measure $I_{t+1/2} = I\left(\boldsymbol{\rho}, t + 1/2; \mathbf{u}_t + \delta\mathbf{u}_t\right)$ and compute $J_{t+1/2} = J\left(I_{t+1/2}\right)$;

(4) Compute metric increment $\delta J_t = J_{t+1/2} - J_t$ and $\gamma_t = a_\gamma\left(1 - J_t\right) + b_\gamma$;

(5) *SPGD optimization*: calculate

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\gamma_t}{\sigma_t}\delta J_t \delta \mathbf{u}_t;$$

(6) *Training phase(optional)*: update trainable parameters

$$\alpha^p_{t+1} = \alpha^p_t + \frac{\gamma_t}{a_\sigma}\delta J_t \sum_{k=1}^{K} \delta u^k_t \frac{\partial U^k\left(I_t, \mathbf{v}_t, \mathbf{w}_{t+1}, J_t; \boldsymbol{\alpha}_t\right)}{\partial \alpha^p}, \;\; p \in \{1, ..., P\};$$

(7) *Inference*: compute $\mathbf{v}_{t+1} = \mathbf{U}\left(I_t, \mathbf{v}_t, \mathbf{w}_{t+1}, J_t; \tilde{\boldsymbol{\alpha}}_{t+1}\right)$, where $\tilde{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}_{t+1}$ if training step was performed or $\tilde{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}_t$ if not.

Note that, in general case SPGD- and DNN-based controls can be multiplied by some gains in order to effectively balance a contribution of each mechanisms in optimization.

3.5 Regularization

It is well-known that control systems operated in accordance with active feedback control scheme have a tendency to forming of positive feedback loops that leads to significant instability in its operation and training. The AI controller is not an exception to this rule and weights updating mechanism represented above does not guarantee an absence of positive feedbacks in control. The traditional solution avoiding this problem is utilizing of regularized optimization metrics as well as smoothing of metric gradients. In order to realize these ideas, at first, let us consider two functionals. First functional is designed to bound deviations in output control trajectories during training process and has the form:

$$S_t\left(\boldsymbol{\alpha}\right) = \gamma^S \sum_{k=1}^{K} \left(\frac{dU_t^k\left(\boldsymbol{\alpha}\right)}{dt} \right)^2, \tag{14}$$

where $\gamma^S \geq 0$ is the gain (typically, $\gamma^S \sim 10^{-2}$). Note that in (14) exactly the full derivative of functions $U_t^k\left(\boldsymbol{\alpha}\right), k \in \{1, ..., K\}$ over time is placed in order to avoid instability originated not only from high-order oscillations of controller's output but from metric and DNN input jitters too.

In order to additionally smooth the output control as well as to prevent DNN from the overtraining the traditional L_2 functional applying directly to DNN training weights is used:

$$L_2\left(\boldsymbol{\alpha}\right) = \gamma^L \|\boldsymbol{\alpha}\|^2 = \gamma^L \sum_{p=1}^{P} \alpha_p^2,$$

where $\gamma^L \geq 0$ is the gain (typically, $\gamma^L \sim 10^{-3}$).

Finally, training of DNN using formula (11) can meet convergence process instability due to chaotic changing of random perturbations. In order to avoid these gradient oscillations the standard momentum [27], or Adam algorithm can be applied [28]. Combining gradient smoothing with metric regularization one can rewrite weights updating step as:

(0) Initialize $t = 0$, $g_0^p = 0$;

(1) Update trainable parameters:

$$g_{t+1}^p = \nu g_t^p + \frac{\partial}{\partial \alpha^p} \left(\sum_{k=1}^{K} \left(\frac{\gamma_t}{a_\sigma} \delta J_t \delta u_t^k U_t^k - \gamma^S \left(U_t^k - U_{t-1}^k \right)^2 \right) \right) - 2\gamma^L \sum_{p=1}^{P} \alpha_p;$$

(2) Calculate $\alpha_{t+1}^p = \alpha_t^p + g_t^p, p \in \{1, ..., P\}$.

Here $\nu \in [0, 1)$ is smoothing factor typically chosen as $\nu = 0.9$. Note that the increment in the step (1) is specially given in the form where partial derivatives over trainable weights are

taken from the weighted sum of DNN outputs that maximally convenient for implementation using popular machine learning frameworks.

3.6 Decoupling of Control Channels

Returning back to physical statement of the problem it is necessary to note that the configuration of control parameters represented by the formula (2) provides extremely strong coupling between control channels. This situation is connected with physics of fiber-array beamlets propagation given by equation (1). For example, effective target-plane focusing of the fiber-array at time $t \geq 0$ needs to configure pistons and tip-tilts so that the sum $\sum_{n=1}^{N_{\text{sa}}} H_n(\boldsymbol{\rho})\varphi_n(\boldsymbol{\rho}, t)$ on fiber-array aperture will approximate paraboloid function representing thin lens. The same principle is fulfilled and for the control of focal spot displacement – in this case one need to approximate the sum of paraboloid and linear function.

It is well-known that robust synthesis of control for a system with coupled channels is much more complicated problem in comparison with control of the same system but with decoupled channels. As it was noted before modern SPDG controller provides extremely high iteration rate and mentioned problem is not quite actual. However, for AI control this problem can be actual especially taking into account that here the optimization process is performed in N_{ws} times less often regarding to SPGD. To avoid this drawback let us partially decouple fiber-array control channels introducing following decomposition of control parameters.

At pupil-plane $z = 0$ consider fiber-array aperture (see FIGURE 2, left) and let $\{Z_q(\boldsymbol{\rho})\}_{q=1}^{Q}$ be the family of mutually-orthogonal and mean-square normalized Zernike polynomials on the aperture disc, where $Q = (N_Z + 1)(N_Z + 2)/2 - 1$ and $N_Z \geq 1$ is maximal degree of polynomials in this set [29]. It is well-known that first five Zernike polynomials ($N_Z = 2$) represent fundamental optical aberrations including x- and y- slopes and defocus and hence can be considered as a basis for reducing complexity of the light-spot control. Extending this assumption to the polynomials of degree more than 2 let us build the transformation between coefficients in Zernike polynomials space and fiber piston/tip-tilt basis. For $q \in \{1, ..., Q\}$ define:

$$r_{kq} = \frac{4}{\pi d^2} \begin{cases} \int H_k(\boldsymbol{\rho}) Z_q(\boldsymbol{\rho}) d^2\boldsymbol{\rho}, & k \in \{1, ..., N_{\text{sa}}\}, \\ \int (x - x_{k-N_{\text{sa}}}) H_{k-N_{\text{sa}}}(\boldsymbol{\rho}) Z_q(\boldsymbol{\rho}) d^2\boldsymbol{\rho}, & k \in \{N_{\text{sa}} + 1, ..., 2N_{\text{sa}}\}, \\ \int (y - y_{k-2N_{\text{sa}}}) H_{k-2N_{\text{sa}}}(\boldsymbol{\rho}) Z_q(\boldsymbol{\rho}) d^2\boldsymbol{\rho}, & k \in \{2N_{\text{sa}} + 1, ..., 3N_{\text{sa}}\}. \end{cases}$$

Then, the matrix $R = [r_{kq}]_{k=1,q=1}^{K,Q}$ of the size $K \times Q$ provides transformation between Zernike control coefficients and conventional piston/tip-tilt representation so that $\mathbf{u} = \mathbf{u}_Z R$, where $\mathbf{u}_Z$ is Q-dimensional control vector in Zernike space.

Note that, for any particular configuration of fiber-array subapertures the matrix R is computed once, each transformation from Zernike representation to fiber-array controls is re-

quired just one matrix multiplication by a vector and can be performed extremely fast even comparing with SPGD iteration speed.

4. NUMERICAL RESULTS

4.1 Numerical Verification of AI Controller's Training Capabilities

In order to verify training and optimization capabilities of the proposed AI-based control approach let us consider simple tracking system. Let $M = 1$, $K = 2$ and

$$I_t\left(x, y; u^1, u^2\right) = e^{-\left[(x-x(t)+u^1)^2+(y-y(t)+u^2)^2\right]/\beta_1^2}, \tag{15}$$

where $\boldsymbol{\rho}(t) = (x(t), y(t))$ is some predefined trajectory in Oxy coordinates and $\beta_1 > 0$. From the formula (15) it is clear that "intensity" distribution simulated by the formula (15) under control $\mathbf{u}_t = \left(u^1(t), u^2(t)\right)$ will represent a small spot located at the point $\left(x(t) - u^1(t), y(t) - u^2(t)\right)$ for any $t \geq 0$. Let $\boldsymbol{\rho}(t) = (\sin(\omega t), \cos(\omega t))$ with some $\omega > 0$ so that circular motion of the spot is simulated.

As an tracking objective consider the problem of "catching and holding" of the spot at the coordinates origin. In this case for the performance metric one can take:

$$J\left(I_t\left(x, y; u^1, u^2\right)\right) = \frac{1}{\pi}\left(\frac{1}{\beta_1^2} + \frac{1}{\beta_2^2}\right)\int I_t\left(x, y; u^1, u^2\right) e^{-(x^2+y^2)/\beta_2^2} dxdy, \tag{16}$$

where $\beta_2 > 0$ defines the metric tolerance to the spot position. It is easy to see that metric (16) stimulates to synthesize control trajectory $\mathbf{u}_t$ maximally closed to spot trajectory $\boldsymbol{\rho}(t)$ and, moreover, this metric is equivalent to smooth Strehl ratio introduced before. For the AI controller consider the same DNN as has been proposed in section 3.2.

In the numerical experiments the simulation area was $[-5.0, 5.0]_x \times [-5.0, 5.0]_y$ and had 256x256 pixels resolution, $\beta_1 = 0.4$ and $\beta_2 = 1.0$, the simulation time step was 0.1 sec so that the frequency factor ω provided 1 full rotation of the spot per approximately 5 nominal minutes. The DNN grayscale input image had the same 256x256 pixel resolution, $N_{\text{ws}} = 4$ and total number of trainable parameters for this DNN was about 4.5×10^4. The simulations were performed in Python v.3.7 [30], the simulator was completely separated from the controller which was implemented using Tensorflow v.1.14 [31], library for Python. For inference and training of the AI controller the two-step algorithm represented in section 3.3 was used. Overall simulation and training time on ASUS ROG Zephyrus[2] took about 30 seconds on 1 minute of simulation time.

The FIGURE 6 represents numerical results of AI control performance and corresponding control trajectories for two different optimization and learning strategies. The first strategy

[2] Intel Core i7-9750H 2.60GHz, 32 Gb RAM with Nvidia GeForce RTX 2080 with Max-Q Design, 8Gb VRAM.

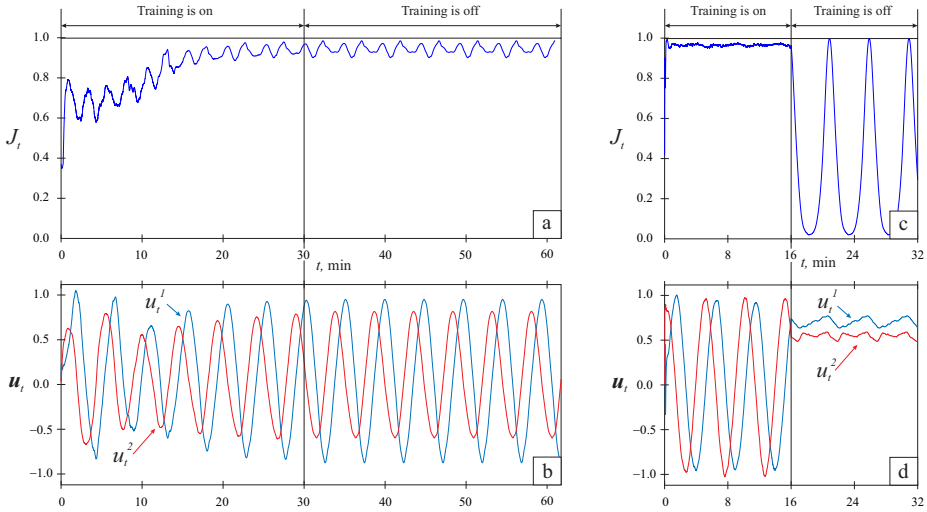

Figure 6: Performance metric J_t (a, c) and trajectories $\mathbf{u}_t$ (b, d) versus time for two learning strategies in the tracking task. Charts (a, b) represent "soft" training strategy with relatively small learning rate and perturbation strength. In opposite, in charts (c, d) more "aggressive" strategy is chosen in order to rapidly achieve metric maximum.

(FIGURE 6, a–b) supposes to make relatively slow motions in the spot's direction with relatively small learning rate and perturbation strength. In opposite to this scenario the second strategy (FIGURE 6, c–d) has the aim of maximally rapid "catching" of the spot and accompany it in its motion. It is easy to see that just the first strategy allows controller to learn the tracking principle and hence allows it to operate without SPGD-type iterations. The second strategy failed teaching the controller so that turning off the SPGD iterations immediately led to loosing of the spot. Moreover, continuing using the controller in such aggressive regime without significant regularization rapidly fell the controller to instability with generation of "saw"-type control.

Note that the same results were achieved using Gaussian randomly drawn two-parametric process $\boldsymbol{\rho}(t)$ with mutually uncorrelated components[3].

4.2 Numerical Results for Power Beaming Problem

Consider hexagonal fiber-array consisting of $N_{\text{sa}} = 19$ densely packed subapertures of the diameter $d = 60$ mm each with distances between subapertures equals also to 60 mm (FIGURE 2 on the left) and Gaussian beamlet diameters $a_0 = 0.9d$. In our analysis, we consider horizontal propagation scenario at the distance $L = 5000$ m and vary the refractive-index-structure parameter characterizing homogeneous atmospheric turbulence strength, C_n^2, in the range $5.0 \times 10^{-16} - 1.5 \times 10^{-15}\,\text{m}^{-2/3}$. Simulation of atmospheric turbulence changes in time is realized by introducing of wind transversal to the propagation direction with constant speed w in the range $1 - 6$ m/s. The Airy diameter at the target-plane is $d_{\text{a}} = 40$ mm for wavelength $\lambda = 1.064\,\mu$m ($k = 5.905 \times 10^6\,\text{m}^{-1}$) and target-plane PVA has square shape with the side $D = 200$ mm.

[3] For this scenario Python code is available for testing by the request to authors.

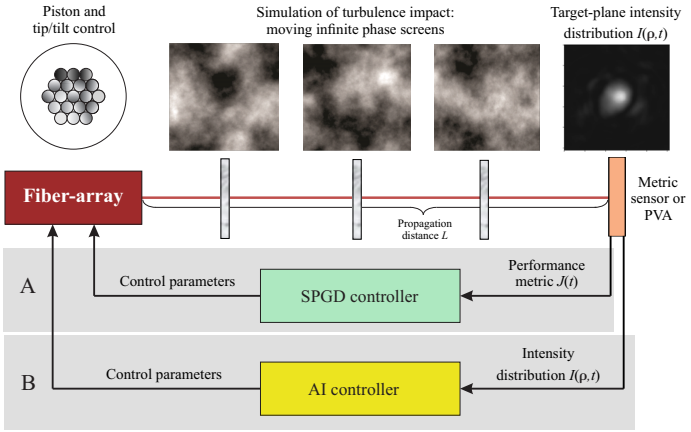

Figure 7: Illustration of simulation setup for fiber-array power beaming problem in presence of atmospheric turbulence. The control of fiber-array output field can be performed either (A) SPGD control system or (B) AI controller.

The numerical simulations of atmospheric propagation of the optical wave from fiber-array to PVA is performed through numerical integration of (1) using so-called split-operator method [32, 33], when the impact of turbulence-induced aberrations is modeled with equidistantly placed thin phase screens (see FIGURE 7). The conventional approach [32], assumes using of phase screens with finite size in both x and y directions. The simulation of atmospheric wind requires continuously shifting of these screens in Oxy plane with the need of their extension if wind-induced displacement will exceed of the screen size. This extension traditionally can be done either using periodical (in both directions) finite phase screens or utilizing phase screens infinitely-long in one specified direction [34]. In our simulations we use both approaches considering propagation scenario with periodic phase screens as a reduced complexity control task. The number of phase screens in numerical simulations is varied in the range 5-10.

The simulation square in xy-plane is centered at the origin, completely covers PVA area and has 800 mm in both directions with corresponding numerical grid having 1024x1024 pixels resolution. The phase screen resolution is chosen the same. The simulation time step sets to 5.0×10^{-5} sec so that the SPGD controller is factually operated at 2×10^4 iteration per second and DNN training has $2\times10^4/N_{\text{ws}}$ weight updates per second. The DNN input is provided by PVA images that have 256x256 pixels resolution. The window size for the training DNN's replica is set to $N_{\text{ws}}=4$.

The performance metric is chosen as smooth Strehl ratio introduced in section 2.2. The numerical integration in the formulas (3)-(4) is performed over PVA area only. The metric smoothing factor is $\beta=d_{\text{a}}/4$.

The fiber-array is supplied by a optimization controller capable to operate with phase pistons and tip-tilts, so that overall number of control parameters is $K=3\times N_{\text{sa}}=57$. In addition, we will assume that this controller is capable to convert controls from Zernike space to piston/tip-tilt representation, i.e. compute the transformation $\mathbf{u}=\mathbf{u}_Z R$ (see section 3.6), and this transformation can be done with no additional time consumption. In our numerical experiments we utilize the family of Zernike polynomials with degree $N_Z\le 4$ ($Q=14$ polynomials totally) including: x- and y- slopes (coefficients u_Z^1 and u_Z^2, correspondingly),

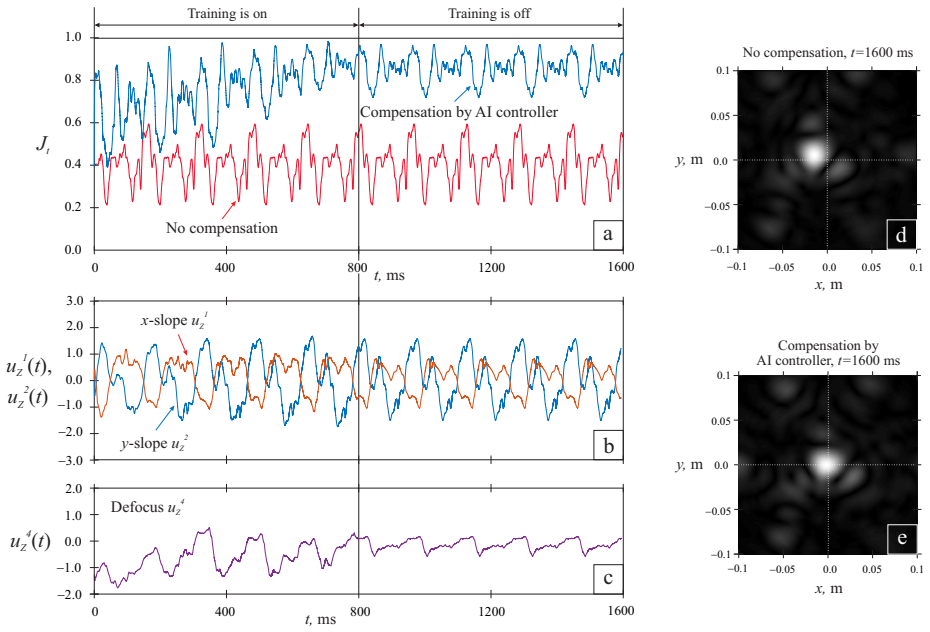

Figure 8: The chart (a) represents comparison of AI controller compensation performance with "no compensation" curve taken for the period $[0, 1600]$ ms. The AI controller operates in both training (during first 800 ms) and inference (during the following 800 ms) modes, control is performed in Zernike space using dimensionless coefficients at first $Q = 14$ Zernike polynomials and periodical phase screens were taken for simulation of atmospheric turbulence. The charts (b) and (c) represent trajectories for x- and y- slopes (u_Z^1 and u_Z^2 control variables) and defocus (u_Z^4 variable) versus time. The images (d) and (e) represent square roots from intensity distributions taken in PVA area $[-0.1, 0.1]_x \times [-0.1, 0.1]_y$ m^2 at $t = 1600$ ms for system operating with no compensation (d) and with AI controller (e). In this numerical experiment $C_n^2 = 1.0 \times 10^{-15}$, m$^{-2/3}$ and wind speed $w = 5$ m/s.

oblique astigmatism (u_Z^3), defocus (u_Z^4) and vertical astigmatism (u_Z^5). Note that numerical simulation and modeling of fiber-array optical system was performed using WONAT software library adapted for Python 3.7 [35].

In our analysis we consider four scenarios of control system operation: (1) propagation with no compensation, (2) compensation by SPGD algorithm (FIGURE 7, A), (3) compensation by AI controller in training mode and (4) compensation by AI controller purely in inference mode (FIGURE 7, B for both modes).

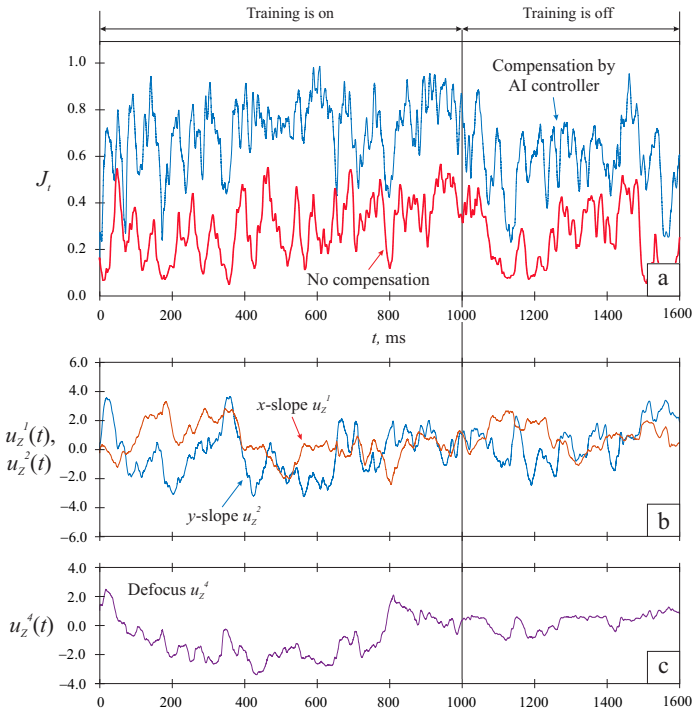

Figure 9: The chart (a) represents comparison of AI controller compensation performance with "no compensation" curve taken for the period $[0, 1600]$ ms. Atmospheric turbulence is modeled using infinite aperiodic phase screens and control is performed in Zernike space with $Q = 14$ polynomials. Training phase of AI controller continues for a first simulation second and all remaining time controller operates in inference mode. The charts (b) and (c) represent trajectories for x- and y-slopes (u_Z^1 and u_Z^2 control variable) and defocus (u_Z^4 variable) versus time. Here $C_n^2 = 1.0 \times 10^{-15}$, $\mathrm{m}^{-2/3}$ and wind speed $w = 5$ m/s.

In order to be assure that AI controller is configured properly and its operating is accompanied by its learning, at first, consider 5 finite and periodic phase screens slowly moving by constant wind $w = 5$ m/s. The total observation period is taken as 1600 ms so that phase screens have a time to make 10 complete cycles. Overall time interval is separated into two equal sub-intervals – for controller training and its inference with no training. The FIGURE 8 represents the results of this experiment. During first 800 ms in the training mode the controller reaches approximately $\langle J_t \rangle = 0.8$ compensation performance via active control of Zernike coefficients including x- and y- slopes and defocus. Turning the training off after 800 ms keeps compensation approximately at the same level, however, the most important that control variables are continuing to repeat the control patterns learned during training period. This example demonstrates that even for such a short training time the controller was

able to remember one of the possible optimization strategy completely rely on observation of target-plane image distributions.

The next numerical experiment is dedicated to verify the controller's generalization properties – can the controller learn the control strategy in conditions of aperiodic changing of atmospheric turbulence. For this task let us consider 6 infinite aperiodic phase screens moving by constant wind $w = 6$ m/s and compare compensation results of AI controller with "no compensation" metric curve. The FIGURE 9 represents results of this experiment. During the first simulation second the performance of AI control oscillates approximately around the value $\langle J_t \rangle = 0.7$ and a wide spread observed for both of curves can be explained by using of infinitely-long phase screens having large turbulence outer-scale [34]. During this period the controller is actively looking for optimal trajectories of control variables that follows from the form of curves represented in FIGURE 9, (b) and (c) on the left of solid vertical line. After turning the training off the overall compensation level is degraded to $\langle J_t \rangle = 0.5$ however the controller proceeds active changing of phase slopes in order to place target-plane light spot in central position. In parallel with slopes correction the controller is tried to optimize the spot size by control of beam focusing however is doing that not so active as before either due to sufficient focusing during training phase or poor training of this capability. Thus, this example explicitly shows that the controller is capable to rather prompt "understand" elementary relations between phase slope and target spot position, beam focusing and spot size and can account these relation during synthesis of the control in inference mode.

The comparison of AI and SPGD control performance will be done in two steps. Let the control is performed directly using piston/tip-tilt variables. In the first experiment we will mostly focus on AI controller learning capabilities instead of its compensation performance and consider a "soft" training strategy with constant learning rate $\gamma_t = 10^{-3}$, random perturbations with $\mu = 1$ and will suppose presence of all types of regularization introduced in section 3.5. The FIGURE 10, (a) represents performance curves for the system operating with: no compensation, compensation by SPGD controller with 2×10^4 iteration per second and compensation by AI controller having 5×10^3 weights updates per second. During training period of 1200 ms the SPGD and AI controllers provide approximately the same compensation quality and corresponding metric curves partially repeat each other except several intervals where AI controller operates better. After turning the training and SPGD iterations off the compensation behavior is dramatically changed – the SPGD curve partially begins to repeat "no compensation" curve whereas AI controller continues to provide the compensation but not so effective as during training phase.

In the second experiment we will focus on compensation performance of AI controller instead of its learning capabilities and use more "aggressive" training strategy with enhanced learning rate $\gamma_t = 10^{-2}$, $\mu = 0.5$ and regularization applied to DNN outputs only (i.e. using functional S_t only, see section 3.5). As before, the control is synthesized directly using piston/tip-tilt variables. The FIGURE 10, (b) represents numerical results for exactly the same realizations of infinite phase screens as in FIGURE 10, (a). It is easy to see that during training the AI control curve is now lying a little bit higher than SPGD curve and, in general, AI control has less falls in performance in comparison with SPGD control. However, turning the training

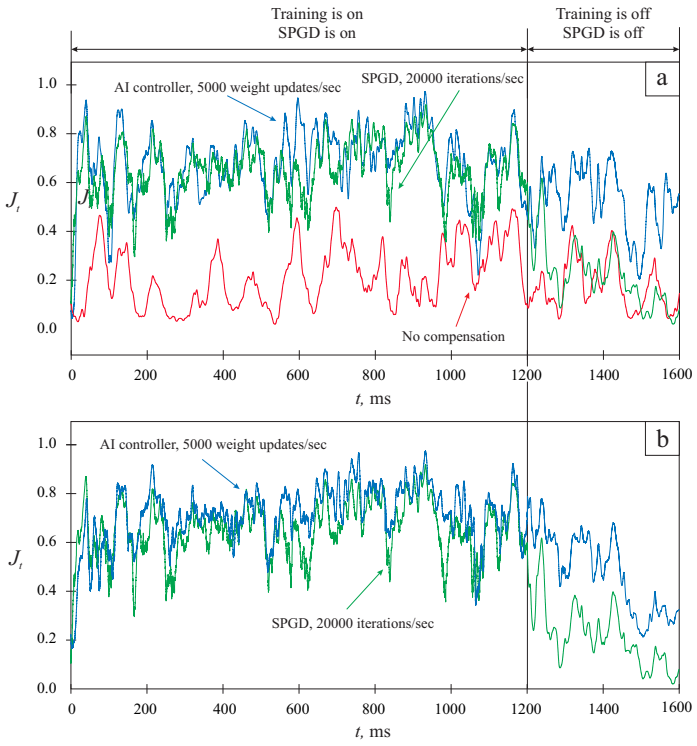

Figure 10: The chart (a) represents comparison of SPGD- and AI-based control system performance, where infinite aperiodic phase screens were used and both algorithms are operated with piston and tip-tilt control. The SPGD optimization as well as training phase of AI controller continue for a first 1200 ms and all remaining time SPGD is turned off and AI controller operates in inference mode. The chart (b) represents analogous metric comparison with the same phase screens realizations as in (a) but for more "aggressive" training strategy of AI controller. Here $C_n^2 = 1.0 \times 10^{-15}$, $\mathrm{m}^{-2/3}$ and wind speed $w = 4$ m/s.

off leads AI controller to significantly worse compensation in comparison with compensation in FIGURE 10, (a) that can be definitely interpreted as the worse training in this mode.

The FIGURE 11 represents graphical summary of the results obtained via averaging over 5 seconds of corresponding metric values for four compensation strategies mentioned before. The results show that AI controller of proposed topology in training mode exceeds conventional SPGD algorithm up to approximately 5-7% and in inference mode lose SPGD up to 10% except slow 1 m/s wind velocity where potentially not a complete training was performed due to slow screens motion.

5. CONCLUSIONS

In this study we introduced self-learning AI controller and applied it for the power beaming problem. The major goal of this research was to verify that controller's DNN can be effectively trained in real time operational regime using solely SPDG-type gradient estimation so that after turning perturbations off the controller keeps a capability to synthesize appropriate control. The numerical results show that this goal can be achieved if during training we use "soft" optimization strategy and do not strive to come to maximal metric performance in a minimal time. In opposite, more "aggressive" optimization leads to low or even absolutely absence of DNN teaching. The trade off between these two optimization types is empirical and mostly depends on choice of SPGD perturbation statistics, DNN learning rate and its regularization parameters.

Another important factor that can be emphasized in connection with AI controller learning is a specific organization of training data stream. In the current implementation training process is performed using data portions (of the size N_{ws}) consequently derived from simulation engine so that DNN weights are updated on each data portion and after that this data is removed. Here we face several problems: (1) the data represented in nearby portions are strongly correlated that has negative effect on training speed, (2) complete removing of used data provides adaptivity of the controller to environment changes but produces instability in training due to factually absence of training dataset, (3) using just one training thread (our batch size equals to 1) leads to essential extension of training process as well as more poor training quality in comparison with multi-thread training. Technically, there is not any obstacle to resolve all these issues however it is a point for a future research.

In addition to different features of training process it is necessary to note the regularization role in proper controller's functioning. As was mentioned before, applying regularization to DNN outputs as well as smoothing of SPGD gradient allow to prevent such undesirable factor as forming of positive feedback which is typically expressed as generation of shock-type control. On the other hand, excessive regularization leads to non-plasticity in control as well as worsening of DNN training so that here we also have to keep a reasonable trade off. Decoupling of control channels can be also noted as important factor for the controller training and regularization but it is not a surprise due to validity of this observation for a wide class of control systems.

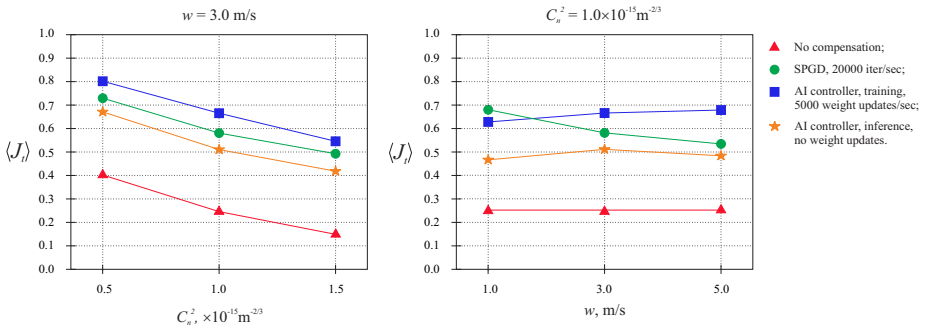

Figure 11: Comparison results of SPGD- and AI-based control performance for different atmospheric conditions. Both SPGD and AI controllers operate with piston/tip-tilt control variables, overall simulation time for averaging the results is 5 sec.

Returning back to power beaming with AI control it is necessary to highlight that DNN's incoming information in the form of PVA sensor data in principle does not contain all required information for synthesis optimal and unique phase pistons and tip-tilts. The lack of this information can be filled or passing additional sensor data to DNN input or including constantly working optimization mechanism like SPGD or DNN weights updating. In the last case the question how long DNN control will be stable during such a continuous training is still open.

In conclusion note that the proposed concept of active AI control potentially can be extended to a wide class of physical systems where SPGD provides appropriate estimation of metric gradient. Presence of metric singular points over control variables or in systems with strong inertia over control this approach should be corrected in the part of proper metric gradient estimation. However, in presence of boundary conditions and other strict limitations in control space and in systems where optimization process requires complicated predictive policy this approach, apparently, is not so effective and Q-function approach is preferable.

6. ACKNOWLEDGMENT

The authors are sincerely grateful to A.B. Lavrentyev who is the head of research in Kaspersky Lab where proposed principals of active AI control were pioneered. The authors are sincerely grateful also to M.A. Vorontsov for useful discussions of fiber-array power beaming problem.

References

[1] Leger JR, Nilsson J, Huignard JP, Napartovich AP, Shay TM, Shirakawa A. Introduction to the Issue on Laser Beam Combining and Fiber Laser Systems. IEEE Journal of Selected Topics in Quantum Electronics. 2009;15:237-239.

[2] Kudielka KH, Kalmar A, Leeb WR. Design and Breadboarding of a Phased Telescope Array For Free-Space Laser Communications. In Proceedings of International Symposium on Phased Array Systems and Technology. IEEE. 1996:419-424.

[3] Lachinova SL, Vorontsov MA. Exotic Laser Beam Engineering With Coherent Fiber-Array Systems. Journal of Optics. 2013;15:105501.

[4] Vorontsov M, Filimonov G, Ovchinnikov V, Polnau E, Lachinova S, et al. Comparative Efficiency Analysis of Fiber-Array and Conventional Beam Director Systems in Volume Turbulence. Applied Optics.2016;55:4170-85.

[5] Filimonov GA, Kolosov VV, Vorontsov AM. Computationally Efficient Methods for Simulating Laser Heating of Bulk Plates. Atmospheric and Oceanic Optics. 2021;34:617-624.

[6] Triviño A, González-González JM, Aguado JA. Wireless Power Transfer Technologies Applied to Electric Vehicles: A review. Energies. 2021;14:1547.

[7] Jin K, Zhou W. Wireless Laser Power Transmission: A Review of Recent Progress. IEEE Transactions on Power Electronics. 2018;34:3842-59.

[8] Kapranov VV, Matsak IS, Tugaenko VY, Blank AV, Suhareva NA. Atmospheric Turbulence Effects on the Performance of the Laser Wireless Power Transfer System. In Free-Space Laser Communication and Atmospheric Propagation XXIX. SPIE. 2017;10096:374-385.

[9] Achtelik MC, Stumpf J, Gurdan D, Doth KM. Design of a Flexible High Performance Quadcopter Platform Breaking the Mav Endurance Record With Laser Power Beaming. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.IEEE. 2011:5166-5172.

[10] https://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf

[11] Wang D, Du Q, Zhou T, Li D, Wilcox R. Stabilization of the 81-Channel Coherent Beam Combination Using Machine Learning. Optics Exp. 2021;29:5694-709.

[12] Guo Y, Zhong L, Min L, Wang J, Wu Y, Chen K, Wei K, Rao C. Adaptive Optics Based on Machine Learning: A Review. Opto-Electron. Adv. 2022;5:200082.

[13] Lohani S, Glasser RT. Turbulence Correction With Artificial Neural Networks. Optics letters. 2018;43:2611-2614.

[14] Mills B, Grant-Jacob JA, Praeger M, Eason RW, Nilsson J, et al. Single-Step Phase Optimisation for Coherent Beam Combination Using Deep Learning. Scientific Reports. 2022;12:5188.

[15] Gu H, Liu M, Liu H, Yang X, Liu W. An Algorithm Combining Convolutional Neural Networks With SPGD for SLAO in FSOC. Optics Communications. 2020;475:126243.

[16] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, et al. Human-Level Control Through Deep Reinforcement Learning. Nature. 2015;518:529-33.

[17] Lötzsch W. Using deep reinforcement learning for the continuous control of robotic arms. 2018. arXiv preprint: https://arxiv.org/pdf/1810.06746.pdf

[18] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. 2015.arXiv preprint: https://arxiv.org/pdf/1509.02971.pdf

[19] Bellman R.E. *Dynamic Programming*, Dover. 2003. ISBN 0-486-42809-5.

[20] Ke H, Xu B, Xu Z, Wen L, Yang P, et al. Self-Learning Control for Wavefront Sensorless Adaptive Optics System Through Deep Reinforcement Learning. Optik. 2019;178:785-793.

[21] Rytov SM, Kravtsov IA, Tatarskiĭ VI. Principles of Statistical Radiophysics: Wave Propagation Through Random Media. Berlin: Springer; 1989.

[22] Spall JC. Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization. IEEE Transactions on aerospace and electronic systems. 1998;34:817-823.

[23] Yenice ZD, Adhikari N, Wong YK, Aksakalli V, Gumus AT, Abbasi B. SPSA-FSR: simultaneous perturbation stochastic approximation for feature selection and ranking. 2018. arXiv preprint: https://arxiv.org/pdf/1804.05589.pdf

[24] Khan A, Sohail A, Zahoora U, Qureshi AS. A Survey of the Recent Architectures of Deep Convolutional Neural Networks. Artif Intell Rev. 2020;53:5455-516.

[25] https://mitpress.mit.edu/9780262035613/

[26] Cho K, Van Merriënboer B, Bahdanau D, Bengio Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. 2014. arXiv preprint: https://arxiv.org/pdf/1409.1259.pdf

[27] Rumelhart DE, Hinton GE, Williams RJ. Learning Representations by Back-Propagating Errors. Nature. 1986;323:533-536.

[28] Kingma D, Ba J. Adam: A Method for Stochastic Optimization In: Proceedings of the 3rd International Conference for Learning Representations (ICLR'15). San Diego. 2015;500.

[29] Von F Z. Beugungstheorie Des Schneidenver-Fahrens UND Seiner Verbesserten Form, Der Phasenkontrastmethode. physica. 1934;1:689-704.

[30] https://www.python.org/

[31] https://www. tensorflow.org/.

[32] Fleck JA, Morris JR, Feit, M.D. *Time-Dependent Propagation of High-Energy Laser Beams Through the Atmosphere*, Appl. Phys. 1976;10:129-160.

[33] Schmidt JD. Numerical Simulation of Optical Wave Propagation with Examples in MATLAB. SPIE.2010:PM199.

[34] Vorontsov AM, Paramonov PV, Valley MT, Vorontsov MA. Generation of Infinitely Long Phase Screens for Modeling of Optical Wave Propagation in Atmospheric Turbulence. Waves in Random and Complex Media. 2008;18:91-108.

[35] https://www.optonicus.com/wonat-software/