An alternative approach to train neural networks using monotone variational inequality

Chen Xu [1] Xiuyuan Cheng [2] Yao Xie [1]

[1]Georgia Institute of Technology [2]Duke University

Motivation and contribution

Neural network training is a highly non-convex and non-linear optimization problem. Thus, theoretical understanding of the training process [2, 4], which is predominantly based on the gradient of the loss objective with respect to parameters, remains limited, especially in providing performance guarantees of testing performance.

The current work investigates an alternative approach to neural network training by reducing to another problem with convex structure — to solve a monotone variational inequality (MVI) — inspired by a recent work of [1]. We propose a practical and completely general algorithm called *stochastic variational inequality* (`SVI`), and demonstrate its applicability in training fully-connected neural networks, graph neural networks (GNN), and convolutional networks (CNN); `SVI` is completely general for training other networks. In special cases, we obtain performance guarantee of ℓ_2 and ℓ_∞ bounds on model recovery and prediction accuracy.

Problem setup

Suppose the conditional expectation of Y is modeled by an L-layer neural network $G(X,\theta)$:

$$\mathbb{E}[Y|X,\theta] = G(X,\theta) = \phi_L(g_L(X_L,\theta_L)), \tag{1}$$

where $X_L = \phi_{L-1}(g_{L-1}(X_{L-1},\theta_{L-1})), X_1 = X$ denote the nonlinear feature transformation from the previous layer, $\theta = \{\theta_1,\ldots,\theta_L\}$ denotes model parameters, and each ϕ_l denotes the non-decreasing activation function at layer l.

Assume there exists θ^* so that $\mathbb{E}[Y|X] = G(X,\theta^*)$. To train the network (i.e., estimate the parameters), it is common to use empirical risk minimization. First, a loss function $\mathcal{L}$ is used to measure the difference between predictions of (1) and true training data Y. Then, gradient-based methods are typically adopted to gradually update estimates of parameters θ.

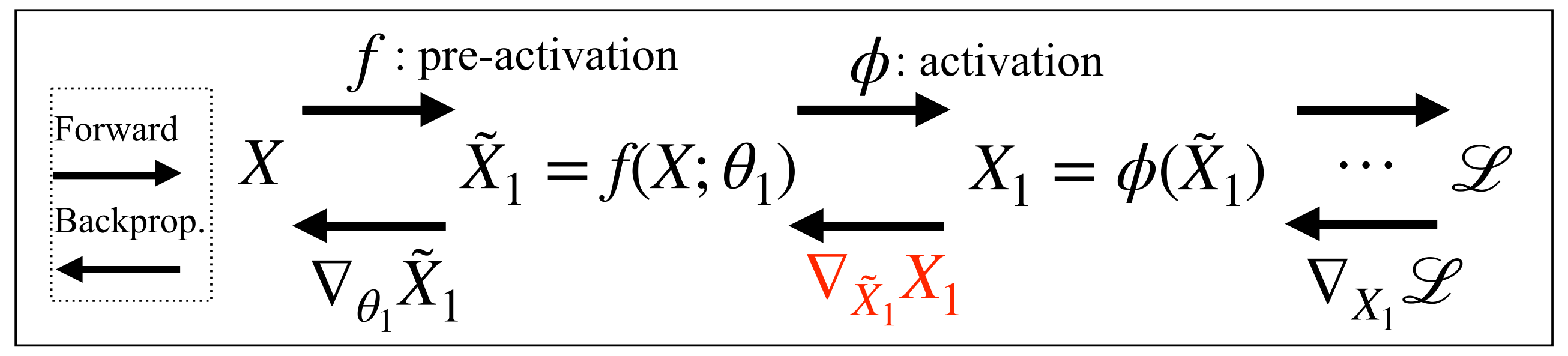
Our approach

Our `SVI` applies to *any* form of $g_l(X_l,\theta_l)$ in (1) for $l = 1,\ldots,L$. In particular, `SVI` provides alternative descent directions that are not the gradient of loss objective with respect to parameters. In short, it "skips" computing derivatives of the point-wise non-linearities with respect to inputs. Figure 1 illustrates and compares `SVI` with gradient-based methods. Algorithm 1 provides the complete procedure for adopting `SVI` in network training.

Algorithm 1 Stochastic variational inequality (`SVI`).

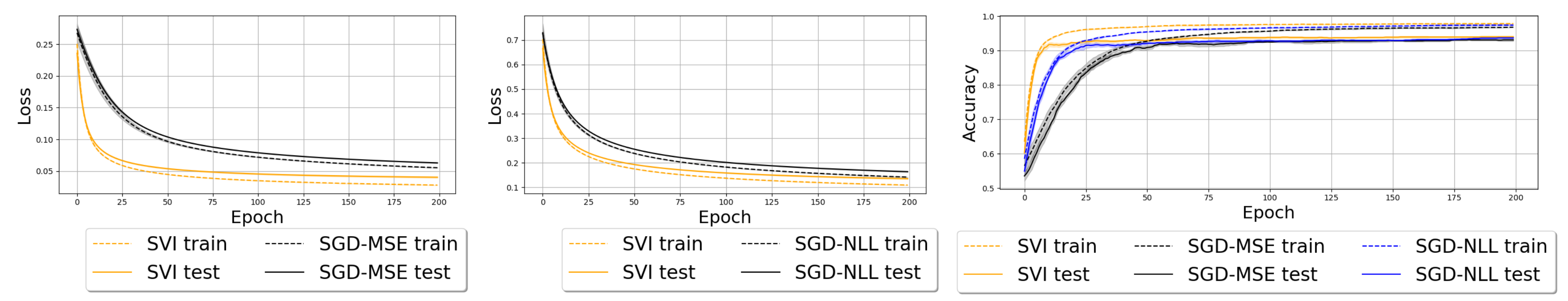
```
Require: Inputs (a) Training data ${(X_i,Y_i)}_{i=1}^N$, (b) An $L$-layer network $G(X,\theta) = \{\phi_l(g_l(X_l,\theta_l))\}_{l=1}^L$ as in (1), (c) Loss function $\mathcal{L} = \mathcal{L}(G(X,\theta),Y)$, (d) Learning rate $\gamma > 0$.
Ensure: Trained model parameters $\hat\theta = \{\hat\theta_l\}_{l=1}^L$
 1: Initialize the network with some $\hat\theta$.
 2: while Training do
 3:     Store $\{(\tilde X_{l+1},X_{l+1})\}_{l=1}^L$ in forward pass on data, where $\tilde X_{l+1} = g_l(X_l,\hat\theta_l)$ and $X_{l+1} = \phi_l(\tilde X_{l+1})$.
 4:     Obtain $\{\nabla_{X_{l+1}}\mathcal{L}\}_{l=1}^L$.
 5:     for Layer $l = 1,\ldots,L$ do
 6:         Compute the surrogate loss $\tilde{\mathcal{L}}_l = \text{sum}(\tilde X_{l+1} \odot \nabla_{X_{l+1}}\mathcal{L})$.
 7:         Obtain $F_l(\hat\theta_l) = \nabla_{\hat\theta_l}\tilde{\mathcal{L}}_l$.
 8:     end for
 9:     Update $\hat\theta_l = \hat\theta_l - \eta F_l(\hat\theta_l)$ for $l = 1,\ldots,L$.
10: end while
```

There are several practical benefits of `SVI`: First, it is easy to implement and directly leverages the benefit of automatic differentiation [3]. We implement the skipping idea via backpropagating another loss $\tilde{\mathcal{L}}(\theta_l)$ (see Alg. 1, line 4) on the parameters, which is simple and efficient to compute. Second, `SVI` barely differs in terms of computational cost against gradient-based methods. Third, it is generalizable by allowing arbitrary form of $g_l(X_l,\theta_l)$ and loss function $\mathcal{L}$. Lastly, `SVI` is very flexible as one can apply `SVI` to all or a subset of layers in the network.



$$\Delta^{GD}\theta_1 = \nabla_{X_1}\mathscr{L} \circ \nabla_{\tilde X_1}X_1 \circ \nabla_{\theta_1}\tilde X_1 \quad \text{(Backprop.)}$$
$$\Delta^{SVI}\theta_1 = \nabla_{X_1}\mathscr{L} \circ \nabla_{\theta_1}\tilde X_1 \quad \text{(Backprop. with skipping)}$$

Figure 1. Gradient descent (GD) vs. `SVI`: the difference appears in the skipping of differentiation with respect to the point-wise non-linearity ϕ. Details are in Algorithm 1.



(a) Losses under MSE objective (b) Losses under NLL objective (c) Accuracies on the same test data

Figure 2. One-layer FC model training. In (a) and (b), we plot training and test losses for both SGD (black) and `SVI` (orange). The loss choice for SGD is the mean-squared error (MSE) in (a) and the negative log-likelihood (NLL) in (b). In (c), we show the prediction accuracies by `SVI` in orange, SGD by MSE (SGD-MSE) in black, and SGD by NLL (SGD-NLL) in blue.

Theoretical Analysis

Our guarantees on model recovery only apply for the *last-layer training* when $g_L(X_L,\theta_L) = \eta_L(X_L)\theta_L$ in (1) and previous layers are known/approximated well; in practice, this can be understood as if previous layers have provided necessary feature extraction. Obtaining theoretical guarantees for multi-layer training with MVI can be very difficult and is in progress. We provide informal statements below.

Theorem (Prediction error bound for model recovery)

We have for a given test signal $X_t, t > N$ that for $p \in [2,\infty]$,

$$\mathbb{E}_{\widehat\theta_L}\{\|\widehat{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]\|_p\} \in \mathcal{O}(1/T), \tag{2}$$
$$\mathbb{E}_{\widehat\theta_L}\{\{\|\mathbb{E}_{X_t}\{\widehat{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]\}\|_p\} \in \mathcal{O}(1/\sqrt{T}), \tag{3}$$

where $\widehat{\mathbb{E}}[Y_t|X_t]$ is prediction using the estimator $\widehat\theta_L$ after T training iterations. One obtains (2) (resp. (3)) if the monotone operator at the last layer L (i.e., $\nabla_{\theta_l}\tilde{\mathcal{L}}(\theta_l)$ in Algorithm 1 at $l = L$ under the MSE loss $\mathcal{L}$) is strongly monotone (resp. monotone).

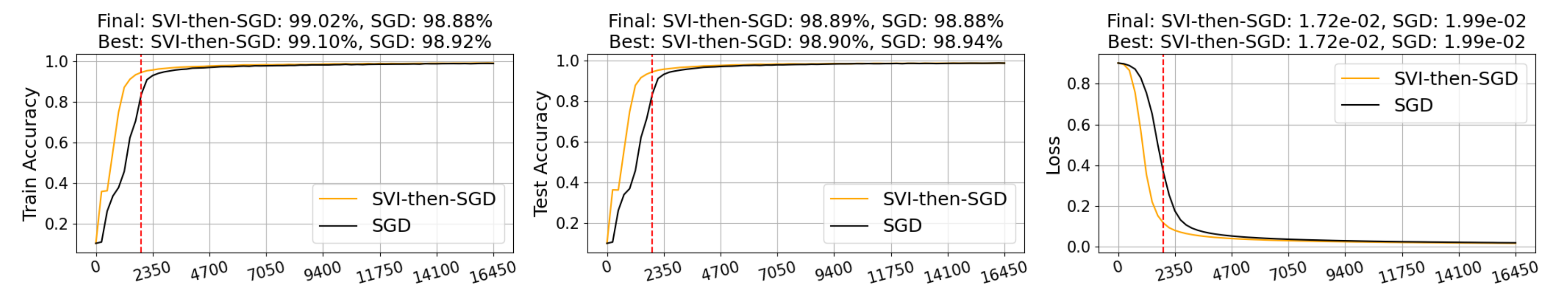
Note that the guarantee is point-wise. Meanwhile, the proof only requires access to an unbiased estimator of the monotone operator $\nabla_{\theta_l}\tilde{\mathcal{L}}(\theta_l)$, so it is independent of the size of mini-batches. Nevertheless, using larger batches further reduces variances of the estimators.

Numerical results

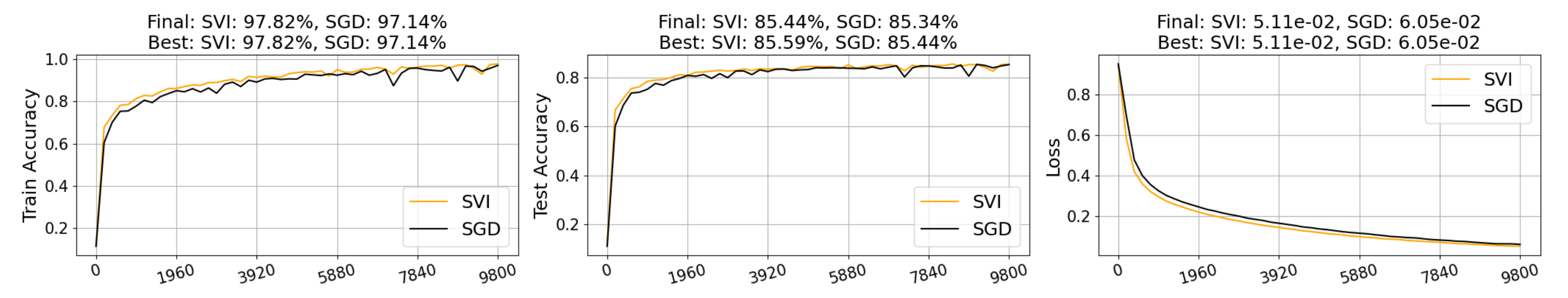
We compare `SVI` with SGD on simulated data (Figure 2) and real-data (Table 3 and Figure 4). We note that `SVI` almost always yields faster initial convergence as gradient-based methods, which likely happens as a result of the "skipping" procedure that induces further neuron displacement after training . Also, `SVI` reaches competitive final performance, especially against SGD baselines.

Table 1: Classification accuracies on the large `ogbn-arxiv` dataset under varying sizes of the GCN. "Initial" (resp. "Final") results indicate prediction accuracies after training 100 (resp. 1000) epochs. Entries in bracket show standard deviation over three trials

# hidden neurons	result type	SVI Train	SVI Valid	SVI Test	SGD Train	SGD Valid	SGD Test
128	Initial	39.64 (1.99)	39.52 (1.84)	39.83 (1.95)	6.95 (4.36)	7.05 (4.37)	6.91 (4.31)
	Final	63.55 (0.25)	63.44 (0.26)	63.47 (0.23)	51.62 (2.22)	51.38 (2.15)	51.63 (2.29)
256	Initial	52.02 (0.95)	51.84 (1.04)	52.02 (1.01)	23.38 (3.86)	23.35 (3.9)	23.43 (3.86)
	Final	66.56 (0.08)	66.2 (0.13)	66.26 (0.09)	59.24 (1.56)	59.14 (1.59)	59.1 (1.48)
512	Initial	57.88 (0.36)	57.57 (0.39)	57.68 (0.42)	33.55 (2.42)	33.46 (2.58)	33.64 (2.46)
	Final	69.12 (0.13)	68.52 (0.07)	68.72 (0.07)	64.28 (0.77)	63.99 (0.71)	64.07 (0.88)



(a) MNIST by LeNet: training accuracy (left), test accuracy (middle), and training loss (right).



(b) CIFAR10 by VGG-16: training accuracy (left), test accuracy (middle), and training loss (right).

Figure 4. Classification accuracies and training losses. We plot the metrics over training batches in each sub-figure. In the title, "Final" represents the metric at the end of all training epochs and "Best" represents the highest/lowest metric throughout training epochs. On MNIST, training before the dashed dotted red line is by `SVI`, and we continue train the SVI-warm-started model afterwards by SGD.

Future directions

We wish to test the performance of `SVI` on training larger models on larger datasets. Theoretically, we wish to extend the analyses beyond last-layer training.

References

[1] Anatoli B. Juditsky and Arkadi S. Nemirovsky. Signal recovery by stochastic optimization. *Autom. Remote. Control.*, 80:1878–1893, 2019.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[3] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[4] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.