Eur. Phys. J. C (2016) 76:235
DOI 10.1140/epjc/s10052-016-4099-4

THE EUROPEAN PHYSICAL JOURNAL C

Regular Article - Experimental Physics

Parameterized neural networks for high-energy physics

Pierre Baldi[1], **Kyle Cranmer**[2], **Taylor Faucett**[3], **Peter Sadowski**[1], **Daniel Whiteson**[3,a]

[1] Department of Computer Science, University of California, Irvine, CA 92697, USA
[2] Department of Physics, NYU, New York, NY, USA
[3] Department of Physics and Astronomy, University of California, Irvine, CA 92697, USA

Received: 8 February 2016 / Accepted: 19 April 2016 / Published online: 27 April 2016

Abstract We investigate a new structure for machine learning classifiers built with neural networks and applied to problems in high-energy physics by expanding the inputs to include not only measured features but also physics parameters. The physics parameters represent a smoothly varying learning task, and the resulting parameterized classifier can smoothly interpolate between them and replace sets of classifiers trained at individual values. This simplifies the training process and gives improved performance at intermediate values, even for complex problems requiring deep learning. Applications include tools parameterized in terms of theoretical model parameters, such as the mass of a particle, which allow for a single network to provide improved discrimination across a range of masses. This concept is simple to implement and allows for optimized interpolatable results.

1 Introduction

Neural networks have been applied to a wide variety of problems in high-energy physics [1,2], from event classification [3,4] to object reconstruction [5,6] and triggering [7,8]. Typically, however, these networks are applied to solve a specific isolated problem, even when this problem is part of a set of closely related problems. An illustrative example is the signal-background classification problem for a particle with a range of possible masses. The classification tasks at different masses are related, but distinct. Current approaches require the training of a set of isolated networks [9,10], each of which are ignorant of the larger context and lack the ability to smoothly interpolate, or the use of a single signal sample in training [11,12], sacrificing performance at other values.

In this paper, we describe the application of the ideas in Ref. [13] to a new neural network strategy, a *parameterized neural network* in which a single network tackles the full set of related tasks. This is done by simply extending the list of input features to include not only the traditional set of event-level features but also one or more *parameters* that describe the larger scope of the problem such as a new particle's mass. The approach can be applied to any classification algorithm; however, neural networks provide a smooth interpolation, while tree-based methods may not.

A single parameterized network can replace a set of individual networks trained for specific cases, as well as smoothly interpolate to cases where it has not been trained. In the case of a search for a hypothetical new particle, this greatly simplifies the task – by requiring only one network – as well as making the results more powerful – by allowing them to be interpolated between specific values. In addition, they may outperform isolated networks by generalizing from the full parameter-dependent dataset.

In the following, we describe the network structure needed to apply a single parameterized network to a set of smoothly related problems and demonstrate the application for theoretical model parameters (such as new particle masses) in a set of examples of increasing complexity.

2 Network structure and training

A typical network takes as input a vector of features, $\bar{x}$, where the features are based on event-level quantities. After training, the resulting network is then a function of these features, $f(\bar{x})$. In the case that the task at hand is part of a larger context, described by one or more parameters, $\bar{\theta}$. It is straightforward to construct a network that uses both sets of inputs, $\bar{x}$ and $\bar{\theta}$, and operates as a function of both: $f(\bar{x}, \bar{\theta})$. For a given set of inputs $\bar{x}_0$, a traditional network evaluates to a real number $f(\bar{x}_0)$. A parameterized network, however, provides a result that is parameterized in terms of $\bar{\theta}$: $f(\bar{x}_0, \bar{\theta})$, yielding different output values for different choices of the parameters $\bar{\theta}$; see Fig. 1.

Training data for the parameterized network has the form $(\bar{x}, \bar{\theta}, y)_i$, where y is a label for the target class. The addi-

[a] e-mail: daniel@uci.edu

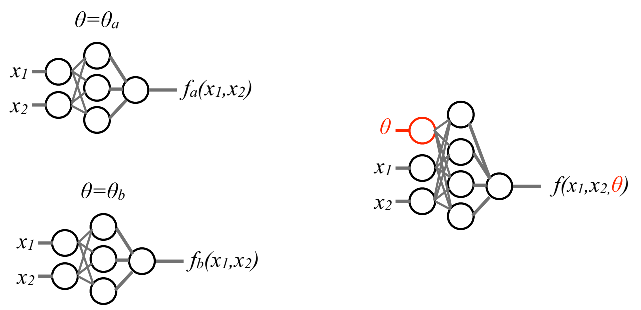

Fig. 1 *Left*, individual networks with input features (x_1, x_2), each trained with examples with a single value of some parameter $\theta = \theta_a, \theta_b$. The individual networks are purely functions of the input features. Performance for intermediate values of θ is not optimal nor does it necessarily vary smoothly between the networks. *Right*, a single network trained with input features (x_1, x_2) as well as an input parameter θ; such a network is trained with examples at several values of the parameter θ

tion of $\bar{\theta}$ introduces additional considerations in the training procedure. While traditionally the training only requires the conditional distribution of $\bar{x}$ given $\bar{\theta}$ (which is predicted by the theory and detector simulation), now the training data has some implicit prior distribution over $\bar{\theta}$ as well (which is arbitrary). When the network is used in practice it will be to predict y conditional on both $\bar{x}$ and $\bar{\theta}$, so the distribution of $\bar{\theta}$ used for training is only relevant in how it affects the quality of the resulting parameterized network – it does not imply that the resulting inference is Bayesian. In the studies presented below, we simply use equal sized samples for a few discrete values of $\bar{\theta}$. Another issue is that some or all of the components of $\bar{\theta}$ may not be meaningful for a particular target class. For instance, the mass of a new particle is not meaningful for the background training examples. In what follows, we randomly assign values to those components of $\bar{\theta}$ according to the same distribution used for the signal class. In the examples studied below, the networks have enough generalization capacity and the training sets are large enough that the resulting parameterized classifier performs well without any tuning of the training procedure. However, the robustness of the resulting parameterized classifier to the implicit distribution of $\bar{\theta}$ in the training sample will in general depend on the generalization capacity of the classifier, the number of training examples, the physics encoded in the distributions $p(\bar{x}|\bar{\theta}, y)$, and how much those distributions change with $\bar{\theta}$.

3 Toy example

As a demonstration for a simple toy problem, we construct a parameterized network which has a single input feature x and a single parameter θ. The network, with one hidden layer of three nodes and sigmoid activation functions, is trained using labeled examples where examples with label 0 are drawn from a uniform background and examples with label 1 are drawn from a Gaussian with mean θ and width $\sigma = 0.25$. Training samples are generated with $\theta = -2, -1, 0, 1, 2$; see Fig. 2a.

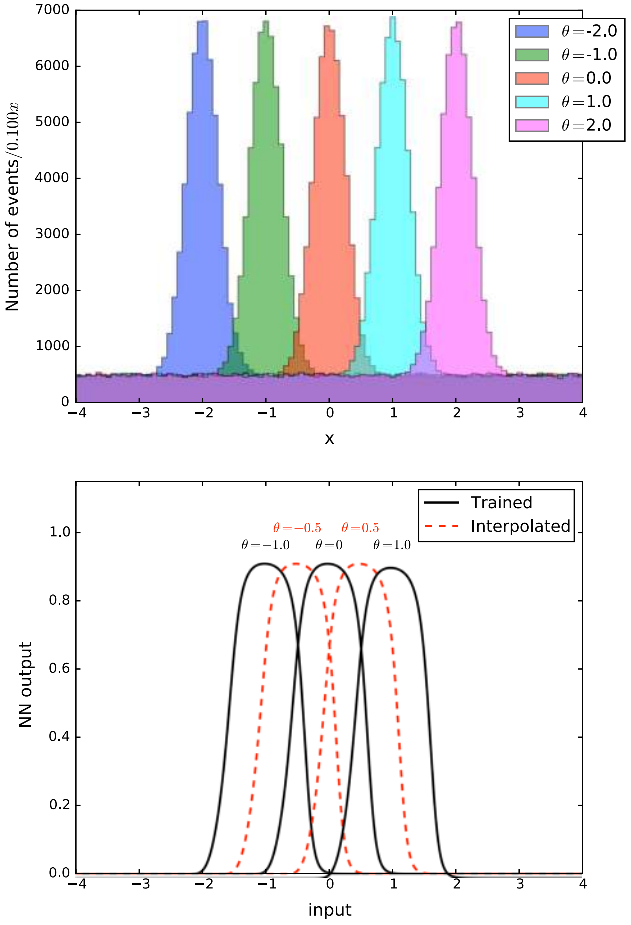

Fig. 2 *Top* training samples in which the signal is drawn from a Gaussian and the background is uniform. *Bottom*, neural network response as a function of the value of the input feature x, for various choices of the input parameter θ; note that the single parameterized network has seen no training examples for $\theta = -1.5, -0.5, 0.5, 1.5$

As shown in Fig. 2, this network generalizes the solution and provides reasonable output *even for values of the parameter where it was given no examples*. Note that the response function has the same shape for these values ($\theta = -1.5, -0.5, 0.5, 1.5$) as for values where training data was provided, indicating that the network has successfully parameterized the solution. The signal-background classification accuracy is as good for values where training data exist as it is for values where training data does not.

4 1D physical example

A natural physical case is the application to the search for new particle of unknown mass. As an example, we consider the search for a new particle X which decays to $t\bar{t}$. We treat the

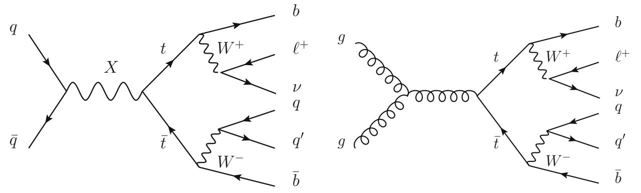

Fig. 3 Feynman diagrams showing the production and decay of the hypothetical particle $X \rightarrow t\bar{t}$, as well as the dominant standard model background process of *top* quark pair production. In both cases, the $t\bar{t}$ pair decay to a single charged lepton (ℓ), a neutrino (ν) and several quarks (q, b)

most powerful decay mode, in which $t\bar{t} \rightarrow W^+bW^-\bar{b} \rightarrow qq'b\ell\nu\bar{b}$. The dominant background is standard model $t\bar{t}$ production, which is identical in final state but distinct in kinematics due to the lack of an intermediate resonance. Figure 3 shows diagrams for both the signal and background processes.

We first explore the performance in a one-dimensional case. The single event-level feature of the network is m_{WWbb}, the reconstructed resonance mass, calculated using techniques described in Ref. [14]. Specifically, we assume resolved top quarks in each case, for simplicity. Events are simulated at the parton level with MADGRAPH5 [15], using PYTHIA [16] for showering and hadronization and DELPHES [17] with the ATLAS-style configuration for detector simulation. Figure 4a shows the distribution of reconstructed masses for the background process as well as several values of m_X, the mass of the hypothetical X particle. Clearly the nature of the discrimination problem is distinct at each mass, though similar across masses.

In a typical application of neural networks, one might consider various options:

- Train a single neural network at one intermediate value of the mass and use it for all other mass values as was done in Refs. [11,12]. This approach gives the best performance at the mass used in the training sample, but performance degrades at other masses.
- Train a single neural network using an unlabeled mixture of signal samples and use it for all other mass values. This approach may reduce the loss in performance away from the single mass value used in the previous approach, but it also degrades the performance near that mass point, as the signal is smeared.
- Train a set of neural networks for a set of mass values as done in Refs. [9,10]. This approach gives the best signal-background classification performance at each of the trained mass values. However, performance degrades for mass values away from the ones used in training. Most importantly, this approach leads to discontinuities in selection efficiencies across masses, and interpolation of the observed limits is not possible, as the degradation of the performance away from the training points is not defined.

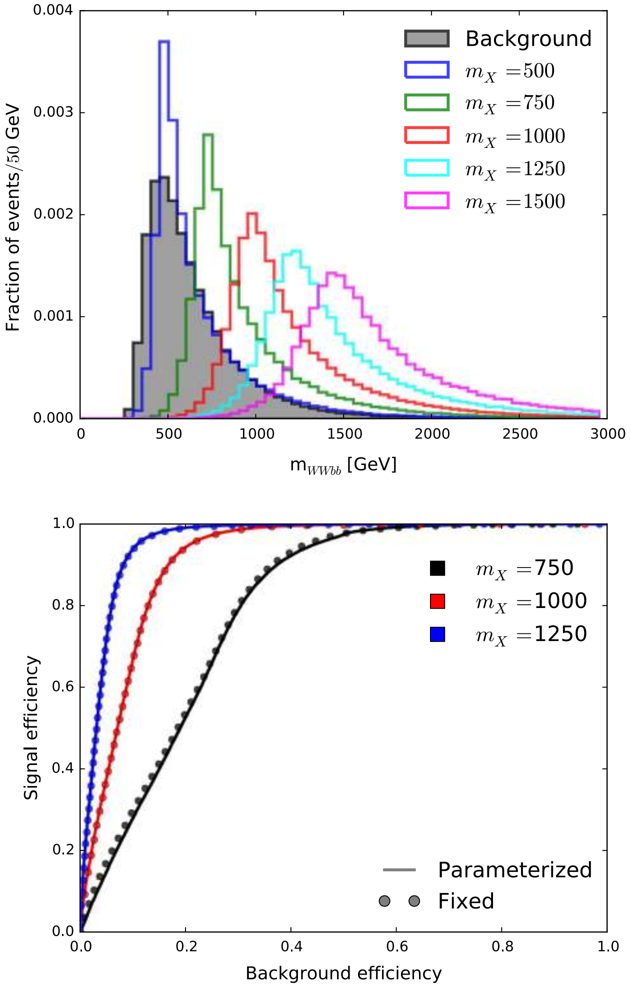

Fig. 4 *Top* distributions of neural network input m_{WWbb} for the background and two signal cases. *Bottom*, ROC curves for individual fixed networks as well as the parameterized network evaluated at the true mass, but trained only at other masses

In contrast, we train a single neural network with an additional parameter, the true mass, as an input feature. For a learning task with n event-level features and m parameters, one can trivially reconcieve this as a learning task with $n + m$ features. Evaluating the network requires supplying the set of event-level features as well as the desired values of the parameters.

We note that Ref. [18] previously applied a similar idea with the same goal of improving the interpolation among model parameters. However, in that study the application of BDTs led to a marked decrease in sensitivity at each point compared to isolated algorithms at specific values, and no demonstration was made of the ability to interpolate complex problems in high-dimensional spaces.

Our parameterized neural networks are implemented using the multi-layer perceptron in PyLearn2 [19], with outputs treated with a regressor method and logistic activation function. Input and output data are subject to preprocessing via a scikit-learn [20] pipeline (i.e. transformation to inputs/outputs with a minimum and maximum of zero and one, respectively). Each neural network is trained with 1 hidden layer of three nodes and using Nesterov's method for stochastic gradient descent [21]. Learning rates were initiated at 0.01, learning momentum was set to 0.9, and minibatch size is set to treat each point individually (i.e. minibatch size of 1). The training samples have approximately 100k examples per mass point.

The critical test is the signal-background classification performance. To measure the ability of the network to perform well at interpolated values of the parameter – values at which it has seen no training data – we compare the performance of a single fixed network trained at a specific value of m_X^0 to a parameterized network trained at the other available values other than m_X^0. For example, Fig. 4 compares a single network trained at $m_X^0 = 750$ GeV to a parameterized network trained with data at $m_X = 500, 1000, 1250, 1500$ GeV. The parameterized network's input parameter is set to the true value of the mass m_X^0, and it is applied to data generated at that mass; recall that it saw no examples at this value of m_X^0 in training. Its performance matches or nearly matches that of the single network trained at that value, validating the ability of the single parameterized network to interpolate between mass values without any appreciable loss of statistical performance. Clearly, however, such arguments cannot be applied to extrapolation beyond the boundaries of the training examples. Moreover, we recommend similar hold-out tests be performed to check the quality of the parameterized network on a case-by-case basis.

Here we focus on the performance of the parameterized classifier itself. In order to perform a statistical test at an intermediate value of m_X one will also need to know the distribution of the neural network output at that parameter point, i.e. $p(f(x, m_X)|m_X)$. The issue of *parametrized calibration* is discussed in more detail in Ref. [13]. Of course, this issue also applies to the case of a fixed network. In both cases, a straightforward, but computationally expensive strategy is to generate signal samples for each value of m_X that will be tested. An approximate, but more computationally efficient strategy is to use an interpolation algorithm to construct the parametrized distribution [22–24]. This is common practice when parameterizing the distributions with respect to nuisance parameters that describe systematic uncertainties. We advocate using hold-out to test the quality of the parametrized calibration in these situations as well.

5 High-dimensional physical example

The preceding examples serve to demonstrate the concept in one-dimensional cases where the variation of the output on both the parameters and features can be easily visualized. In this section, we demonstrate that the parameterization of the problem and the interpolation power that it provides can be achieved also in high-dimensional cases.

We consider the same hypothetical signal and background process as above, but now expand the set of features to include both low-level kinematic features which correspond to the result of reconstruction algorithms, and high-level features, which benefit from the application of physics domain knowledge. The low-level features are the four-vectors of the reconstructed events, namely:

- the leading lepton momenta,
- the momenta of the four leading jets,
- the b-tagging information for each jet
- the missing transverse momentum magnitude and angle
- the number of jets

for a total of 22 low-level features; see Fig. 5. The high-level features combine the low-level information to form approximate values of the invariant masses of the intermediate objects. These are:

- the mass ($m_{\ell\nu}$) of the $W \to \ell\nu$,
- the mass (m_{jj}) of the $W \to qq'$,
- the mass (m_{jjj}) of the $t \to Wb \to bqq'$,
- the mass ($m_{j\ell\nu}$) of the $t \to Wb \to \ell\nu b$,
- the mass (m_{WWbb}) of the hypothetical $X \to t\bar{t}$,

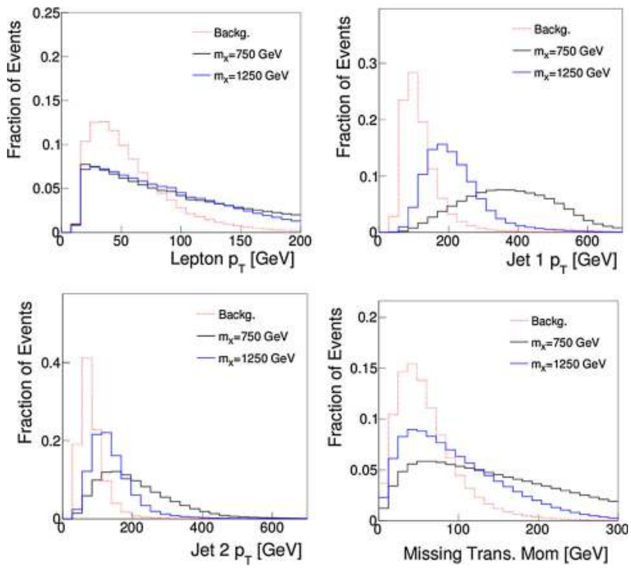

Fig. 5 Distributions of some of the low-level event features for the decay of $X \to t\bar{t}$ with two choices of m_X as well as the dominant background process

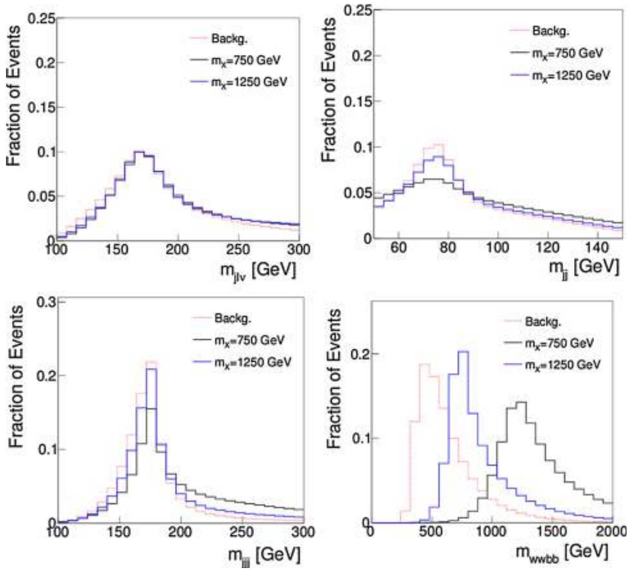

Fig. 6 Distributions of high-level event features for the decay of $X \to t\bar{t}$ with two choices of m_X as well as the dominant background process; see text for definitions

for a total of five high-level features; see Fig. 6.

The parameterized deep neural network models were trained on GPUs using the Blocks framework [25–27]. Seven million examples were used for training and one million were used for testing, with 50 % background and 50 % signal. The architectures contain five hidden layers of 500 hidden rectified linear units with a logistic output unit. Parameters were initialized from a Gaussian distribution with mean zero and width 0.1, and updated using stochastic gradient descent with mini-batches of size 100 and 0.5 momentum. The learning rate was initialized to 0.1 and decayed by a factor of 0.89 every epoch. Training was stopped after 200 epochs.

The high dimensionality of this problem makes it difficult to visually explore the dependence of the neural network output on the parameter m_X. However, we can test the performance in signal-background classification tasks. We use three types of networks. A single parameterized network is trained using 7M training samples with masses $m_X = 500, 750, 1000, 1250, 1500$ GeV and tested in a sample generated with $m_X = 1000$ GeV; the performance is compared to a single fixed network trained with samples at $m_X = 1000$ (with 7M training examples). In each case, we use approximately the same number of training and testing examples per mass point. Figure 7 shows that the parameterized network matches the performance of the fixed network. A more stringent follow-up test removes the $m_X = 1000$ sample from the training set of the parameterized network, so that this network is required to interpolate its solution. The performance is unchanged, demonstrating that the parameterized network is capable of generalizing the solution even in a high-dimensional example.

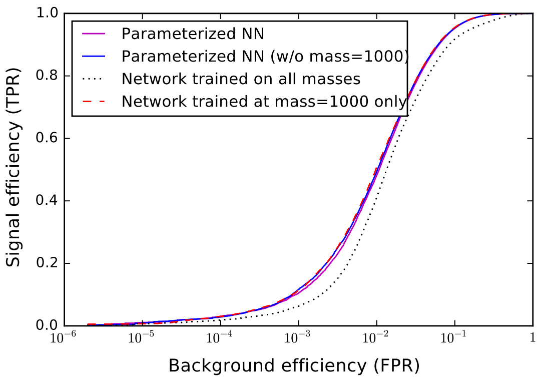

Fig. 7 Comparison of the signal-to-background discrimination for four classes of networks for a testing sample with $m_X = 1000$ GeV. A parameterized network trained on a set of masses ($m_X = 500, 750, 1000, 1250, 1500$) is compared to a single network trained at $m_X = 1000$ GeV. The performance is equivalent. A second parameterized network is trained only with samples at $m_X = 500, 750, 1250, 1500$, forcing it to interpolate the solution at $m_X = 1000$ GeV. Lastly, a single non-parameterized network trained with all the mass points shows a reduced performance. The results are indistinguishable for cases where the networks use only low-level features (shown) or low-level as well as high-level features (not shown, but identical)

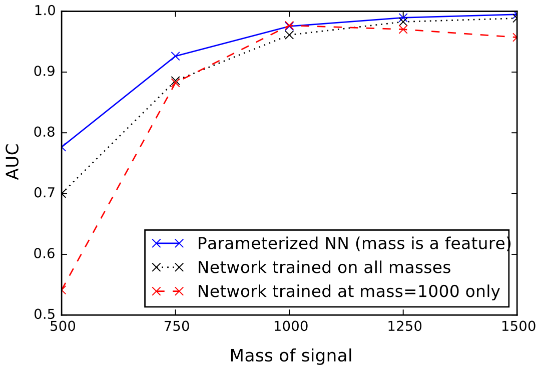

Fig. 8 Comparison of the performance in the signal-background discrimination for the parameterized network, which learns the entire problem as a function of mass, and a single network trained only at $m_X = 1000$ GeV. As expected, the AUC score (integral of the *curves* in Fig. 7) decreases for the single network as the mass deviates from the value in the training sample. The parameterized network shows improvement over this performance; the trend of improving AUC versus mass reflects the increasing separation between the signal and background samples with mass, see Figs. 5 and 6. For comparison, also shown in the performance a single network trained with an unlabeled mixture of signal samples at all masses

Conversely, Fig. 8 compares the performance of the parameterized network to a single network trained at $m_X = 1000$ GeV when applied across the mass range of interest, which is a common application case. This demonstrates the loss of performance incurred by some traditional approaches and recovered in this approach. Similarly, we see that a single network trained an unlabeled mixture of signal samples from all masses has reduced performance at each mass value tested.

In previous work, we have shown that deep networks such as these do not require the addition of high-level features [28,29] but are capable of learning the necessary functions directly from the low-level four-vectors. Here we extend that by repeating the study above without the use of the high-level features; see Fig. 7. Using only the low-level features, the parameterized deep network achieves essentially indistinguishable performance for this particular problem and training sets of this size.

6 Discussion

We have presented a novel structure for neural networks that allows for a simplified and more powerful solution to a common use case in high-energy physics and demonstrated improved performance in a set of examples with increasing dimensionality for the input feature space. While these example use a single parameter θ, the technique is easily applied to higher dimensional parameter spaces.

Parameterized networks can also provide optimized performance as a function of nuisance parameters that describe systematic uncertainties, where typical networks are optimal only for a single specific value used during training. This allows statistical procedures that make use of profile likelihood ratio tests [30] to select the network corresponding to the profiled values of the nuisance parameters [13].

Datasets used in this paper containing millions of simulated collisions can be found in the UCI Machine Learning Repository [31] at http://archive.ics.uci.edu/ml/datasets/HEPMASS.

Acknowledgments We thank Tobias Golling, Daniel Guest, Kevin Lannon, Juan Rojo, Gilles Louppe, and Chase Shimmin for useful discussions. KC is supported by the US National Science Foundation Grants PHY-0955626, PHY-1205376, and ACI-1450310. KC is grateful to UC-Irvine for their hospitality while this research was initiated and the Moore and Sloan foundations for their generous support of the data science environment at NYU. We thank Yuzo Kanomata for computing support. We also wish to acknowledge a hardware grant from NVIDIA, NSF Grant IIS-1550705, and a Google Faculty Research award to PB.

References

1. B.H. Denby, Neural networks and cellular automata in experimental high-energy physics. Comput. Phys. Commun. **49**, 429–448 (1988)
2. C. Peterson, T. Rognvaldsson, L. Lonnblad, JETNET 3.0: a versatile artificial neural network package. Comput. Phys. Commun. **81**, 185–220 (1994)
3. P. Abreu et al., Classification of the hadronic decays of the Z0 into b and c quark pairs using a neural network. Phys. Lett. B **295**, 383–395 (1992)
4. H. Kolanoski, Application of artificial neural networks in particle physics. Nucl. Instrum. Meth. A **367**, 14–20 (1995)
5. C. Peterson, Track finding with neural networks. Nucl. Instrum. Meth. A **279**, 537 (1989)
6. G. Aad et al., A neural network clustering algorithm for the ATLAS silicon pixel detector. JINST **9**, P09009 (2014)
7. L. Lonnblad, C. Peterson, T. Rognvaldsson, Finding gluon jets with a neural trigger. Phys. Rev. Lett. **65**, 1321–1324 (1990)
8. H. Bruce, M. Denby, F.B. Campbell, N. Chriss, C. Bowers, F. Nesti, Neural networks for triggering. IEEE Trans. Nucl. Sci. **37**, 248–254 (1990)
9. T. Aaltonen et al., Evidence for a particle produced in association with weak bosons and decaying to a bottom–antibottom quark pair in Higgs boson searches at the Tevatron. Phys. Rev. Lett. **109**, 071804 (2012)
10. S. Chatrchyan et al., Combined results of searches for the standard model Higgs boson in pp collisions at $\sqrt{s} = 7$ TeV. Phys. Lett. B **710**, 26–48 (2012)
11. G. Aad et al., Search for $W' \rightarrow t\bar{b}$ in the lepton plus jets final state in proton–proton collisions at a centre-of-mass energy of $\sqrt{s} = 8$ TeV with the ATLAS detector. Phys. Lett. B **743**, 235–255 (2015)
12. S. Chatrchyan et al., Search for Z ' resonances decaying to $t\bar{t}$ in dilepton+jets final states in pp collisions at $\sqrt{s} = 7$ TeV. Phys. Rev. D **87**(7), 072002 (2013)
13. K. Cranmer, J. Pavez, G. Louppe. Approximating likelihood ratios with calibrated discriminative classifiers (2015). arXiv:1506.02169
14. G. Aad et al., Search for a multi-Higgs-boson cascade in $W^+W^b\bar{b}$ events with the ATLAS detector in pp collisions at $\sqrt{s} = 8$TeV. Phys. Rev. D **89**(3), 032002 (2014)
15. J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, Tim Stelzer, MadGraph 5: going beyond. JHEP **1106**, 128 (2011)
16. S. Torbjörn, M. Stephen, S. Peter. PYTHIA 6.4 physics and manual. JHEP **0605**, 026 (2006)
17. J. de Favereau et al., DELPHES 3, a modular framework for fast simulation of a generic collider experiment. JHEP **1402**, 057 (2014)
18. V.M. Abazov et al., Search for the Higgs boson in lepton, tau and jets final states. Phys. Rev. D **88**(5), 052005 (2013)
19. I.J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, Y. Bengio. Pylearn2: a machine learning research library. arXiv:1308.4214, 2013
20. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python. J Mach Learn Res **12**, 2825–2830 (2011)
21. Y. Nesterov et al., Gradient methods for minimizing composite objective function. Technical report, UCL (2007)
22. A.L. Read, Linear interpolation of histograms. Nucl. Instrum. Meth. A **425**, 357–360 (1999)
23. K. Cranmer, G. Lewis, L. Moneta, A. Shibata, W. Verkerke. HistFactory: a tool for creating statistical models for use with RooFit and RooStats (2012)
24. Max Baak, Stefan Gadatsch, Robert Harrington, Wouter Verkerke, Interpolation between multi-dimensional histograms using a new non-linear moment morphing method. Nucl. Instrum. Meth. A **771**, 39–48 (2015)
25. B. van Merrinboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, Y. Bengio. Blocks and fuel: frameworks for deep learning (2015). arXiv:1506.00619

26. F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I.J. Goodfellow, A. Bergeron, N. Bouchard, Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop (2012)
27. J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio. Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy). Oral Presentation, Austin, TX (2010)
28. P. Baldi, P. Sadowski, D. Whiteson, Searching for exotic particles in high-energy physics with deep learning. Nature Commun. **5**, 4308 (2014)
29. P. Baldi, P. Sadowski, D. Whiteson, Enhanced Higgs boson to $\tau^+\tau^-$ search with deep learning. Phys. Rev. Lett. **114**(11), 111801 (2015)
30. Glen Cowan, Kyle Cranmer, Eilam Gross, Ofer Vitells, Asymptotic formulae for likelihood-based tests of new physics. Eur. Phys. J. C **71**, 1554 (2011)
31. P. Baldi, K. Cranmer, T. Faucett, P. Sadowski, D. Whiteson. UCI machine learning repository (2015). http://archive.ics.uci.edu/ml/datasets/HEPMASS