EPJ Web of Conferences **137**, 11007 (2017) DOI: 10.1051/epjconf/201713711007
XIIth Quark Confinement & the Hadron Spectrum

Deep Learning and Bayesian Methods

Harrison B. Prosper[1,a]

[1] *Department of Physics, Florida State University, Tallahassee, FL 32306, USA*

Abstract. A revolution is underway in which deep neural networks are routinely used to solve difficult problems such as face recognition and natural language understanding. Particle physicists have taken notice and have started to deploy these methods, achieving results that suggest a potentially significant shift in how data might be analyzed in the not too distant future. We discuss a few recent developments in the application of deep neural networks and then indulge in speculation about how such methods might be used to automate certain aspects of data analysis in particle physics. Next, the connection to Bayesian methods is discussed and the paper ends with thoughts on a significant practical issue, namely, how, from a Bayesian perspective, one might optimize the construction of deep neural networks.

1 Introduction

The idea that artifacts could somehow be endowed with human-like intelligence has a long history measured in centuries. But, the term artificial intelligence (AI) was coined by John McCarthy a mere sixty years ago [1]. Since then, few would argue that the promise of AI has far exceeded what AI enthusiasts could deliver and that popular culture was overly optimistic about when general AI would become mainstream, witness the famous Stanley Kubrick and Arthur C. Clarke movie, *2001: A Space Odyssey*, in which the HAL 9000 computer is said to have become operational in 1992! Well, it is 2016 and we are only just now seeing glimpses, albeit spectacular ones, of a future characterized by the widespread use of machines capable of feats generally associated with intelligence.

It is easy to get carried away with heady thoughts of such a future, one filled with autonomous vehicles, autonomous personal assistants, predictive personalized medicine, adaptive individualized education, automatic legal document summary, etc. However, understanding of the promised feats of artificial intelligence, including the many already in routine use, is enhanced by recognizing that current applications and those in the foreseeable future are "merely" approximations of well-defined mathematical quantities, specifically, probabilities. In 1990, it was pointed out [2, 3] that neural networks trained as classifiers approximate the class probabilities

$$p(C|x) = \frac{p(x|C)\, p(C)}{\sum_C p(x|C)\, p(C)}, \tag{1}$$

where x represents data that characterize the object to be classified, $p(x|C)$ is the associated likelihood, and $p(C)$ is the prior probability of class C. Equation (1) is Bayes' theorem in which $p(C|x)$

[a] e-mail: harry@hep.fsu.edu

is the probability that an object, say an image, characterized by data x belongs to class C. Without in any way diminishing the spectacular progress that has occurred since 1990, and which has greatly accelerated since 2006, it is worth noting that much of this progress has been the development of more accurate ways to approximate equation (1), or a one-to-one function of it, and combining approximations to equation (1) in creative ways. Perhaps, one day, applications will evolve into the kind that defy simple mathematical characterizations. But we are not there yet.

In this paper, we do not discuss AI *per se*. Rather, we focus on a class of machine learning methods, namely deep neural networks (DNN), which underlies many recent successful applications of machine learning on real-world problems. Figure 1 shows a fully connected (3, 5, 4, 1) deep neural network, that is, a network with three inputs, two hidden layers of five and four nodes each and one output node. There are many variations of this basic architecture, notably convolutional

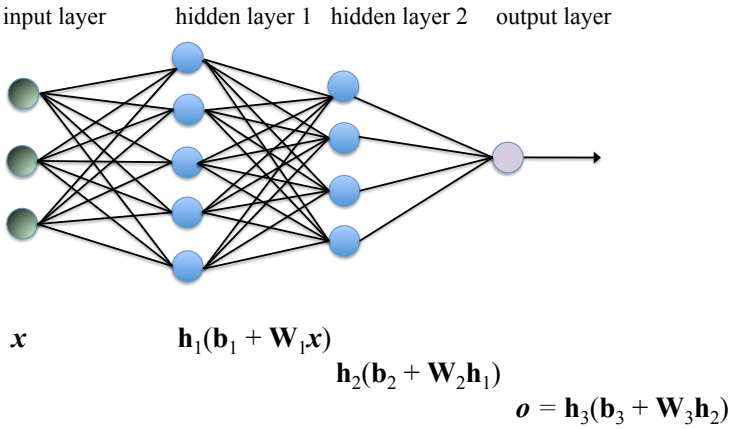

Figure 1. A fully connected deep neural network with a (3, 5, 4, 1) architecture, that is, a network with 3 inputs, $\boldsymbol{x}$, one hidden layer with 5 nodes, a second hidden layer with 4 nodes, and one node in the output layer. The elements of the column vectors $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ and the matrices $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ are free parameters. The column vectors $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, and $\boldsymbol{h}_3$ are the outputs of the two hidden layers and the output later, respectively, while their elements are defined by $h_{ij} = h_i((\mathbf{b})_j + (\mathbf{W}_i\mathbf{y})_j)$, where $h_i\,(i = 1, 2, 3)$ are functions—for example, the Rectified Linear Unit (ReLU) function $h(y) = \max(0, y)$ is a common choice for the hidden nodes, and $\mathbf{y}$ is the output vector from the previous layer.

networks that have proven their worth in applications such as image recognition (see, for example, Ref. [6]), however, in this paper we restrict our attention to fully connected networks that are used for classification.

Classification is arguably the current killer app of machine learning in that many tasks requiring a modicum of intelligence can be deconstructed into a series of classification tasks. Consider, for example, the task of transcribing text from an image, say a cellphone picture of a page from a book, to an editable document. After some transformations to remove distortions, the machine-based algorithm has to distinguish characters from other elements of the picture. This requires classification. Then, the algorithm has to identify contiguous series of characters, that is, potential words; another classification task. Next, in order to minimize transcription errors, the algorithm has to identify the language and perhaps even the font type. Yet more classification. Now that the language has been identified, the words can be identified more reliably, which, again, is a classification task. Errors can of course occur. Therefore, in addition to the multiple levels of classification, the algorithm would, ideally,

provide some measure of confidence in its classification decisions. We shall take up this important point later in the paper.

The paper is organized as follows. We start with a discussion of some recent successful applications of deep neural networks and follow with some thoughts about what kinds of applications of deep learning might be possible in experimental particle physics. We continue with a discussion of the connection between deep learning and Bayesian methods and how one might go about optimizing one important aspect of the training of deep neural networks. We end with a summary.

2 Going Deep

In 2014, machine learning enthusiasts were challenged [4] to find the best classifier of simulated signal and background events created by the ATLAS collaboration and inspired by the evidence for the process $pp \to H \to \tau^+\tau^-$ first reported by the collaboration in 2013 [5]. The competition was won, not by a physicist, but by computer scientist Gábor Melis (Franz Inc., Fixnum Services, Hungary) who created a classifier using deep neural networks. Deep in this context, as in deep learning, refers to the use of neural network models with multiple hidden layers, as illustrated in figure 1[1]. The winning classifier was the average of the outputs of 70 deep neural networks, each with architecture (35, 600, 600, 600, 2), that is, 35 inputs, 3 hidden layers of 600 nodes each, and 2 outputs, implying a classifier with more than 70 million fitted parameters.

In the early 2000s, such a computational feat seemed out of reach as most attempts to train large neural networks with more than one hidden layer ended in failure, a frustrating state of affairs blamed, in part, on the vanishing gradient problem, the tendency for the gradients needed for minimization via back-propagation [7] to become extremely small for networks with many layers that use sigmoidal functions in the hidden nodes. Interest therefore shifted away from neural networks to models such as boosted decision trees and support vector machines [7]. But in 2006, Hinton, Osindero and Teh (HOT) [8] succeeded in training a deep neural network by first initializing its parameters sequentially, layer by layer. Each layer was trained to produce a representation of its inputs that served as the training data for the next layer. The network was then tweaked using gradient descent. This breakthrough was regarded as compelling evidence that the training of deep neural networks is feasible, but requires careful initialization of parameters and sophisticated machine learning algorithms.

One way to view the HOT model is as a stack of sequentially trained auto encoders. An auto encoder is a neural network, such as the one shown in figure 2, that models the identity function $f : \boldsymbol{x} \to \boldsymbol{x}$, but with a significant twist. The basic idea is to compress the inputs into a smaller set of variables by forcing a mapping from the input layer to a hidden layer structure with fewer nodes in the layer that precedes the output layer. The output layer has the same number of nodes as inputs and the network is trained so that, ideally, the output $\boldsymbol{o} = \boldsymbol{x}$. In figure 2, the auto encoder is the subnetwork with structure (5, 3, 3), which compresses five dimensions to three. The key idea here is that the compression algorithm is learned through the training procedure, typically, stochastic gradient descent implemented using back-propagation. The success of the HOT model rekindled interest in neural networks, especially those with multiple layers. It quickly became conventional wisdom that sophistication and ingenuity were the key to the successful training of deep neural networks.

But, in 2010, Cireşan, Meier and Schmidhuber [9] published a surprising counter example that challenged the conventional wisdom. The authors trained deep neural networks to classify the handwritten digits in the MNIST data set [10], which comprises $28 \times 28 = 784$-pixel images, 60,000 for training and 10,000 for testing. Figure 3 shows the first digit from each of the ten classes in the MNIST training sample. The surprising result is that a plain deep neural network with architecture (784, 2500,

[1] Deep neural networks are essentially multilayer perceptrons reincarnated with additional features.

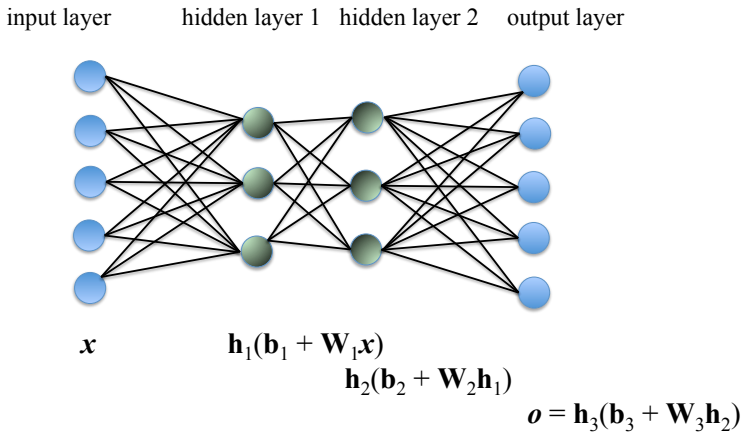

Figure 2. An auto encoder that compresses five variables to three by forcing the network to map five inputs to three hidden nodes, then from three hidden nodes back to five output nodes with the same values as the inputs.

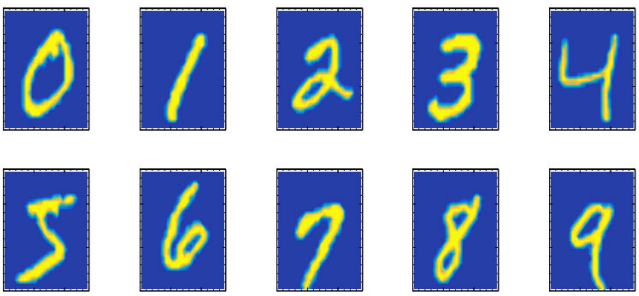

Figure 3. The first digit from each class in the MNIST training sample.

2000, 1500, 1000, 500, 10), that uses the scaled hyperbolic tangent node function ($h(y) = A \tanh By$), trained using standard back-propagation, outperformed all other methods that had been applied to the MNIST data set as of 2010. The error rate of this ~ 12 million-parameter DNN was 35 images out of 10,000. The authors also noted that of the 35 incorrectly classified images, 30 were correctly classified using the network output with the second highest output.

How is it possible to fit a 12 million-parameter model in a matter of hours, with a mere 60,000 images, avoid overfitting, and best even the most sophisticated alternative models? The answer: deploy huge quantities of computing power, in this case, graphics processing units (GPU), which can run large numbers of identical calculations in parallel, and use huge quantities of training data. In the MNIST work, the latter was brilliantly achieved by randomly distorting every one of the 60,000 images in the training set, every training epoch, thereby generating an effectively endless supply of training data. This idea was a veritable stroke of genius because it meant that the entire set of 60,000 undeformed images could be used as the validation set during training, since none were used as train-

ing data, and overfitting was avoided because the training algorithm was fed an unlimited quantity of data. We are currently investigating something similar in particle physics classification problems.

As alluded to above, before the Cireşan paper, conventional wisdom held that some combination of clever architecture, algorithms, feature engineering (machine learning-speak for finding good variables), parameter initialization, and regularization was necessary to achieve state of the art results with deep neural networks. However, the Cireşan paper and subsequent ones, such as a 2012 paper on the recognition of German traffic signs [11] in which super-human pattern recognition was achieved, show that none of this cleverness is strictly necessary in order to achieve spectacular results on difficult problems. These results have a profound implication. They imply that at some threshold, perhaps soon to be crossed as computing power and datasets grow ever larger, the cost of deploying massive amounts of computing on difficult problems will be far lower than the cost of deploying human ingenuity. Then it would make no economic sense to use people to solve such problems. This is not to say we shall become redundant, rather it is to say that in the future it is likely that our attention shall be drawn to much more interesting problems, while our synthetic assistants crunch away at hard problems rendered mundane by brute force.

3 The Automated Physicist

The goal of particle physicists during the next two decades is to find evidence of new physics, that is, patterns in data not predicted by the Standard Model (see, for example, Ref. [12]). The current approach entails deploying human intelligence to do the feature engineering, which has yielded variables such as M_{T2} [13], α_T [14], and razor [15].

Surely, variables such as these will shed light on the nature of the new physics once found. Until recently, practical considerations necessitated feature engineering by humans, inspired by the physics of the problem at hand. But, given our discussion in the previous section, is this really still necessary to discover new patterns in data? Variable like M_{T2} provide insight, but cannot possible add information beyond that already present in the observables on which it is based. This truism underlies the so-called matrix element method [16–19], an *ab initio* approximation to the full probability density over the space of 4-vectors and particle identity. This probability density necessarily contains all the information that can be extracted from the observed particles. The recent successes in solving extremely difficult problems, such as

- face recognition in photographs (e.g., Facebook),
- natural language understanding (e.g., Google, Microsoft, Apple),
- self-learning of the game Go [21] (Google),

offer potentially important lessons for the future of analysis in particle physics. One such lesson is this. In an era in which models with tens of millions of parameters can be trained in a matter of hours, and in which the goal is to discover scientifically significant discrepancies between observations and the predictions of the Standard Model, perhaps there is less need to devote as much time as we do inventing clever variables.

Indeed, a few particle physicists, notably, Daniel Whiteson and collaborators, have begun to explore the potential of deep learning in particle physics. In a recent paper [20], deep neural networks, trained with low-level observables, were used to effect a 25% reduction in the amount of data needed to achieve the same result as an optimized shallow neural network (NN) trained with the same inputs and with the same number of free parameters ($\sim 56,000$). Better results were obtained when high-level observables were added to the low-level ones, but the authors noted that one expects better

results by using deeper networks and more training data. The results discussed in the previous section strongly suggest that this is likely to be the case.

Could the era of the *automated physicist* be at hand? A few years from now, what might our tireless, friendly, automaton do on our behalf?

- Automatically determine the set of characteristics that distinguish particles from the primary vertex from those from other vertices and automatically classify particles based on this information.
- Automatically reduce particle event data, e.g., (p_T, η, ϕ) and particle identity, perhaps using auto encoding, into a smaller fixed set of numbers, say $N \sim 1600$—which may be thought of as 40×40 "pixelized image"—that can be the basis of further analysis.
- Automatically classify these "images" into two sets: those that look like simulated events and those that do not.
- Automatically construct a hyper-fast simulator by auto encoding the mapping from parton level events to reconstruction level events using all available fully simulated events.

But, if all the fun stuff is automated, won't that precipitate existential angst in the clever variable industry? Perhaps. But, after a spot of therapy, our very smart colleagues who work in this industry would come to realize that there are still lots of very hard problems to solve, such as understanding the nature of the physics that (we hope) our automated physicists will have discovered.

4 The Bayesian Connection

Deep neural networks are now the method of choice to solve really difficult problems [21], even though they seem even more inscrutable than, for example, boosted decision trees, or even shallow neural networks. Moreover, like most machine learning methods, a DNN provides, in effect, a point estimate of the function being approximated, but absent a measure of uncertainty. The absence of a measure of uncertainty could become a serious deficiency. There will be numerous circumstances in which it will be helpful to have a measure of the reliability of the decision taken by an automaton. Consider again the MNIST classifier created by Cireşan et al. The authors noted that the second highest output of their classifier was correct for 30 out of the 35 images incorrectly classified using the highest output. If a confidence level for each decision were available, we may have found that the confidence levels associated with the first and second guesses were about the same thereby indicating ambiguity in the classification of the 35 images. A self-driving car may alert the passengers that it is unsure about which of two courses of action it should take because the confidence levels associated with each are comparable. The ability to assign a confidence level or an uncertainty measure of some sort to decisions will surely become important as our reliance on AI deepens.

Some progress in that direction has occurred recently. Gal and Ghahramani [22] demonstrated that a deep neural network in which nodes are randomly dropped during training (a procedure referred to as dropout) approximates variational inference in Bayesian neural networks. Inspired by this insight, a method was suggested by the authors for assigning uncertainty measures to the outputs of deep neural networks. Their insight also suggests that Bayesian deep neural networks (BDNN) merit consideration even though they surely pose a considerable computational challenge.

4.1 Bayesian Neural Networks

Many machine learning methods, including DNN, amount to finding an optimal function $f(x, \omega^*)$ from a parameterized function class F_ω, by minimizing a cost function $C(T, \omega, \alpha)$ using some variation

of stochastic gradient descent. Here, $T = (t, x)$ denotes the training data, ω the parameters to be found by the minimizer to yield a best-fit value ω^*, and α denotes the training parameters. The quantities t are the known targets for inputs x. In the Bayesian approach to neural networks, a posterior density

$$\begin{aligned} p(\omega|T) &= \frac{p(T|\omega)\, p(\omega)}{p(T)}, \\ &= \frac{p(t|x,\omega)\, p(\omega)}{p(t|x)}, \end{aligned} \tag{2}$$

is calculated for the network parameters ω, where $p(t|x, \omega)$ is the likelihood for the training data and $p(\omega)$ is the prior density that encodes whatever constraints we wish to impose on the function class F_ω. From an abstract viewpoint, each neural network architecture simply defines a different function class.

Given the posterior density, $p(\omega|T)$, we can compute the predictive distribution

$$p(t|x, T) = \int p(t|x, \omega)\, p(\omega|T)\, d\omega, \tag{3}$$

which gives the probability density that the target is t for new input vector x and given that the network has been trained with training data T. For classification with $t = 1$ for the class of interest, e.g., the signal, and $t = 0$ for the other class, e.g, the background, equation (3) becomes

$$f(x) = \int f(x, \omega)\, p(\omega|T)\, d\omega, \tag{4}$$

with

$$\delta[f(x)] = \sqrt{\int [f(x, \omega) - f(x)]^2\, p(\omega|T)\, d\omega}, \tag{5}$$

one measure of the point-by-point uncertainty in $f(x)$. In particle physics, we refer to equation (4) as a Bayesian neural network (BNN). A Bayesian deep neural network is just a deep version of the same.

The virtue of casting the training of deep neural networks as an inference problem is that it permits a broader view of machine learning and connects it firmly to standard statistical methodology. A severe impediment, however, to a fully Bayesian approach is that the calculation of the posterior density $p(\omega|T)$ is intractable, so we must resort to approximations of which the two most common are either to

- sample $p(\omega|T)$ using Markov Chain Monte Carlo (MCMC), or
- approximate $p(\omega|T)$ with the closest tractable approximation $q(\omega|\phi)$, where ϕ are variational parameters.

Another difficulty is the need to specify the prior $p(\omega)$. In practice, we do so using a prior $p(\omega, \alpha)$ parameterized with hyper-parameters α, which have to be chosen in some way. There are at least three ways,

- trial and error,
- Bayesian optimization [23], and
- empirical Bayes [24].

In the next section, we sketch a possible way the choice of hyper-parameters might be optimized using empirical Bayes.

4.2 Optimization

In a fully Bayesian approach, the hyper-parameters would be constrained by hyper-priors, thereby building a hierarchical model. Alas, the hyper-priors themselves typically contain their own set of hyper-parameters, which need to be determined. One way to nip the hierarchy in the bud is to abandon Bayesian purity and embrace empirical Bayes. The basic idea of empirical Bayes is to replace the hyper-parameters of the prior with estimates thereof. Here is a tentative suggestion.

- Use MCMC to approximate,

$$p(\omega|T,\alpha_0) = \frac{p(T|\omega)\, p(\omega,\alpha_0)}{p(T,\alpha_0)}, \tag{6}$$

  where α_0 denotes some default choice of the hyper-parameters.
- Then, given training data T', optimize the integral

$$\begin{aligned} p(T'|T,\alpha,\alpha_0) &= \int p(T'|\omega)\, p(\omega|T,\alpha_0)\, w(\omega,\alpha,\alpha_0)\, d\omega, \\ &\approx \frac{1}{M}\sum_{i=1}^{M} p(T'|\omega_i)\; w(\omega_i,\alpha,\alpha_0), \end{aligned} \tag{7}$$

  with respect to α. The count M is the number of points (DNN functions) sampled and $w = p(\omega,\alpha)/p(\omega,\alpha_0)$ is a known weighting function.

Of course, none of this would work if the sampling of the posterior density for a DNN with millions of parameters proves to be infeasible. One way to render the above feasible, if we are prepared to sample billions of points, is to abandon MCMC and sample from the prior, which can be trivially parallelized. While a good many of these points, perhaps the vast majority, will be far from the support of the likelihood $p(T|\omega)$ and contribute little, we might salvage several tens of thousands that are in the support. That subset could be used to approximate the posterior density.

5 Summary

There were ample reasons to be skeptical of the claim by artificial intelligence researchers that machines would soon perform complex tasks at near-human or super-human levels, tasks such as face recognition which most humans perform without difficulty. Today, however, skepticism is no longer warranted given the ongoing machine learning revolution that began in 2006 when it became feasible to train deep neural networks. Today, networks with millions of parameters are routinely trained. Networks of this size have yet to be used in particle physics. But, Whiteson and collaborators have demonstrated promising results using smaller DNNs. We indulged in speculation about where DNN may be helpful in particle physics and evoked the notion of the automated physicist.

That the training of deep neural networks, using on-the-fly modifications of the network architecture such as dropout, can be cast as an approximation to inference with Bayesian neural networks is a significant conceptual advance. This suggests that a breakthrough in approximating posterior densities over enormously large parameter spaces would constitute another watershed event in machine learning and, therefore, in data analysis in particle physics.

References

[1] McCathy, J., Dartmouth Conference, (1956).
[2] Ruck, D. W. et al., IEEE Trans. Neural Networks **1**, 296-298 (1990).
[3] Wan, E. A., IEEE Trans. Neural Networks **1**, 303-305 (1990).
[4] The HiggsML Challenge, https://higgsml.lal.in2p3.fr/.
[5] The ATLAS collaboration, Aad G. et al. J. High Energ. Phys **04**, 117 (2015).
[6] LeCun Y., Bengio Y. and Hinton G., Nature **521**, 437 (2015).
[7] Bishop, C. M., *Pattern Recognition and Machine Learning* (Springer, Singapore, 2007) 1-743.
[8] Hinton, G. E., Osindero, S. and Teh, Y. **18**, 1527-1554 (2006).
[9] Cireşan D. C, Meier U., Gambardella L. M., Schmidhuber J., Neural Comput. **12**, 3207-3220 (2010).
[10] MNIST, http://yann.lecun.com/exdb/mnist/.
[11] Cireşan D., Meier U., Schmidhuber J., CVPR, 3642-3649 (2012).
[12] Ellis J., Phil. Trans. R. Soc. A **370**, 818-830 (2012).
[13] Lester, C. G. and Summers D. J., Phys. Lett. B **463**, 99 (1999).
[14] The CMS collaboration, Khachatryan V. et al., Phys. Lett. B **698**, 196-218 (2011).
[15] Rogan C., CALT **68-2790**, 1-10 (2010).
[16] The D0 collaboration, V. M. Abazov V. M. et al., Nature **429**, 638 (2004).
[17] The D0 collaboration, V. M. Abazov V. M. et al., Phys. Lett. B, 1 **617** (2005).
[18] Kondo K., J. Phys. Soc. Jpn. **60**, 836 (1991).
[19] Dalitz R. H. and Goldstein G. R., Phys. Rev. D **45** 1531 (1992).
[20] Baldi P., Sadowski P. and Whiteson D., Phys. Rev. Lett. **114**, 111801 (2015).
[21] Silver D. et al., Nature **529**, 484-489 (2016).
[22] Gal Y. and Ghahramani Z., ICML (2016).
[23] Loupe G., https://indico.cern.ch/event/516435.
[24] Casella G., The American Statistician **39**, 83-87(1985).