A BOON FOR EVALUATING ARCHITECTURE PERFORMANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

We point out several important problems with the common practice of using the best single model performance for comparing Deep Learning architectures, and we propose a method that corrects these flaws. Each time a model is trained, one gets a different result due to random factors in the training process, which include random parameter initialization and random data shuffling. Reporting the best single model performance does not appropriately deal with this stochasticity. Furthermore, the expected best result increases with the number of experiments run, among other problems. We propose a normalized expected best-out-of-n performance (Boo_n) as a way to correct these problems.

1 INTRODUCTION

Replicating results in Deep Learning (DL) research is often hard. This harms the usefulness of the research to industry and can lead to significant waste of effort by other researchers, not to mention very limited scientific value of such results.

One reason for these problems may be that insufficient information for replication is provided in many papers. Many details of the experimental setup can have a significant impact on the results (Henderson et al., 2017; Fokkens et al., 2013; Raeder et al., 2010), so these should be provided at least in appendices, ideally alongside the source code, as was strongly emphasized e.g. by Ince et al. (2012).

However, there is a second factor that can hinder replicability: the stochasticity of experiment results, which is inherent to practically all DL training methods, and which usually comes from *random* data ordering in *stochastic* gradient descent and from *random* parameter initialization, though there can be additional sources of randomness such as Dropout or gradient noise. Consequently, even if we fix the model architecture and the experimental setup (including the hyper-parameters), we obtain a different result each time we run the experiment. Hence, statistical techniques are needed to handle this variability when reporting results of experiments. However, in DL research such techniques are heavily underused. What is usually done instead?

The most common way of reporting quantitative results in DL research is currently the performance of the best single model – as we will show for some sub-domains. Given the result stochasticity, this method is statistically flawed. The best model performance is not robust under experiment replication, and its expected value improves with an increasing number of experiments performed, among other problems. Since these problems are largely ignored in current DL research publications, we dedicate the first part of this article to explaining and illustrating them in some detail.

These problems point to the necessity of statistical techniques for evaluating (and comparing) the performance of Machine Learning (ML) architectures. Methodology exists for such comparisons (a great introduction to such methods is given for instance by Cohen (1995)). However, most existing methodology focuses on comparing the *mean* performance. This may be one of the reasons why statistical methods are being underused, since mean may be unattractive to researchers in certain situations.

There may be multiple reasons for this. The one that we do consider sound[1] is that when deploying models in practice, it is often natural to train multiple instances of a model and then deploy the best one to production based on a validation set evaluation.[2] Underperforming models can be discarded, so the final deployed model does come from the higher tier of the model performance population, and the use of mean may be inappropriate.

Hence, in this paper, rather than completely abandon the method, we propose a way to fix the flaws of reporting the performance of the best model. We do this by estimating the expected Best out of n (Boo_n) performance by running more than n experiments, which gives the estimate statistical validity if a sufficient number of experiments are run. We discuss how this measure relates to the performance distribution of the model and also give a method to empirically estimate Boo_n.

2 BEST SINGLE MODEL PERFORMANCE

In most articles proposing new DL architectures, the performance is often reported as the score of the "best single model" or simply "single model". In practice, this usually means that the researchers train multiple instances of the proposed architecture, often with different sets of hyper-parameters, evaluate these instances on some validation set, select the best-performing model, which is then evaluated on a test set – this test score is then reported as the score characterizing the architecture and used for comparing it to previous models. If the score is better than those reported in previous work, the architecture is presented as superior. This current practice results in several issues:

Population variance The first problem is that since results of experiments are stochastic, the performance of the single model is just a single instance drawn from a possibly disparate population. If others train the model on their own, they get another sample from the architecture's performance distribution, which may substantially differ from the one listed in the original paper. The paper thus gives insufficient information about what to expect from the new architecture, which should be one of the article's main points.

One may object that the result published in the paper is not chosen from the population at random – it is selected using a validation result. However, the correlation between the validation and test results is generally imperfect; in fact, in some of our experiments, it is almost zero, as we show in Section 4. Furthermore, if we indeed do have a strong correlation, we get another problem:

Increasing expected result with increasing number of experiments Simply put, the more samples from a population we get, the more extreme the best among them is likely to be. In other words, the expectation of the best result is dependent on the number of experiments that the researchers run. There are three closely related problems with this: Firstly, this makes the number of experiments run an important explanatory variable; however, this variable is usually unreported, which is a severe methodological flaw in itself. However, it also leads to the second problem: since each research team runs a different number of experiments, the results are not directly comparable. Thirdly, this motivates researchers to run more experiments and gives an advantage to those that are able to run more experiments. This pushes publishing quantitative results towards a race in computational power rather than a fair comparison of architectures themselves.

Best model performance is not a meaningful characteristic of the performance distribution Even if we knew the underlying theoretical performance distribution – that is, if we had perfect information about the architecture's performance – it would not be clear what we would mean by

[1] Other reasons why researchers resort to the best performance as opposed to the mean may come from the current highly competitive atmosphere in the field with (possibly excessive) focus on performance on standard datasets (see Church (2017) for further discussion), which may motivate researchers to publish only their best results. Also, statistically sound estimation of performance does require repeatedly re-running experiments, which does incur additional cost, which researchers may prefer to invest in additional model tuning, especially in the present situation where reviewers do not require statistically sound evaluation of models and on the other hand may favour high-performing models. Of course that these reasons should instead give place to effort to do good science as opposed to a race on standard benchmarks.

[2] In some applications there is focus on speed of training and on reducing computational costs – there it does make sense to focus on the performance of the typical model as opposed to the best out of n, so the use of mean or median is appropriate.

"best model performance" without specifying the size of the pool from which we are choosing the best model. Imagine some architecture having a Gaussian performance distribution. Asking what is the best possible performance does not make sense in such a case, so using the best-model performance to characterize the performance distribution is not meaningful. Even for capped metrics such as accuracy, where the performance distribution necessarily has bounded support, the best (possible) model[3] may be so unlikely, that it would be of no practical importance. Hence, best model performance is not a meaningful characteristic of the performance distribution.

Generality / Falsifiability Finally, there is the question of what the authors are trying to express. Using "best single model performance", the authors are essentially claiming: "There once existed an instance of our model that had accuracy X % on a dataset Y". Such fact is not of that much interest to the scientific community, which would rather need to know how the architecture behaves *generally*. Relatedly, a frequently given characteristic of science is *falsifiability* of theories (Popper, 1959). A theory claiming that there are invisible unicorns running among us is not science, since we cannot think of any potential empirical evidence that could prove the theory false. Similarly, any number of replication experiments that produce substantially worse results cannot prove the above claim wrong. If, for instance, a confidence interval were given, replications could very quickly show a published result at least extremely implausible, if not false.

We will quantify the former two problems for two concrete architectures in Section 4.

2.1 Prevalence

Despite all these problems, reporting the performance of the best model is still the main way of reporting results in many areas of ML, and such practice seems to be tolerated by reviewers even at prime conferences. For instance, what concerns models published on the popular Children's Book Test dataset for Reading Comprehension (on which we run experiments later), none of the (more than ten) papers used any form of statistical testing or confidence intervals, and most reported only performance of the best single model without even mentioning the total number of experiments run. These include papers published at NIPS (Hermann et al., 2015), ICLR (Hill et al., 2016; Munkhdalai & Yu, 2017), ACL (Chen et al., 2016; Dhingra et al., 2017; Cui et al., 2016), or EMNLP (Trischler et al., 2016).

The same is true for the recently popular SQuAD dataset: for instance, none of the four papers (Yang et al., 2017; Wang & Jiang, 2017; Seo et al., 2017; Xiong et al., 2017) that published results on this dataset at ICLR 2017 have used any statistical testing or confidence intervals nor published mean (or otherwise aggregated) results across multiple runs.

Overall, while a significant proportion of papers at ICLR do contain an empirical evaluation of models or methods (as a rough guide, 174 out of 194 ICLR papers have "experiment" in a (sub-)section heading), only 11 mention terms related to hypothesis testing[4] and 11 contain the string "confidence interval". Further details can be found in Appendix A.

While this is certainly not meant as a comprehensive survey, it does suggest that while DL research is to a large extent an empirical science, the usage of statistical methods may be lagging behind and need improving.

3 Expected best-out-of-n (Boo$_n$) performance

The issues outlined above point to desiderata for a more suitable way of reporting an architecture's performance. Firstly, it should provide information about general behaviour of the architecture under specified conditions. Secondly, it should be invariant under the number of experiments run (and other similar variables). Finally, the resulting conclusion should be falsifiable and provide information relevant to the rest of the scientific community.

[3] In the sense of validation performance being at or near the supremum of the validation performance distribution's support.

[4] This was checked by searching for any of the following strings: "hypothesis test", "p-value", "t-test" "confidence level", "significance level", "ANOVA", "analysis of variance", "Wilcoxon", "sign test".

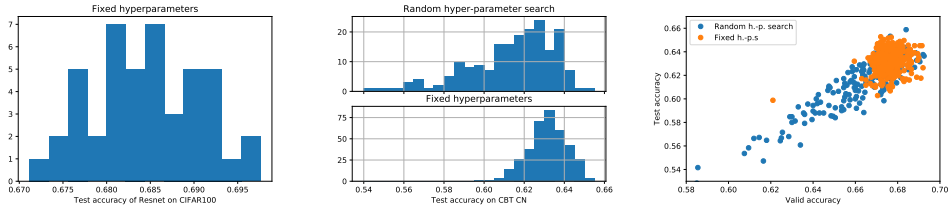


Figure 1: (a) & (b) Distribution of test accuracies of 75 instances of Resnet evaluated on CIFAR100, 370 instances of the AS Reader model with fixed hyper-parameters and 197 with random hyper-parameters trained and evaluated on CBT Common Noun subset. (b) Relationship between the test and validation accuracies of the AS Reader for random hyper-parameter search and for fixed hyper-parameters.

Given these requirements, traditional statistical measures, such as mean or median, probably come to mind of many readers. They do indeed fix the above issues; still, they express only the performance of a typical member of the population. However, in many ML applications, it may be the best model from a pool that is of interest. When practitioners are choosing a model for deployment, they may indeed train multiple models and then deploy the best-performing one [5]. This gives some justification to the practice of reporting the performance of the best model and gives us a reason to attempt to fix its problems rather than completely dismiss it. Such corrected best-model measure would be more useful than mean or median in situations outlined above.

A natural way to remove the lack of comparability between models, each evaluated in a different number of experiments, is to normalize the results to the expected result if the number of experiments were the same, say n. This can be quite easily estimated, if we run m experiments, $m \geq n$. The greater the number of experiments m, the more robust the estimate of the expected best, which also helps us eliminate the problem of statistical robustness. We are proposing the expected best-out-of-n performance Boo_n to be used where the performance of the best model from a pool seems as an appropriate measure.

Let us first examine how the expected best-out-of-n (Boo_n) performance relates to a known theoretical performance distribution; we will then proceed with empirical estimation, which is of value in practice.

3.1 Boo_n of a probability distribution

We will now examine how to calculate Boo_n from a known theoretical probability distribution. This will serve two purposes: Firstly, since we are proposing Boo_n as a way to characterize the performance distribution, this will make it explicit how Boo_n relates to the underlying theoretical distribution. Secondly, in some cases we may be able to make an assumption about the family to which the theoretical distribution belongs. The analytic calculation below will allow us to leverage this information when empirically estimating Boo_n, which may be especially useful when our sample size m is small.

3.1.1 Single evaluation

Let us first look on the case of validation performance as it is easier to grasp: How do we calculate an expected best $\overline{\text{Boo}_n}(\mathfrak{P})$ of independent identically distributed (i.i.d.) random variables $X_1, ..., X_n$ with probability distribution $\mathfrak{P}$ (the *performance distribution* of an architecture) with a probability density function (p.d.f.) f and a cumulative distribution function (c.d.f.) F? In the case where best means maximal (minimum can be calculated similarly), the maximum $\max\{X_1, ..., X_n\}$ has a c.d.f.

[5] This would usually be the case when a model is trained once and then deployed for longer-term usage, which may be the case for instance for Machine Translation systems. In other cases, when it is practical to train only as single model instance due to hardware constraints (either because training is extremely costly, or because it needs to be done repeatedly, e.g. for individual customers, we may indeed be interested in a typical model and hence in mean or median performance.

equal to

$$F_{\max}(x) = \mathbb{P}[\max\{X_1, ..., X_n\} \leq x] = \mathbb{P}[X_1 \leq x, ..., X_n \leq x] = F^n(x)$$

using the independence of the X_is in the last step. To obtain the p.d.f. of the maximum (in case of a continuous distribution), we just need to differentiate with respect to x:

$$f_{\max}(x) = \frac{d}{dx} F_{\max}(x) = nf(x)F^{n-1}(x)$$

Using the p.d.f., we can now calculate the expected value of the maximum as

$$\overline{\text{Boo}_n}(\mathfrak{P}) = \int_{-\infty}^{\infty} x f_{\max}(x) dx = \int_{-\infty}^{\infty} xnf(x)F^{n-1}(x)dx \quad . \tag{1}$$

We can get a precise numerical estimate of the above integrals in any major numerical computation package such as *numpy*. For illustration, for the standard normal distribution we have $\text{Boo}_5\left(\mathfrak{N}(0,1)\right) \approx 1.163$, $\text{Boo}_{10}\left(\mathfrak{N}(0,1)\right) \approx 1.539$. More generally, $\overline{\text{Boo}_n}\left(\mathfrak{N}(\mu,\sigma^2)\right)$ can then be expressed as $\mu + \sigma\overline{\text{Boo}_n}\left(\mathfrak{N}(0,1)\right)$. Thanks to this form we can get numerical estimates of $\overline{\text{Boo}_n}\left(\mathfrak{N}(\mu,\sigma^2)\right)$ just by estimating the two usual parameters of the Gaussian, $\overline{\text{Boo}_n}\left(\mathfrak{N}(0,1)\right)$ becoming just a constant coefficient if we fix n. The full details of calculation for the Gaussian distribution can be found in Appendix B.

In the case of a discrete performance distribution, which will be useful for empirical estimation below, we get a probability mass function

$$\mathbb{P}[\max\{X_1, \ldots, X_n\} = m] = \mathbb{P}[\max\{X_1, \ldots, X_n\} \leq m] - \mathbb{P}[\max\{X_1, \ldots, X_n\} < m]$$

so if p_i is the probability weight associated with value x_i, i.e. $\mathbb{P}[X_i = x_j] = p_j$ for all i, this gives us

$$\overline{\text{Boo}_n}(\mathfrak{P}) = \sum_i \left(\left(\sum_{j:\, x_j \leq x_i} p_j \right)^n - \left(\sum_{j:\, x_j < x_i} p_j \right)^n \right) x_i \quad . \tag{2}$$

3.2 Validation-test evaluation

In the previous part, we were choosing the best model with respect to the metric whose expectation we were calculating. Hence, that method can be used to calculate the expected best validation performance of n models. In practice, the best model is usually chosen with respect to the validation performance, while test performance is then of interest. To calculate the expectation of the test performance of the best-validation model, we need to substitute the direct value of x in Equation 1, with the expectation of the test performance X_{test} conditional on the validation performance x_{val},

$$E_{tv}\left(x_{\text{val}}\right) := \mathbb{E}\left[X_{\text{test}} | X_{\text{val}} = x_{\text{val}}\right]$$

yielding an expression for the expected test performance of the best-validation model chosen from a pool of size n

$$\text{Boo}_n(\mathfrak{P}) = \int E_{tv}(x_{\text{val}}) d\mathfrak{P}_{\text{val}}(x_{\text{val}}) \;=\; \int_{-\infty}^{\infty} E_{tv}(x_{\text{val}}) n f_{\text{val}}(x_{\text{val}}) F_{\text{val}}^{n-1}(x_{\text{val}}) dx_{\text{val}}$$

where $\mathfrak{P}_{\text{val}}$ is the marginal probability distribution of the validation performance. Similar simple substitution can be done in the discrete case.

Expanding the expression for the bivariate Gaussian distribution with mean test performance μ_{test} and test-validation covariance $\text{cov}\left(X_{\text{valid}}, X_{\text{test}}\right)$ as in Appendix B.2 again yields a convenient expression

$$\text{Boo}_n = \mu_{\text{test}} + \text{cov}\left(X_{\text{valid}}, X_{\text{test}}\right) \overline{\text{Boo}_n}\left(\mathfrak{N}(0,1)\right) \quad , \tag{3}$$

which can again be used for parametric estimation.

3.3 EMPIRICAL ESTIMATION

We usually do not know the exact performance distribution of the model; we only have samples from this distribution – the results of our experiments. In such case, we can estimate the expected maximum empirically, and in fact it is the empirical estimates that are likely to be used in practice to compare models.

To get a non-parametric estimator, for which we do not make any assumption about the family of the performance distribution, we take the empirical distribution arising from our sample as an approximation of the architecture's true performance distribution, similarly to Bootstrap methods. The empirical performance distribution $\widehat{\mathfrak{P}}$ assigns a probability weight of $\frac{1}{m}$ to each of our m samples. We approximate Boo_n of the true performance distribution by Boo_n of this empirical distribution. For the uniform empirical distribution, all the p_i in Equation 2 are equal to $\frac{1}{m}$. Hence if we rank our samples from worst-validation to best-validation as $(x_{i_1}^{\text{valid}}, x_{i_1}^{\text{test}}), \ldots, (x_{i_m}^{\text{valid}}, x_{i_m}^{\text{test}})$ we get

$$\widehat{\text{Boo}_n}\left((x_1^{\text{valid}}, x_1^{\text{test}}), \ldots, (x_m^{\text{valid}}, x_m^{\text{test}})\right) = \sum_{j=1}^{m} \left(\left(\frac{j}{m}\right)^n - \left(\frac{j-1}{m}\right)^n\right) x_{i_j}^{\text{test}} \quad .$$

This estimator does not make any assumption about the performance distribution from which our observations are drawn. If we do use such an assumption (e.g. we know that the performance distribution of our architecture usually belongs to a certain family, e.g. Gaussian), we can add information to our estimator and possibly get an even better estimate (i.e. one with lower sampling error). For the Gaussian distribution, we can use the standard estimators of the parameters in Equation 3 to get a parametric estimator

$$\widehat{\mu_{\text{test}}} + \widehat{\text{cov}}\left(\mathbf{x}_{\text{valid}}, \mathbf{x}_{\text{test}}\right) \overline{\text{Boo}_n}\left(\mathfrak{N}(0,1)\right).$$

where $\widehat{\mu_{\text{test}}}$ and $\widehat{\text{cov}}$ are standard estimators of mean and covariance respectively. A similar parametric estimator could be calculated for other distributions.

3.4 CHOICE OF n

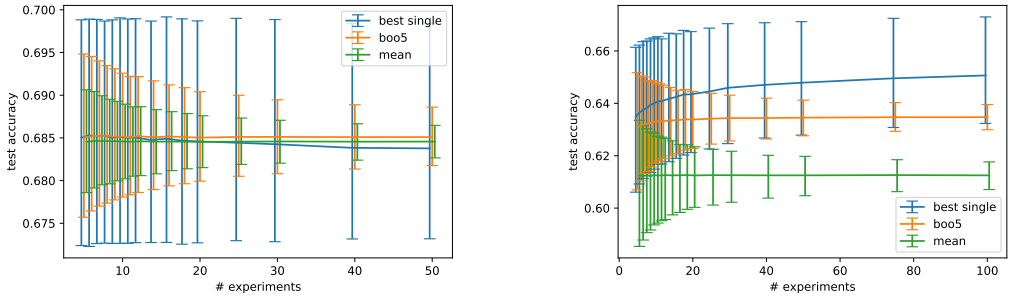
Our chief motivation for the above normalization was the elimination of the dependence of the performance score on the number of experiments, m. However, we still need to choose n, the number of experiments to which we normalize. This is similar to the choice one is facing when using a quantile – should one use the 75% one, the 95% one, or some other?

The choice of n most useful to a reader is when n is the number of candidate models that a practitioner would train before choosing the best one for some target application. Such number will differ from domain to domain and will heavily depend on the computational cost of training models in such domain. Even though the optimal choice is a matter of opinion, what is important is to create some convention to ensure comparability of models within a single domain. For end-users, providing results for multiple values of n would be most useful, however such practice may bring more confusion into the comparison of models in articles, since it allows for cherry-picking for the value of n, and consequently, performing statistical tests would require corrections for multiple testing.

In our experiments we decided to use $n = 5$.

3.5 ACCOUNTING FOR ESTIMATOR UNCERTAINTY

However, even Boo_n is still just a single number whose estimate can be noisy. Hence, with Boo_n, as well as with mean and other ways of aggregating results of a wider population, we should always use appropriate statistical methods when trying to compare the quantitative performance of a new model against a baseline. This can be done using significance testing (such as the t-test), or the uncertainty in performance estimation can be expressed using a confidence interval, which seems to be the method currently preferred by a significant part of the scientific community, since it allows us to disentangle the effect size from the uncertainty associated with noise and sample size (Berrar & Lozano, 2013; Gardner & Altman, 1986). For some theoretical distributions, there exist ways to calculate the hypothesis test or confidence interval analytically (e.g. using the t-test or standard normal quantiles for the Gaussian). However in cases where the family of the performance distribution or of the estimator is not known, we need to resort to computational methods - usually Monte Carlo (if we do know at least the family of the performance distribution) or Bootstrap (Efron, 1979) (if we do not).



(a) Resnet (fixed hyperparameters)

(b) AS Reader (random hyperparam. search)

Figure 2: Averages and 95% confidence intervals for three ways of aggregating results of multiple experiments for various numbers of experiments run. Each confidence interval was constructed using smoothed[6]Bootstrap sampling from our pool of 75 for Resnet and 197 experiments for the AS Reader with random hyperparameters respectively. Since we strongly encourage researchers to provide confidence intervals for their results, we provide and overview of how to construct them using the Bootstrap in Appendix D.1.

4 Experimental results

We have run several experiments to quantify the scope of the problems outlined in Section 2. We just briefly summarize the main results here for illustration; a more detailed description of the experiments can be found in Appendix C.

Performance variation To estimate the random variation of the results, we repeatedly [7] trained models from two domains of DL: the ResNet (Huang et al., 2017) on the CIFAR-100 dataset (Krizhevsky & Hinton, 2009) in image recognition and the Attention Sum Reader (AS Reader) (Kadlec et al., 2016) on the Children's Book Test Common Nouns (CBT CN) (Hill et al., 2016) to represent Reading Comprehension. The resulting performance distributions are illustrated in Figure 1. If we fix all hyper-parameters, the interquartile ranges of the models' error rates are 0.98% and 1.20% (absolute). Compare this to the median improvements of subsequent published results on these datasets: 0.86% and 1.15% respectively. Hence, random variation in performance cannot be considered negligible as is currently often done. Furthermore, if we allow the hyper-parameters to vary (in our case by random search), the result variance further increases, which further amplifies the outlined effects. In the case of the AS Reader the interquartile range increased to 2.9% when we randomly picked hyper-parameters from a range that we previously found to work reasonably well.

Several other articles confirm significant variation in model performance due to different random seed: e.g van den Berg et al. (2016) in Speech Recognition, Henderson et al. (2017) in Deep Reinforcement Learning, or Reimers & Gurevych (2017) in Named Entity Recognition. They all conclude that reporting performance scores of single models is insufficient to characterize model performance.

Best-model performance improves with the number of experiments We also mentioned that if only the performance of the best model is reported, the more experiments are run, the better the expected result. Figure 2b illustrates that this effect can indeed be fairly strong, if the validation performance is a good predictor of the test performance, as is the case of the AS Reader with random hyper-parameter search, where the expectation of the best single model performance increases from 61.3% if we train it once, to 63.3% if we train it 5 times, to 63.5% for 20 times. This effect is further explained e.g. by Jensen & Cohen (2000). It gives a further argument for refraining from using this

[6]While Boo_n and mean could be sampled using vanilla Bootstrap, best-validation result is influenced only by a single value from the sample and hence uses only few values from the upper tier of our result pool, which makes our pool size insufficient. Hence we use Gaussian kernel smoothing (Scott, 1992) to expand our result pool.

[7]Specifica;ly, 75 times for Resnet, 370 times for the AS Reader with fixed hyperparameters, and 197 times for the AS Reader with random hyperparameters.

method and certainly also for publishing the number of experiments run, which is often not done. Boo_n is not subject to this effect.

Validation-test correlation However, note that the assumption that validation performance is a good predictor of the test performance is sometimes not true. In the two cases with fixed hyper-parameters that we looked at, Spearman correlation between validation and test results was only 0.10 and 0.18 respectively for the two models. The correlation significantly increases if we allow the hyper-parameters to vary – to 0.83 for the AS Reader. These results are also illustrated in Figure 1. Also larger validation sets are likely to improve this correlation, which can be understood as the degree of generalization from validation to test. Note that the problem of increasing expected performance mentioned above is relevant only in the case of correlation between validation and test results. However, the effect becomes very strong in the case where the performance we are reporting is also used for choosing the best model, which emphasizes the need for honest separation of validation and test data.

Estimator variance Figure 2 shows the 95% confidence intervals of best single model results compared to the Boo_5 performance for a range of result-pool sizes m. This is shown for the cases of both strong and weak test-validation correlation. In both cases Boo_5 is significantly less noisy than the best-single-model result. In fact in the case of random hyper-parameter search, Boo_n shows even smaller variation than the mean (due to the negative skew of the performance distribution).

5 Conclusion

In the first part of this article, we have outlined several major problems with the common practice of using the "best single model" performance as a common way of reporting quantitative performance of Deep Learning architectures. We do not think these flaws will come as a surprise to anyone active in this area; still, they seem being ignored, which makes them worth pointing out. We believe each of the problems would be sufficient to make the practice inappropriate.

Hence, we need a better way of characterizing the performance distributions of proposed architectures. Before returning to our case of single-number characterization, we should emphasize that one ought to attempt to characterize the distribution as fully as possible to give the reader reasonable expectations about how the model behaves – for instance, if the performance distribution seems to be (approximately) Gaussian, authors can provide the distribution's parameters (the mean and the variance in this case), ideally with appropriate confidence intervals. This allows readers to subsequently choose the aspect of the distribution that they find the most interesting, whether it be median or the expected best-out-of-n performance. Of course, to complete the picture, such distribution should ideally be provided for multiple test datasets and associated metrics, since each brings to light a different aspect of the model performance, and such practice helps avoid fine-adjusting models to particular datasets, which harms the generalizability of the results.

However, *alongside* this detailed characterization, describing an architecture's performance by a single number may still have its appeal, especially for the purpose of comparison among architectures and choosing the best one according to some criterion[8]. We have presented elements of one such possible method, which could be summarized as follows.

After choosing the most appropriate metric, such as accuracy, together with an appropriate test task – or possibly multiple of these – one should choose a suitable way of aggregating the results of models, to get some sensible characteristic of the architecture's performance distribution – this can be the traditional mean or median, it can be the expected best model performance from a pool of n, which we have proposed as a substitute for the currently used "best single model" performance, or it can be some other characteristic. This choice can differ domain from domain. However, such method should be reasonably robust with respect to observation sampling, and it should be invariant under choices of experiment design that often differ between research teams, such as the number of experiments run, which was discussed in this article.

[8]In fact, each quantitative score can be understood as a proxy for ordering architectures with respect to some criterion of interest.

Once one chooses the appropriate measure of converting the pool of observations into a score characterizing the architecture's performance distribution, one also needs to account for the uncertainty in the estimation of such score. We described how to construct confidence intervals using the Bootstrap as a possible way of accounting for this uncertainty. This may allow researchers to assess the significance of the quantitative improvement that their architecture brings, which is unfortunately done rarely in recent research.

We believe that methodology currently common in DL research is lagging behind the standards usual in most other sciences. Improving it, which can be worked on by both authors and reviewers, would bring us better reproducibility of results, as well as make the results more meaningful to both other researchers and to industry practitioners.

The main new theoretical contribution of the paper was in proposing a method that fixes the main problems of the popular practice of reporting best single model performance. The article firstly described both how it relates to the underlying (theoretical) performance distribution. Secondly, it looked on how to estimate the the expected best-out-of-n performance from empirical observations.

We do not claim that the proposed method is appropriate for all situations – for instance, when we are interested chiefly in the performance of a typical model from a population, mean or median may be a better choice. However, the target application often involves deploying the best model from a pool, and in that case, the performance of the expected best model out of n may be more informative. Hence we believe it will be a useful contribution to the toolbox for describing and comparing the performance of ML models.

REFERENCES

T. W. Anderson and D. A. Darling. A test of goodness of fit. *Journal of the American Statistical Association*, 49(268):765–769, 1954.

Daniel Berrar and Jose A Lozano. Significance tests or confidence intervals: which are preferable for the comparison of classifiers? *Journal of Experimental & Theoretical Artificial Intelligence*, 25(2): 189–206, 2013.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A Thorough Examination of the CNN / Daily Mail Reading Comprehension Task. In *Association for Computational Linguistics (ACL)*, 2016.

Kenneth Ward Church. Emerging trends: I did it, I did it, I did it, but .... *Natural Language Engineering*, 23(03):473–480, 2017.

Paul R Cohen. *Empirical methods for artificial intelligence*. MIT press Cambridge, MA, 1995.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-Attention Neural Networks for Reading Comprehension. 2016. URL `http://arxiv.org/abs/1607.04423`.

Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan Salakhutdinov. Gated-Attention Readers for Text Comprehension. In *Proceedings of ACL*, 2017.

B Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, pp. 1–26, 1979.

Bradley Efron. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82(397):171–185, 1987.

Antske Fokkens, Marieke Van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. Offspring from reproduction problems: What replication failure teaches us. In *ACL (1)*, pp. 1691–1701, 2013.

M J Gardner and D G Altman. Confidence intervals rather than p values: estimation rather than hypothesis testing. *British Medical Journal*, 292(6522):746–750, 1986.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv:1709.06560*, 2017.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1684–1692, 2015.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*, 2016.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

Darrel C Ince, Leslie Hatton, and John Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485, 2012.

David D Jensen and Paul R Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text Understanding with the Attention Sum Reader Network. *Proceedings of ACL*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Tsendsuren Munkhdalai and Hong Yu. Reasoning with memory augmented neural networks for language comprehension. In *ICLR*, 2017.

Karl R Popper. *The logic of scientific discovery*. 1959.

Troy Raeder, T Ryan Hoens, and Nitesh V Chawla. Consequences of variability in classifier performance estimates. In *IEEE 10th International Conference on Data Mining (ICDM)*, pp. 421–430, 2010.

Nils Reimers and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of EMNLP*, 2017.

David W Scott. Multivariate density estimation: Theory, practice, and visualization. 1992.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *ICLR*, 2017.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural Language Comprehension with the EpiReader. In *Proceedings of EMNLP*, 2016.

Ewout van den Berg, Bhuvana Ramabhadran, and Michael Picheny. Training variance and performance evaluation of neural networks in speech. pp. 1–5, 2016. URL `http://arxiv.org/abs/1606.04521`.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *ICLR*, 2017.

Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *ICLR*, 2017.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. Words or characters? fine-grained gating for reading comprehension. *ICLR*, 2017.

APPENDIX A SURVEY OF ICLR 2017 PAPERS: METHOD

To roughly estimate the usage of statistical methods in ICLR 2017 papers, we downloaded the pdfs of all accepted papers[9], extracted text from them using the OpenSource Xpdf package[10] and then searched the resulting text documents using the grep command as follows.

Firstly, to roughly estimate the usage of experiments in the papers, we searched for the capitalized string "EXPERIMENT" in the documents, since all (sub-)section headings are capitalized in the ICLR format. This was matched in 174 documents. Further 6 contained the string "EVALUATION" yielding a total of 180 out of 194 papers containing one of the two strings, which suggests that many ICLR papers indeed have an empirical component, though our rough method is only very approximate.

We then searched for the string "confidence interval", which was matched in only 11 papers, and further 11 documents matched one of expressions related to hypothesis testing (curiously, a set completely disjoint from the "confidence interval" set). These terms were: "hypothesis test", "p-value", "t-test" "confidence level", "significance level", "ANOVA", "analysis of variance", "Wilcoxon", and "sign test". This may be only a lower bound since mentioning the term does not necessarily mean that the method was employed in the experimental setup.

APPENDIX B BOO_n OF THE GAUSSIAN DISTRIBUTION

In this section we will calculate Boo_n of the Gaussian distribution. This can serve as a basis for a parametric estimator of Boo_n, when we assume a performance distribution to be (approximately) Gaussian, which was the case of some of the performance distributions we have examined, for instance the AS Reader with fixed hyper-parameters.

B.1 SINGLE EVALUATION DATASET

In the simpler case in which the best model is chosen with respect to the same dataset on which the performance is then reported, we can substitute the p.d.f. and c.d.f. of the Gaussian distribution into Equation 1 to get

$$\overline{\text{Boo}_n}(\mathfrak{N}(\mu,\sigma^2)) = \int_{-\infty}^{\infty} xn \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \Phi^{n-1}\left(\frac{x-\mu}{\sigma}\right) dx$$

where Φ is the c.d.f. of a standard normal random variable. Substituting $z = \frac{x-\mu}{\sigma}$, $dx = \sigma dz$, yields

$$\int_{-\infty}^{\infty} n(\mu+\sigma z) \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{z^2}{2}} \Phi^{n-1}(z)\, \sigma dz =$$

$$= \mu \int_{-\infty}^{\infty} n \frac{1}{\sqrt{2\pi}} e^{\frac{z^2}{2}} \Phi^{n-1}(z)\, dz + \sigma \int_{-\infty}^{\infty} nz \frac{1}{\sqrt{2\pi}} e^{\frac{z^2}{2}} \Phi^{n-1}(z)\, dz = \mu + \sigma \overline{\text{Boo}_n}(\mathfrak{N}(0,1))$$

(the first integrand has the form of the p.d.f. found above and hence integrates to one) so the expected maximum is neatly expressed in terms of a maximum of a standard normal and is linearly proportional to both the mean and the standard deviation. Once n is fixed for comparison purposes, $\text{Boo}_n(\mathfrak{N}(0,1))$ is just a constant, e.g. $\text{Boo}_5(\mathfrak{N}(0,1)) \approx 1.163,\ \text{Boo}_{10}(\mathfrak{N}(0,1)) \approx 1.539$.

B.2 TEST-VALIDATION EVALUATION

Let us turn to the case of reporting a test set performance of a best-validation model. If we model the validation and test performances by a Bivariate Normal Distribution with valid-test correlation ρ, means $\mu_{\text{val}}, \mu_{\text{test}}$, and variances $\sigma_{\text{val}}, \sigma_{\text{test}}$, then given a validation performance x_{val}, the test performance is distributed normally with conditional expectation

$$E_{tv}(x_{\text{val}}) = \mu_{\text{test}} + \sigma_{\text{test}}\rho\,(x_{\text{val}} - \mu_{\text{val}})$$

[9] Downloaded from https://openreview.net/group?id=ICLR.cc/2017/conference from sections "Paper decision: Accept (Oral)" and "Paper decision: Accept (Poster)".

[10] http://www.xpdfreader.com/; we used version 4.00.01 on Debian Linux 9.2.

which gives

$$\text{Boo}_n(\mathfrak{N}(\mu,\sigma^2)) = \int_{-\infty}^{\infty} (\mu_{\text{test}} + \sigma_{\text{test}}\rho(x_{\text{val}} - \mu_{\text{val}}))\, n \frac{1}{\sqrt{2\pi\sigma_{\text{val}}^2}} e^{\frac{(x-\mu_{\text{val}})^2}{2\sigma_{\text{val}}^2}} \Phi^{n-1}\left(\frac{x-\mu_{\text{val}}}{\sigma_{\text{val}}}\right) dx.$$

Using the same two tricks as above, this can be simplified to

$$\text{Boo}_n(\mathfrak{N}(\mu,\sigma^2)) == \mu_{\text{test}} + \sigma_{\text{test}}\sigma_{\text{val}}\ \rho\ \overline{\text{Boo}_n}\left(\mathfrak{N}(0,1)\right) = \mu_{\text{test}} + \text{cov}\left(X_{\text{valid}}, X_{\text{test}}\right) \overline{\text{Boo}_n}\left(\mathfrak{N}(0,1)\right) \tag{4}$$

where $\text{cov}\left(X_{\text{valid}}, X_{\text{test}}\right)$ is the covariance between the validation and test performances, and $\text{Boo}_n\left(\mathfrak{N}(0,1)\right)$ is the single-evaluation expected maximum of the standard normal distribution as defined above.

APPENDIX C EXPERIMENTS: DETAILS

Note: We will provide data and code for our experimental analysis when the paper is no longer subject to the anonymity requirement.

Here we provide further details of our experiments quantifying the extent of result stochasticity and the resulting effects.

C.1 MODELS

To run our experiments we have chosen Open Source implementations[11] of models from two popular domains of Deep Learning, namely ResNet (Huang et al., 2017) on the CIFAR-100 dataset (Krizhevsky & Hinton, 2009) for Image Classification and the AS Reader (Kadlec et al., 2016) on the CBT CN dataset (Hill et al., 2016) for Reading Comprehension. We believe these two models representative of models in their respective areas – Resnet is based on a deep convolutional network architecture as most models in Machine vision, while the AS Reader is based on a bidirectional GRU network with attention, as is the case for many models in Natural Language Processing.

C.2 DATA COLLECTION

To collect the data for our experiments, we repeatedly trained the two models. Each training instance had a different random parameter initialization and random data shuffling. We evaluated the model on the validation and test datasets at least once per epoch. We then took the validation and test performance at the best-validation epoch as a data point for our further analyses.

All training was done on Ubuntu 14.04 on a single GPU per training, either Nvidia Tesla K80 or GTX 1080.

C.2.1 RESNET

Resnet was trained with a single set of hyperparameters, the default ones for the above Open Source implementation. That means 5 residual units resulting in a 32-layer Resnet. The model was trained using the 0.9 momentum optimizer, with batch size 128, initial learning rate of 0.1 lowered to 0.01 after 40,000 steps and to 0.001 after 60,000 steps. Data augmentation included padding to 36x36 and then random cropping, horizontal flipping and per-image whitening. L2 regularization weight was set 0.002.

Training was done using Tensorflow 1.3.

C.3 AS READER

The AS Reader was trained in two different settings. Firstly 370 times with hyper-parameters fixed to embedding dimension of 128 and 384 hidden dimensions in the GRU units, with all other hyper-parameters as used in the original AS Reader paper (Kadlec et al., 2016).

[11] The source code for Resnet can be found at `https://github.com/tensorflow/models/tree/master/research/resnet`; the code for the AS Reader at `https://github.com/rkadlec/asreader`.

In the second setting, the hyper-parameters for each training instance were chosen randomly from the following ranges: The batch size was chosen from the range $[16, 128]$, and the embedding size and hidden state size were each chosen from the range $[16, 512]$ with the $\log_2$ value of the parameter being distributed uniformly in the interval. Parameters from these ranges worked reasonably well in our preliminary experiments.

Training was done using Theano 0.9.0 and Blocks 0.2.0.

C.4 PERFORMANCE DISTRIBUTION RESULTS

Figure 1 plots the histograms of test performances of the evaluated models. The mean test accuracy for Resnet was 68.41% with standard deviation of 0.67% (absolute), the range was $67.31\% - 69.41\%$. For AS reader with fixed hyperparameters the mean was 63.16% with standard deviation 0.94% and range of $61.52\% - 64.60\%$. In the case of random hyper-parameter search the mean was 61.26%, standard deviation 2.48%, and values ranged from 56.61% to 64.01%.

In both cases with fixed hyper-parameters the collected results are consistent with coming from a Gaussian distribution according to the Anderson-Darling test[12] Anderson & Darling (1954); the histograms also make it appear plausible that the performance distribution is approximately Gaussian. This is not the case for the random hyper-parameter search where the distribution has a clear negative skew.

To put the above numbers into context, we also examined the margin of improvement of successive architectures published on the corresponding datasets, as listed in Munkhdalai & Yu (2017); Huang et al. (2017). We sorted the results with respect to the test performance and then calculated the differences between successive models. The median difference was for 0.86% for Resnet and 1.15% for the AS Reader.

Note that the median differences are smaller than two standard deviations for each model. Two standard deviations from the mean approximately give the 95% confidence interval for a Gaussian distribution – hence we could typically fit three successive published results within the width of one such confidence interval. The magnitude of the performance variation due to random initialization and data shuffling is therefore not negligible compared to the improvements in performance, which often hold an important place within articles in which they are presented. We hence think it is inappropriate to completely ignore this random variation in evaluation protocols, which is currently the usual practice.

C.5 TEST-VALIDATION CORRELATION

The best model is usually selected using validation performance [13]. This practice is based on the assumption that the validation accuracy is a reasonably good predictor of test accuracy. The results of our experiments, illustrated also in Figure 1, suggest that this assumption holds for performance variation due to hyper-parameter choice. However, if we fix the hyper-parameters, the correlation almost disappears. To some extent, this implies that selecting the best validation model means we are picking randomly with respect to the test performance. Since we are picking from a random test performance distribution, this further calls for better characterization of the distribution than a single instance drawn from it.

On the other hand if the correlation is strong, as seems to be the case if we do perform hyper-parameter search, we face the second problem with reporting the best-validation performance:

C.6 EFFECT OF THE NUMBER OF EXPERIMENTS

If the validation performance is a good predictor of the test performance, then the more models we train the better the best-validation model is likely to be even on the test set since we are able to select models high up the right tail of the performance distribution. This effect has been described in more

[12] That is, despite the relatively large sample sizes, gaussianity cannot be ruled out at 0.05 significance level based on collected evidence.

[13] Or at least should be.

detail in Jensen & Cohen (2000), though with focus on induction algorithms; here we present an estimate of its effect in the case of Resnet and AS Reader.

To test this effect we took the pool of trained modelsn. For each m in the range from 1 to 50 (or 100 for the AS Reader), we randomly sampled $100,000$ samples of size m from the pool, and selected the best-validation model from each sample. The mean test performance across the $100,000$ samples for each m is plotted in Figure 2.

The results show that when there is suitable correlation between validation and test performances, increasing the number of experiments does increase the expected performance of the best-validation model. This makes the number of experiments an important explanatory variable, which however usually goes unreported. Furthermore, it makes results reported by different research teams not directly comparable. Finally, it gives an advantage to those that can run more experiments. We believe that this again makes the practice of reporting the performance of the best single model unsuitable.

APPENDIX D DEALING WITH ESTIMATOR UNCERTAINTY

D.1 CONFIDENCE INTERVALS

If an estimator characterizing a performance distribution, say $\widehat{\text{Boo}_n}$ or average, is calculated from experimental observations, it is subject to random variation, so if another research team tries to reproduce the experiments, they generally get a different estimate. The more observations are collected, the more precise the estimate generally is. Confidence intervals provide a natural way to express this uncertainty. Their usage also gives a sense whether the number of performed experiments was sufficient to reduce the uncertainly to a reasonable level, which is again not frequently addressed in ML papers.

The construction of the confidence interval would be trivial if we knew the distribution from which our estimate was drawn – it is simply the interval between the appropriate quantiles, e.g. the 2.5th and 97.5th quantiles in the case of the 95% confidence interval. Such distribution has been studied extensively for instance in the case of a mean of Gaussian random variables. However, in other cases, it is not known. If we know at least the distribution from which the individual observations were drawn, we can use Monte Carlo methods to precisely estimate the confidence interval; however, if we are not able to make an assumption about the underlying distribution, we need to use only what we have: our samples from the distribution. In such case the variability of our estimator can be approximated using the Bootstrap (Efron, 1979) or similar methods.

The Bootstrap consists of repeatedly sampling *with replacement* m random observations from our pool of m observations, say B times. Each such sample is then used to calculate an estimate of our quantity of interest, say Boo_n or mean. This creates a sample of B values of the estimator. The confidence interval can then be easily estimated taking the appropriate quantiles from this resulting *Bootstrap* distribution of the estimator, which should be approximating the unknown underlying sampling distribution. The Bootstrap distribution has been shown to converge to the true underlying performance distribution.

If we know the underlying distribution (up to some parameters), we can estimate its parameters and then generate a simulated *Monte Carlo* sample from the distribution, which can be used to calculate a sample of the estimator and the corresponding confidence interval in a similar way as above with the advantage of the distribution being smoother.

Beside estimating the confidence interval for the value of Boo_n or mean itself, either re-sampling method can be used to construct a confidence interval for the relative improvement of the newly proposed architecture compared to a baseline. The improvement can then be considered significant if zero is not included in the confidence interval. More details on constructing Bootstrap confidence intervals can be found in many standard texts on computational statistics, for instance in Efron (1987).

For illustration, we calculated the Bootstrap confidence interval for several sample sizes m for Resnet and the AS Reader. Each was constructed using $B = 100,000$. The results are plotted in Figure 2.

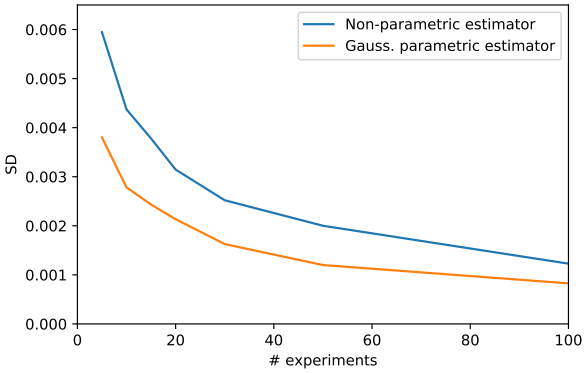


Figure 3: Empirical comparison of the variances of the non-parametric and Gaussian parametric Boo_n estimators. The data come from experiments with the AS Reader on CBT with fixed hyperparameters.

APPENDIX E COMPARISON OF ESTIMATORS

Figure 3 shows the comparison of the non-parametric and Gaussian parametric estimators of Boo_n, both introduced in Section 3.3, in terms of their variance for various sample sizes. The parametric estimator shows a somewhat lower variance. This is an advantage if the performance distribution is indeed approximately Gaussian, which is the case for both cases with fixed hyperparameters that we tested in our experiments. However, this can introduce bias if the true performance distribution differs from the theoretical distribution assumed by a parametric estimator, so one should be prudent to use it.