Appendices to Deep-n-Cheap: An Automated Search Framework for Low Complexity Deep Learning

All references, equation, table and figure numbers in this document refers to the main paper: "Deep-n-Cheap: An Automated Search Framework for Low Complexity Deep Learning" by authors Sourya Dey, Saikrishna C. Kanala, Keith M. Chugg and Peter A. Beerel.

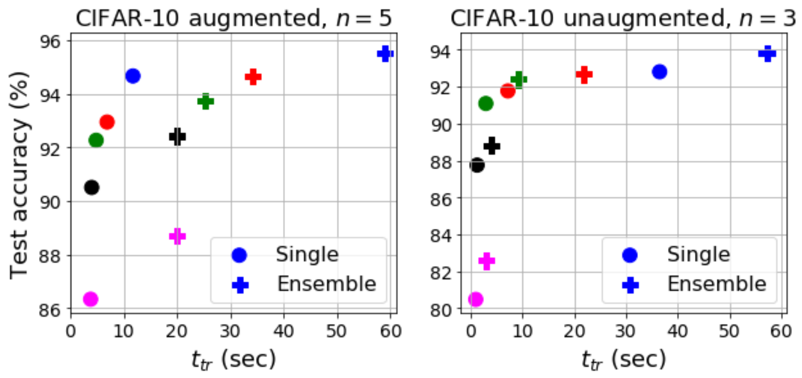
Appendix A. Validity of our covariance kernel

The validity of our covariance kernel can be proved as follows. We note that since x_{ik} and x_{jk} are scalars, d in eq. (3a) is the Euclidean distance. It follows from the properties of the squared exponential kernel that $\sigma\left(x_{ik}, x_{jk}\right)$ in eq. (3b) is a valid kernel function. So if a kernel matrix $\boldsymbol{\Sigma_k}$ were to be formed such that $\Sigma_{k_{ij}} = \sigma\left(x_{ik}, x_{jk}\right)$, then $\boldsymbol{\Sigma_k}$ would be positive semi-definite. Writing eq. (3c) in matrix form gives $\boldsymbol{\Sigma} = \sum_{k=1}^{K} s_k \boldsymbol{\Sigma_k}$. Since a convex combination of positive semi-definite matrices is also positive semi-definite, it follows that $\boldsymbol{\Sigma}$ is a valid covariance matrix.

Appendix B. Ensembling

One way to increase performance such as test accuracy is by having an ensemble of multiple networks vote on the test set. This comes at a complexity cost since multiple neural networks (NNs) need to be trained. We experimented on ensembling by taking the n best networks from Bayesian Optimization (BO) in Stage 3 of our search. Note that this *does not increase the search cost* as long as $n \leq n_1 + n_2$. However, it does increase the effective number of parameters by a factor of exactly n (since each of the n best configurations (configs) have the same architecture), and t_{tr} by some indeterminate factor (since each of the n best configs might have a different batch size).

We experimented on CIFAR-10 unaugmented using $n = 3$ and augmented using $n = 5$. The impact on the performance-complexity tradeoff is shown in the figure below. Note how the plus markers – ensemble results – have slightly better performance at the cost of significantly increased complexity as compared to the circles – single results. However, we did not use ensembling in other experiments since the slight increases in accuracy do not usually justify the significant increases in t_{tr}.



Performance-complexity tradeoff for single configs (circles) vs ensemble of configs (pluses) for $w_c = 0$ (blue), 0.01 (red), 0.1 (green), 1 (black), 10 (pink). Results using ensemble of 5 for CIFAR-10 augmented, and 3 for CIFAR-10 unaugmented.

Appendix C. Changing hyperparameters of Bayesian Optimization

The BO process itself has several hyperparameters that can be customized by the user, or optimized using marginal likelihood or Markov chain Monte Carlo methods (Swersky et al. (2013)). This section describes the default values we used. Expected improvement involves an exploration-exploitation tradeoff variable ξ. The recommended default is $\xi = 0.01$ (Brochu et al. (2010)), however, we tried different values and empirically found $\xi = 10^{-4}$ to work well. Secondly, f is a noisy function since the computed values of network performance are noisy due to random initialization of weights and biases for each new state. Accordingly, and also considering numerical stability for the matrix inversions involved in BO, our algorithm incorporates a noise term σ_n^2. We calculated its value from the variance in f values as $\sigma_n^2 = 10^{-4}$, which worked well compared to other values we tried.

Appendix D. Adaptation to various platforms

While most deep NNs are run on GPUs, situations may arise where GPUs are not readily or freely available and it is desirable to run simpler experiments such as multilayer perceptron (MLP) training on CPUs. Deep-n-Cheap (DnC) can adapt its penalty metrics to any platform. For example, the Fashion MNIST (FMNIST) results shown in Fig. 4 were on CPU, while Table 3 shows results on GPU (to do a fair comparison with other frameworks). As a result, the t_{tr} values are an order of magnitude faster, while the performance is the same as expected.