Model Agnostic Conformal
Hyperparameter Optimization

Riccardo Doyle  
R.DOYLE.EDU@GMAIL.COM

Spotify  
London, UK

Abstract

Several novel frameworks for hyperparameter search have emerged in the last decade, but most rely on strict, often normal, distributional assumptions, limiting search model flexibility. This paper proposes a novel optimization framework based on Conformal prediction, assuming only exchangeability, and allowing for a larger choice of search model architectures and variance estimators. Several such models are explored and benchmarked against random hyperparameter search on both dense and convolutional neural networks with consistent overperformance both in final loss achieved and time to achievement.

Keywords: hyperparameter optimization, conformal prediction, deep learning, Bayesian Optimization

1. Introduction

Identifying optimal model parameters is deeply desirable for high prediction performance in machine learning, but challenging due to non-convexity and expensive search costs.

Common approaches involving grid search - exhaustive iterative search of a confined parameter interval - or random search [1] - random sampling from a broader parameter space – display complimentary weaknesses and form no expectation of hyperparameter performance ahead of search. Much focus has instead been placed on search frameworks involving pre-sampling inference, generally dominated by Sequential Model-Based Optimization (SMBO) [2] – sequential training of a performance estimator on previously sampled hyperparameters to guide future sampling based on estimator expectations. Early applications [2, 3] found positive outperformance on expert consensus across a range of benchmarked datasets through Gaussian Process or Tree-structured Parzen estimators. Encouraging results promoted further research with additions involving search cost inclusion as an optimization criterion [4], early forms of online resource allocation and distributed search [5], unwanted parameter space pruning [6], or replacement of single estimators with ensemble methods [7], among others.
Though alternative approaches have surfaced [8], SMBO-based search has become one of the most popular non-naïve search methods in the training of complex machine learning predictors, with frequent appearances in competitions [9] and package releases [10]. Its Bayesian architecture allows for rigorously constructed normally distributed confidence intervals and the prevalent use of Gaussian Processes as point estimators allows for flexible and inexpensive retraining, however while its impositions of normality are beneficial to validity, they restrict point estimators’ functional forms.

In this study we replace SMBO with a similarly structured, but less constraining, point-estimate and variance process with no reliance on normally distributed outputs. Specifically, SMBO’s commonplace Gaussian point estimators will be substituted with a choice of regression models and its normal confidence intervals with conformal prediction intervals limited only by assumptions of exchangeability. This approach retains the benefits of a fitted predictor and sequential framework, but offers wider flexibility in the choice of both point and variance estimators, allowing for more complex, and, expectantly, better fitting architectures.

2. Review of Conformal Prediction

Common hyperparameter sampling methods involving statistical optimization rely on confidence intervals to identify areas of high uncertainty. While robust, these bounds impose distributional assumptions on the data – often normality – resulting in a limited choice of models. As an alternative, in this section we introduce the concept of conformal prediction intervals.

2.1 Split Conformal Prediction (SCP)

For a training set \(X_{\text{train}}, Y_{\text{train}} = \{(X_i, Y_i)\}_{i=1}^n\) and a validation set \(X_{\text{val}}, Y_{\text{val}} = \{(X_j, Y_j)\}_{j=n+1}^N\), let us calibrate a chosen point estimator model to the training set generating the mapping function \(f(X) = \hat{Y}\).

To construct a conformal prediction interval [11] for the estimator’s outputs, we must define some nonconformity function \(C(\hat{Y}, Y)\) quantifying the divergence between observed and model function-predicted outputs \(Y\) and \(\hat{Y}\). Functional form is arbitrary and needs to satisfy only exchangeability of outputs. For simplicity we define this as the absolute deviation between observed and predicted outputs:

\[
C(\hat{Y}, Y) = |Y - \hat{Y}|
\]  

(1)

We then obtain predicted outputs for all validation set observations \(Y_{\text{val}} = f(X_{\text{val}})\), which can be in turn passed jointly with their observed counterparts through the conformity scoring function to obtain a multiset of divergences \(D\):

\[
D = |Y_{\text{val}} - \hat{Y}_{\text{val}}|
\]  

(2)

For a given confidence level \(\alpha\), we now denote our conformal prediction interval on an unseen new test input \(X_{N+1}\) as:
\[ CI(X_{N+1}) = [f(X_{N+1}) \pm q_{1-\alpha}(D)] \]  
\[ (3) \]

Where \( q_{1-\alpha}(D) \) is the \( 1 - \alpha \) quantile of the multiset \( D \).

Consequently, we state the probability bound with which a new \( Y_{N+1} \) label falls within our previously defined confidence interval as:

\[ 1 - \alpha + \frac{1}{(N - n) + 1} \geq P(Y_{N+1} \in CI(X_{N+1})) \geq 1 - \alpha \]  
\[ (4) \]

### 2.2 Locally Weighted Conformal Prediction (LWCP)

Prediction intervals formed with SCP can produce valid expectations of uncertainty for a custom fitted point estimator, however the width of the interval is constant and \( X_i \) invariant. This is not suited to SMBO inspired frameworks, as expected variance at each point is constant and the algorithm would be purely exploitative.

To remedy this, we can revise the conformity score to include a conditional estimate of variance \( [12] \) or uncertainty \( V(X) \):

\[ C(\hat{Y}, Y) = \frac{|Y - \hat{Y}|}{V(X)} \]  
\[ (5) \]

Which in turn updates our final prediction interval to:

\[ CI(X_{N+1}) = [f(X_{N+1}) \pm V(X)q_{1-\alpha}(D)] \]  
\[ (6) \]

Such that higher variance or uncertainty expands the interval and lower variance contracts it, conditional on the predicted value \( X_{N+1} \).

Much like the first fitted point estimator, the uncertainty function \( V(X) \) can be obtained by training a custom predictor model – may be the same model architecture as that used to calibrate \( f(X) \) – on \( X_{train} \) and some output measure of uncertainty \( v \), then predicting the test uncertainties \( v_{val} = V(X_{val}) \) and updating the previously defined multiset of divergences as:

\[ D = \frac{|Y_{val} - \hat{Y}_{val}|}{v_{val}} \]  
\[ (7) \]

In this study, we select mean absolute deviation as the uncertainty measure \( v_i = |Y_i - \mu(X_i)| \) and linear regression as the model’s architecture.

### 2.3 Quantile Weighted Conformal Prediction (QWCP)

Variance estimation methods described in section 2.2 can be exchanged for a number of other dispersion estimators to varying degrees of performance improvement. For instance, the variance estimator in equation 7 measures dispersion around a global training data mean with no local concentration. To supplement this, we replace traditional variance with the spread between two custom percentile quantiles conditional on \( X = x \), with quantiles produced via quantile regression [13].
In its simplest linear form, quantile regression adopts a recognizable formulation \( Y_i = \sum_{j=1}^{p} X_{i,j} \beta_j + e_i \) but as opposed to deriving parameters \( \beta_j \) through OLS, it optimizes the pinball loss function:

\[
L_\alpha(u_i) = \begin{cases} 
  u_i \alpha & \text{if } u_i > 0 \\
  u_i(\alpha - 1) & \text{if } u_i \leq 0 
\end{cases}
\]

(8)

Where \( u_i \) is the absolute error between predicted and observed outputs \( u_i = |Y_i - \sum_{j=1}^{p} X_{i,j} \beta_j| \). Regression parameters are then estimated according to the minimization:

\[
\arg \min_{\beta_1, \ldots, \beta_p} \sum_{i:Y_i \geq \sum_{j=1}^{p} X_{i,j} \beta_j}^{N} \alpha u_i + \sum_{i:Y_i < \sum_{j=1}^{p} X_{i,j} \beta_j}^{N} (1 - \alpha) u_i
\]

(9)

It can separately be shown that the \( \alpha \)-th quantile minimizes the above equation, resulting in a regression whose output, per observation, is a pre-specified interval conditional on the desired confidence level and \( X = x \).

Let us define the conditional quantile function generated by the aforementioned quantile regression as \( Q(X_i, \alpha) \). To make use of its outputs as a dispersion metric in conformal prediction we simply substitute \( V(X) \) in equation 6 with \( Q(X_i, \alpha) \) and specify a discretionary percentile \( \alpha \) to size the dispersion impact.

Above, we derived quantile regression using a linear framework, however several non-linear quantile regression formulations exists such as Gradient Boosted Regression – leveraging conventional gradient boosting but adding the previously outlined pinball loss function minimization – or Quantile Forest – repurposing the outputs of individual training trees in a Random Forest algorithm to produce confidence bounds [14]. The performance of a range of quantile frameworks, including those just outlined, will be explored in later result sections.

3. Conformal Hyperparameter Optimization (CHO)

3.1 Algorithm Structure

Let \( \theta_t \) be the hyperparameter configuration sampled at iteration \( t \) of a random search framework tuning a chosen machine learning model \( \Theta \). For a training set \( X_{\text{train}}, Y_{\text{train}} = \{(X_i, Y_i)\}_{i=1}^{n} \) we begin by fitting a prediction function \( f(X) \) on the sampled configuration \( \theta_t \) as defined by:

\[
f(X) \leftarrow \Theta(X_{\text{train}}, \theta_t)
\]

(10)

We then introduce some custom loss function \( L(Y, f(X)) \) evaluating the error in \( f(X) \)’s outputs and define the loss incurred by configuration \( \theta_t \) at a given iteration on a validation set \( X_{\text{val}}, Y_{\text{val}} = \{(X_i, Y_i)\}_{i=n+1}^{n'} \) to be:

\[
l_t = L(Y_{\text{val}}, f(X_{\text{val}}))
\]

(11)
The loss incurred by each sequentially randomly sampled configuration is stored and after a minimum number of arbitrary iterations – set to \( t_{CHO} = t > 40 \) in this study’s specifications – random search is replaced by a pure CHO process.

We first split the accumulated pairs of sampled hyperparameter configurations and their respective loss into a training set \( \theta_{train}, l_{train} = \{ (\theta_j, l_j) \}_{j=1}^m \) and validation set \( \theta_{val}, l_{val} = \{ (\theta_j, l_j) \}_{j=m+1}^M \). We then fit a secondary learning model \( \Lambda \) – who’s form can take that of most machine learning models – on the training set relation between past hyperparameter configurations and their loss:

\[
e(\theta) \leftarrow \Lambda(l_{train}, \theta_{train}, \lambda) \tag{12}
\]

Where \( \lambda \) is some chosen hyperparameter configuration for model \( \Lambda \), to be tuned separately through random search.

Using equation 12 to sample future observations by lowest expected loss would result in a purely exploitative framework, so uncertainty is added by computing variance-informed lower bounds through conformal methodology previously described in sections 2.1, 2.2 and 2.3.

First, we compute a measure of distance or nonconformity between expected and realized observations. This may take several forms as previously stated and for simplicity we take the case of LWCP as outlined in section 2.2. An additional custom model \( \tau \) is fitted on training set pairs of hyperparameter configurations \( \theta_{train} \) and local mean absolute deviations in loss \( |l_j - \mu(l_{train})| \) resulting in a point estimator of variance conditional on \( \theta \):

\[
V(\theta) \leftarrow \tau(\theta_{train}, |l_j - \mu(l_{train})|, \delta) \tag{13}
\]

Where \( \delta \) is some chosen hyperparameter configuration for model \( \tau \).

A nonconformity score can then be produced on the validation set by redefining section 2.2’s equation 7 with our previously fitted estimators:

\[
D = \frac{|l_{val} - e(\theta_{val})|}{V(\theta_{val})} \tag{14}
\]

Consequently we define the conformal prediction interval on a given hyperparameter \( \theta \) to be:

\[
CI(\theta) = [f(\theta) \pm V(\theta)q_{1-\alpha}(D)] \tag{15}
\]

Lastly, we transition from previously sampled hyperparameters to the wider hyperparameter space of all possible configurations \( \theta_{space} = \{ (\theta_k)_{k=1} | k \in \mathbb{Z}, |k| \in [0...t] \} \) or in practice, a very large number of randomly combined hyperparameter configurations. Each unsampled configuration is passed through equation 15 producing the prediction interval:

\[
CI(\theta_{space}) = [f(\theta_{space}) \pm V(\theta_{space})q_{1-\alpha}(D)] \tag{16}
\]

With a lower bound \( l_{ob} \) obtained from \( f(\theta_{space}) - V(\theta_{space})q_{1-\alpha}(D) \). The parameter configuration \( \theta_i \) with the lowest lower bound is then selected as the next to sample in place of
\( \theta_t \) in equation 10, with the process repeating until a desired early stopping point. A detailed pseudo code representation of the above methodology is reported in Algorithm 1.

**Algorithm 1: Conformal Hyperparameter Optimization (CHO)**

| initialization: MAD\((l_i) = |l_i - \mu(l_{\text{train}})|\). iter = 0, min_iter = 40, \( P = \) [], \( L = \) [] |
|---|
| 1: \( X_{\text{train}}, Y_{\text{train}}, X_{\text{val}}, Y_{\text{val}} \leftarrow \text{train_val_split}(X, Y) \) |
| 2: \( \theta_{\text{space}} = \{ \theta_1; \theta_N \} \) |
| 3: for \( i = 1; \text{len(\( \theta_{\text{space}} \))} \) do |
| 4: else if iter \( \leq \) min_iter do |
| 5: \( (f: X \mapsto Y) \leftarrow \Theta(X_{\text{train}}, Y_{\text{train}}, \theta_i) \) |
| 6: \( l_i = |f(X_{\text{val}}) - Y_{\text{val}}| \) |
| 7: \( L.\text{append}(l_i) \) |
| 8: \( P.\text{append}(\theta_i) \) |
| 9: else do |
| 10: \( P_{\text{train}}, P_{\text{val}}, L_{\text{train}}, L_{\text{val}} \leftarrow \text{train_val_split}(P, L) \) |
| 11: \( (\varepsilon: \theta \mapsto \ell) \leftarrow \Lambda(P_{\text{train}}, L_{\text{train}}, \lambda) \) |
| 12: \( (V; \theta \mapsto \text{MAD}(\ell)) \leftarrow \tau(P_{\text{train}}, \text{MAD}(L_{\text{train}}), \delta) \) |
| 13: \( D = \frac{|L_{\text{val}} - e(P_{\text{val}})|}{V(P_{\text{val}})} \) |
| 14: lower_bound of \( \theta_{\text{space}} = e(\theta_{\text{space}}) - V(\theta_{\text{space}})q_{1-a}(D) \) |
| 15: \( \theta_i = \arg\min_{\theta}(\text{lower_bound}_{\theta_{\text{space}}}) \) |
| 16: \( (f: X \mapsto Y) \leftarrow \Theta(X_{\text{train}}, Y_{\text{train}}, \theta_i) \) |
| 17: \( l_i = |f(X_{\text{test}}) - Y_{\text{test}}| \) |
| 18: \( L.\text{append}(l_i) \) |
| 19: \( P.\text{append}(\theta_i) \) |
| 20: iter = iter + 1 |
| 21: end |

Some algorithm components are purposefully undefined as user discretion may apply. To elaborate on the functional forms chosen in this study, the primary prediction model \( \Theta \) is set to be either a dense or convolutional neural network. The loss function in equation 11 is set to log loss, as all benchmarked datasets in later results sections involve classification. The conformal point estimator model \( \Lambda \) is set to be either a Random Forest, K-Nearest Neighbours, or Dense Neural Network predictor. The MAD predictor \( \tau \) is set to be a simple linear regression, although this – as per previous methodology – assumes a LWCP framework (section 2.2). Two other choices of \( V(\theta) \) will be tested involving QWCP (section 2.3), first with a gradient boosted quantile estimator and secondly with a quantile forest estimator. Lastly, the user may specify training and validation breakdowns across the algorithm, but an election worth noting in this study is the split value for hyperparameter and loss datasets \( \theta_{\text{train}}, l_{\text{train}} \) and \( \theta_{\text{val}}, l_{\text{val}} \), which employs a fixed validation window of 10 randomly sampled observations. This maximizes available inference data in the training set – as validation data is solely used to determine conformal quantiles. The reduced window limits coverage accuracy of conformal intervals but improves predictive accuracy of the conformal point estimator, with the validation split always balancing this trade-off for slow training models \( \Theta \) with fewer hyperparameter samples per unit time.
Beyond outlined components, this study’s implementation of CHO includes additional runtime efficiency heuristics. Namely the retraining of the $e(\theta)$ and $V(\theta)$ predictive functions in equation 12 and 13 occurs every $\xi$-th iteration, where $\xi$ is a user specified integer this study sets to between 3 and 5. A higher $\xi$ value offers less runtime overhead at the expense of slower adapting functions between iterations and vice versa. A second efficiency parameter concerns the number of random hyperparameters to sample in tuning the conformal point estimator $\Lambda$ at each CHO iteration, where a larger number increases computational overhead. To assess this, let us first take some function $\bar{R}(x)$ to return the time taken to fit any kind of predictor model on a given input dataset and its labels using a single hyperparameter configuration. We can then define random search’s estimated runtime on the subject model $\Theta$ as the mean runtime

$$\frac{1}{t_{\text{CHO}}} \sum_{t=1}^{t_{\text{CHO}}} \bar{R}(\Theta(X_{\text{train}}, \theta_t)).$$

To return a comparable estimate on the conformal point predictor $\Lambda$, we tune its first CHO iteration ($t = t_{\text{CHO}} + 1$) on an arbitrarily large number of hyperparameter configurations $n$ and obtain mean runtime $\frac{1}{n} \sum_{i=1}^{n} \bar{R}(\Lambda(l_{\text{train}}, \theta_{\text{train}}, \lambda_i))$. A constraint function can then be defined, limiting the ratio of conformal runtime to random search runtime by a user specified upper bound $\kappa$:

$$\frac{\frac{1}{n} \sum_{i=1}^{n} \bar{R}(\Lambda(l_{\text{train}}, \theta_{\text{train}}, \lambda_i)) \cdot \xi \cdot n_{\text{optimal}}}{\frac{1}{\max_{\text{iter}}} \sum_{t=1}^{\max_{\text{iter}}} \bar{R}(\Theta(X_{\text{train}}, \theta_t))} < \kappa$$

(18)

Where $n_{\text{optimal}}$ is the number of conformal randomly sampled hyperparameter configurations that, if used for every subsequent iteration, would guarantee a runtime surplus below $\kappa$. Thus trivially rearranging equation 18 to solve for $n_{\text{optimal}}$ returns the number of efficient random searches to perform at each CHO iteration to tune model $\Lambda$, pending some desired ratio $\kappa$. This study always sets $\kappa = \xi$, which enforces a maximum runtime matching that of random search.

As the training cost of $e(\theta)$ is fixed and invariant of both $\Theta$ model complexity and $X_{\text{train}}$ data size, its runtime overhead on random search would tend to zero as either complexity or size increase. Thus, above overrides are primarily recommended for simpler models or smaller input datasets in which nominal training cost of $e(\theta)$ is meaningfully higher than random search’s, though they can be applied to any model or data context for improved performance.

4. Experiments and Results

4.1 CHO Performance on Dense Artificial Neural Network (DNN)

Performance is first tested on the parameters of a dense fully connected artificial neural network with a searchable hyperparameter space comprised of 20,000 randomly generated combinations of individual parameter values reported in Table 1.

We consider four benchmarking classification datasets: sci-kit learn’s integer recognition dataset [15] (DIGITS), a reduced (5,000 samples) variation of the IMDb sentiment analysis dataset [16] (IMDB), the 1994 US Census income bracket prediction dataset [17] (CENSUS) and the Diagnostic Wisconsin Breast Cancer dataset [18] (DWBC).
Table 1: Individual hyperparameter values of searchable hyperparameter space for dense fully connected artificial neural network.

| Hyperparameter                  | Search Values                      |
|--------------------------------|------------------------------------|
| **Solver**                     | [Adam, SGD]                         |
| **Learning Rate**              | [0.00001, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1] |
| **Alpha Regularization**       | [0.00001, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.5, 1, 3, 10] |
| **Nodes in Given Layer**       | [2, 4, 6 … 28, 30, 32, 50, 75, 100] |
| **Number of Layers**           | [2, 3, 4, 5]                        |

Performance on the DIGITS dataset using several CHO variations alongside random search (RS) can be found in Fig. 1. Loss on the validation partition of the dataset is plotted over training runtime using a selection of three conformal variance estimators $V(\theta)$ – Linear LWCP, Gradient Boosted QWCP and Quantile Regression Forest QWCP – and three conformal point estimators $\hat{\theta}$ - K Nearest Neighbours (KNN), Random Forest (RF) and a Dense Neural Network (DNN). We note the choice of variance estimator has an impact on search performance, with the Linear LWCP model achieving the best cross-architecture loss profile. We also note CHO substantially outperforms random search with eight of its nine variations resulting in a lower end of run loss and the best performing combination achieving a 35% loss reduction on random selection (0.324 random search log loss compared to 0.209 for RF Linear LWCP). There is substantial variation in the extent of prediction interval coverage granted by each CHO combination. Linear LWCP provides the lowest excess breaches, achieving good performance on DNN and KNN models (21.4% and 23.4% breach rate against a desired 20% rate for the specified 80% confidence level) but less so on RF, while both QWCP models produce noisy, less accurate coverage (possibly due to their increased predictive complexity), which could partly explain their worse overall sampling performance. It is worth reaffirming that all conformal intervals were formed from a 10 observation validation set to maximize training set hyperparameter observations, thus suboptimal coverage accuracy would be expected.

Further benchmarking was carried out on the IMDb, CENSUS and DWBC datasets using a single best performing variance estimator - Linear LWCP – and runtime performance is summarized in Fig. 2. Plots show both minimum validation loss and 15 iteration moving average loss over the course of training. Accuracy is again improved by the introduction of CHO, with all search model combinations in each dataset performing at or above random search levels. Time savings are of particular note in the IMDb dataset, with random search finding a single improved parameter configuration close to 1.5 hours into runtime, compared to CHO’s best variation finding comparable improvement after 21 minutes on its 41st iteration (the first optimized sampling iteration following its enforced 40 random search iterations). Performance gains were similar between smaller CENSUS and DWBC datasets and the heavier IMDb dataset proving context versatility. Computational overhead from CHO’s statistical architecture and potentially higher complexity parameter sampling is minimal, with the best search models by runtime in both the CENSUS and DWBC datasets achieving a lower runtime per iteration than random search, and the best model in the IMDb dataset exacting only a 17.1% increase.
Fig 1: Best DIGITS validation log loss across various conformal point predictors and conformal variance predictors over primary model training time.

Table 2: Summary of key benchmarking metrics across various conformal point predictors and conformal variance predictors on the DIGITS dataset.

| CHO Point Predictor | CHO Variance Predictor | Min. Validation Loss | Avg. Validation Loss** | Final Holdout Loss* | Runtime per Search | CI Breach Rate*** |
|---------------------|------------------------|----------------------|------------------------|---------------------|--------------------|------------------|
| RS                  | -                      | 0.324                | 1.495                  | 0.487               | 3.399              |                  |
| DNN                 | Linear LWCP            | 0.240                | 1.312                  | 2.315               | 4.309              | 21.4%            |
|                     | Gradient Boosted QWCP  | 0.254                | 1.331                  | 0.167               | 4.497              | 0%               |
|                     | Quantile Regression Forest QWCP | 0.324    | 1.248                  | 0.487               | 4.192              | 9.1%             |
| KNN                 | Linear LWCP            | 0.210                | 1.150                  | 0.244               | 3.065              | 23.4%            |
|                     | Gradient Boosted QWCP  | 0.253                | 1.290                  | 0.425               | 2.642              | 7.0%             |
|                     | Quantile Regression Forest QWCP | 0.318| 1.201                  | 0.333               | 3.306              | 0.7%             |
| RF                  | Linear LWCP            | 0.209                | 0.821                  | 0.147               | 4.139              | 11.7%            |
|                     | Gradient Boosted QWCP  | 0.295                | 1.133                  | 1.567               | 3.037              | 4.5%             |
|                     | Quantile Regression Forest QWCP | 0.237   | 1.166                  | 0.274               | 3.919              | 3.6%             |

* Holdout loss from best performing validation hyperparameter configuration at end of runtime.
** Calculated as the average validation set loss across all sampled configurations by end of runtime.
*** Where ideal breach rate is 20% following specification of an 80% confidence level.
Fig 2: Best validation log loss (top row) and 30 iteration moving average log loss (bottom row) across various conformal point predictors and conformal variance predictors over primary model training time using multiple datasets (IMDb, CENSUS, DWBC).

Table 3: Summary of key benchmarking metrics across various conformal point predictors and conformal variance predictors on the IMDb, CENSUS and DWBC datasets.

| Dataset      | CHO Point and Variance Predictor | Min. Validation Loss | Avg. Validation Loss* | Final Holdout Loss* | Runtime per Search | CI Breach Rate*** |
|--------------|----------------------------------|----------------------|-----------------------|---------------------|--------------------|-------------------|
| IMDb 3000    | DNN Linear LWCP                  | 0.308                | 0.605                 | 0.356               | 43.560             | 32.5%             |
|              | KNN Linear LWCP                  | 0.310                | 0.551                 | 0.359               | 42.275             | 43.7%             |
|              | RF Linear LWCP                   | 0.308                | 0.448                 | 0.350               | 48.057             | 33.3%             |
|              | RS                               | 0.327                | 0.689                 | 0.358               | 36.094             | -                 |
| 1994 Census  | DNN Linear LWCP                  | 0.326                | 0.461                 | 0.332               | 5.694              | 22.2%             |
|              | KNN Linear LWCP                  | 0.320                | 0.430                 | 0.331               | 4.885              | 16.3%             |
|              | RF Linear LWCP                   | 0.323                | 0.393                 | 0.346               | 3.990              | 5.5%              |
|              | RS                               | 0.326                | 0.457                 | 0.338               | 4.121              | -                 |
| Breast Cancer| DNN Linear LWCP                  | 0.058                | 0.366                 | 0.105               | 1.647              | 16.6%             |
|              | KNN Linear LWCP                  | 0.047                | 0.281                 | 0.136               | 0.487              | 18.9%             |
|              | RF Linear LWCP                   | 0.059                | 0.189                 | 0.143               | 1.861              | 5.0%              |
|              | RS                               | 0.067                | 0.345                 | 0.124               | 0.767              | -                 |

* Holdout loss from best performing validation hyperparameter configuration at end of runtime.

** Calculated as the average validation set loss across all sampled configurations by end of runtime.

*** Where ideal breach rate is 20% following specification of an 80% confidence level.
4.2 CHO Performance on Convolutional Neural Network (CNN)

We further test our methodology on a convolutional neural network (CNN) architecture to validate performance on a more complex configuration of parameters, this time with an increased searchable hyperparameter space comprised of 40,000 randomly generated combinations of individual parameter values reported in Table 4.

We consider four benchmarking classification datasets: a reduced (2,000 samples) variation of the MNIST digit recognition dataset [19] (MNIST), a reduced (2,000 samples) variation of the multi background Fashion MNIST digit recognition dataset [20] (FMNIST), and a reduced variation (2,000 samples) of the CIFAR-10 object detection dataset [21] (CIFAR 10).

| Hyperparameter                        | Search Values                  |
|---------------------------------------|--------------------------------|
| Solver                                | [Adam, SGD]                     |
| Learning Rate                         | [0.00001, 0.0001, 0.001, 0.01, 0.1] |
| Drop Out Rate                         | [0.01, 0.05, 0.03, 0.1, 0.2]    |
| Pooling Layer: Pooling Size           | [None, 2x2]                     |
| First Convolutional Layer: Number of Convolutions | [4, 6, … 18, 20] |
| First Convolutional Layer: Convolution Size | [3x3, 5x5] |
| Second Convolutional Layer: Number of Convolutions | [5, 10, … 35, 40] |
| Second Convolutional Layer: Convolution Size | [3x3, 5x5] |
| Third Convolutional Layer: Number of Convolutions | [5, 10, … 45, 50] |
| Third Convolutional Layer: Convolution Size | [3x3, 5x5] |
| First Dense Layer: Number of Neurons   | [150, 200]                     |
| Second Dense Layer: Number of Neurons  | [0, 50, 100]                   |

Differentials in validation loss across CHO variations on the three benchmarked datasets can be found in Fig. 3. We note performance is once again superior to random search, with all nine CHO variations achieving a lower end of runtime validation loss across datasets. The outperformance is particularly marked in the MNIST dataset, with the best CHO variation (KNN Linear LWCP) achieving a minimum validation loss of 0.016 compared to random search’s 0.353 (22 fold higher). Strong performance is also achieved on the FMNIST dataset, with a best minimum validation loss of 0.099 against 0.312, while CIFAR 10 yields smaller improvements, with both CHO and random search ending with relatively high loss for what would be expected of this dataset - likely due to slower runtime resulting in less iterations for inference and the loss of explanatory data from reducing CIFAR 10’s comparably higher complexity to 2,000 samples. The conformal interval is breached between 5.6% and 22.7% of the time depending on the CHO variation, with only 4 of the 9 variations having a breach rate within ± 3% of the intended 20% level, though this is once again reflective of the small 10 observation fixed window used to determine the interval.
Fig 3: Best validation log loss (top row) and 30 iteration moving average log loss (bottom row) across various conformal point predictors and conformal variance predictors over primary model training time using multiple datasets (MNIST, FMNIST, CIFAR 10).

Table 5: Summary of key benchmarking metrics across various conformal point predictors and conformal variance predictors on the MNIST, FMNIST and CIFAR 10 datasets.

| Dataset   | CHO Point and Variance Predictor | Min. Validation Loss | Avg. Validation Loss** | Final Holdout Loss* | Runtime per Search | CI Breach Rate*** |
|-----------|----------------------------------|----------------------|------------------------|--------------------|--------------------|-------------------|
| MNIST     | DNN Linear LWCP                  | 0.055                | 8.389                  | 0.144              | 84.867             | 12.8%             |
|           | KNN Linear LWCP                  | **0.016**            | 7.134                  | 0.261              | 70.497             | 22.7%             |
|           | RF Linear LWCP                   | 0.046                | **4.554**              | 0.144              | 71.803             | 5.6%              |
|           | RS                               | 0.353                | 7.931                  | 2.552              | **44.510**         | -                 |
| FMNIST    | DNN Linear LWCP                  | **0.099**            | 7.902                  | **0.052**          | 49.994             | 11.5%             |
| Dataset   | KNN Linear LWCP                  | 0.150                | 8.062                  | 0.254              | 64.370             | 20.6%             |
|           | RF Linear LWCP                   | 0.179                | **4.028**              | 0.139              | **35.762**         | 16.2%             |
|           | RS                               | 0.312                | 8.427                  | 0.288              | 39.548             | -                 |
| CIFAR 10  | DNN Linear LWCP                  | 2.076                | 7.538                  | 4.757              | 101.741            | 19.0%             |
| Dataset   | KNN Linear LWCP                  | 1.963                | 7.918                  | **2.221**          | 123.349            | 9.3%              |
|           | RF Linear LWCP                   | **1.782**            | **6.422**              | 9.690              | 105.231            | 17.9%             |
|           | RS                               | 2.144                | 7.639                  | 3.608              | **93.806**         | -                 |

* Holdout loss from best performing validation hyperparameter configuration at end of runtime.
** Calculated as the average validation set loss across all sampled configurations by end of runtime.
*** Where ideal breach rate is 20% following specification of an 80% confidence level.
5. Conclusion

This study introduced a novel optimization framework for hyperparameter selection based on conformal prediction intervals and a broad choice of point estimator models. Performance across a range of benchmarked datasets was superior to random search in both final loss achieved and time to achievement. Empirical validity of conformal intervals was mixed, though linear variance estimators reached acceptable coverage and wider excessive breaches are easily explained by expected low overall – and in particular, validation set – data availability.

Further performance improvements could be achieved with the introduction of early stopping logic based on expected improvement inference, the inclusion of expected search cost as an additional sampling criterion, and – for large datasets – bootstrapping-based sub-partition of the input data for concurrent search.
References

1. Hutter, F., Holger, H., and Leyton-Brown, K., 2011. Sequential Model-Based Optimization for General Algorithm Configuration. International Conference on Learning and Intelligent Optimization, pp.507-523.

2. Bergstra, J., Bengio, Y., 2012. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13, pp.281-305.

3. Bergstra, J., Bardenet, R., Bengio, Y., and Kegl, B., 2011. Algorithms for Hyper-Parameter Optimization. Advances in Neural Information Processing Systems, 24.

4. Snoek, J., Larochelle, H., and Adams, R., 2012. Practical Bayesian Optimization of Machine Learning Algorithms. Advances in Neural Information Processing Systems, 24, pp. 2960-2968.

5. Swersky, K., Snoek, J., and Adams, R., 2013. Multi-Task Bayesian Optimization. Advances in Neural Information Processing Systems, 26.

6. Witsuba, M., Schilling, N., and Schmidt-Thieme, L., 2015. Hyperparameter Search Space Pruning – A New Component for Sequential Model-Based Hyperparameter Optimization. Machine Learning and Knowledge Discovery in Databases, pp. 104-119.

7. Lacoste, A., Larochelle, H., Laviolette, F., and Marchand, M., 2014. Sequential Model-Based Ensemble Optimization. arXiv.

8. Li, L., Jamieson, K., Rostamizadeh, A., and Talwalkar, A., 2018. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. Journal of Machine Learning Research, 18, pp. 1-52.

9. Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., and Guyon, I., 2021. Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. arXiv.

10. Kandasamy, K., Vysyaraju, K., Neiswanger, W., Paria, B., Collins, C., Schneider, J., Poczos, B., and Xing, E., 2020. Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly. Journal of Machine Learning Research, 21, pp. 1-27.

11. Shafer, G., and Vovk, V., 2008. A Tutorial on Conformal Prediction. Journal of Machine Learning Research, 9, pp. 371-421.

12. Lei, J., G’Sell, M., Rinaldo, A., Tibshirani, R., and Wasserman, L., 2018. Distribution-Free Predictive Inference for Regression. Journal of The American Statistical Association, 0, pp. 1-18.

13. Koenker, R., and Bassett, G., 1978. Regression Quantiles. Econometrica, 46, 33–50.

14. Meinshausen, N., 2006. Quantile Regression Forests. Journal of Machine Learning Research, 7, 983–999.

15. scikit-learn. sklearn.datasets.load_digits. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html [Accessed 13 June 2022].

16. Keras. IMDB movie review sentiment classification dataset. Available at: https://keras.io/api/datasets/imdb/ [Accessed 13 June 2022].
17. UCI Machine Learning Repository. Census Income Data Set. Available at: https://archive.ics.uci.edu/ml/datasets/census-income [Accessed 13 June 2022].

18. UCI Machine Learning Repository. Breast Cancer Wisconsin (Diagnostic) Data Set. Available at: https://archive.ics.uci.edu/ml/datasets/breast-cancer-wisconsin+(diagnostic) [Accessed 13 June 2022].

19. Keras. MNIST digits classification dataset. Available at: https://keras.io/api/datasets/mnist/ [Accessed 13 June 2022].

20. Keras. Fashion MNIST dataset, an alternative to MNIST. Available at: https://keras.io/api/datasets/fashion_mnist/ [Accessed 13 June 2022].

21. Keras. CIFAR10 small images classification dataset. Available at: https://keras.io/api/datasets/cifar10/ [Accessed 13 June 2022].