Measuring training variability from stochastic optimization using robust nonparametric testing

Sinjini Banerjee *Student Member, IEEE*, Tim Marrinan, Reilly Cannon, Tony Chiang, Anand D. Sarwate *Senior Member, IEEE*

***Abstract*—Deep neural network training often involves stochastic optimization, meaning each run will produce a different model. This implies that hyperparameters of the training process, such as the random seed itself, can potentially have significant influence on the variability in the trained models. Measuring model quality by summary statistics, such as test accuracy, can obscure this dependence. We propose a robust hypothesis testing framework and a novel summary statistic, the α-trimming level, to measure model similarity. Applying hypothesis testing directly with the α-trimming level is challenging because we cannot accurately describe the distribution under the null hypothesis. Our framework addresses this issue by determining how closely an approximate distribution resembles the expected distribution of a group of individually trained models and using this approximation as our reference. We then use the α-trimming level to suggest how many training runs should be sampled to ensure that an ensemble is a reliable representative of the true model performance. We also show how to use the α-trimming level to measure model variability and demonstrate experimentally that it is more expressive than performance metrics like validation accuracy, churn, or expected calibration error when taken alone. An application of fine-tuning over random seed in transfer learning illustrates the advantage of our new metric.**

***Index Terms*—DNN variability, non-parametric testing, Kolmogorov-Smirnov test, robust statistics, ensembling**

I. Introduction

Deep learning models have achieved state-of-the-art performance on complex tasks in healthcare, education, cybersecurity, and other critical domains. Training these models takes significant time, energy, and hence financial resources. Training algorithms use stochastic optimization for non-convex objectives, meaning that models produced by different training runs, in general, converge to different solutions. It is clear that these trained models correspond to distinct functions, but is this a distinction without a difference? Models with a similar objective value and validation/test accuracy may still differ significantly.

In practice, models are often retrained as new data arrives. This necessitates algorithmic and architectural changes in state-of-the-art models to improve their performance on new data. The run-to-run variability in training models makes it difficult to conclude if a specific initialization or hyperparameter made a meaningful difference in model performance or if it just "got lucky" due to the presence of randomness in the optimization. Without this knowledge, comparing training configurations to assess relative quality becomes difficult.

We consider a stylized model of a machine learning training process. A practitioner trains M models in an identical manner, using fresh randomness for each training run, but they must output only a single model (due to computational constraints in deployment, perhaps). If test accuracy is the criterion they use, then they may have many models with nearly the same test accuracy. This work proposes a new framework to measure how representative a given model is of the training process.

Gundersen et al. [1] provided a taxonomy of barriers to reproducibility in machine learning practice: they identified randomness in model initialization, batch shuffling during mini-batch stochastic gradient descent (SGD), and data sampling as major sources of variability during training. There are two sources of randomness in this framing of the problem: the sampling randomness of the test data, and the randomness from the stochastic optimization. The latter is under the control of the pracitioner. The importance of using random seeds as part of hyperparameter tuning in deep neural network (DNN) training has been previously highlighted in the literature [2]–[6]: these works found the effect of random seed on model performance to be statistically significant. Other works have focused on understanding the random seed effect on specific sources of randomness in training. For example, Fort et al. [7] found that initialization has a larger effect than batch order in SGD on model performance, while Bouthillier et al. [8] showed the opposite. Summers and Dinneen [9], and Jordan [10] show that networks converge to vastly different values even with the change of a single bit of network parameters during initialization. While studying run-to-run variability in pre-trained BERT models over different random seeds, Dodge et al. [11] noted a validation performance gain of 7% over previously reported results, highlighting the importance of fine-tuning over the random seed. All the above-mentioned works have assessed the impact randomness has on the *validation accuracy*, or *churn* [12] between models, which quantifies how

Manuscript received June 28, 2024; revised March 24, 2025.

This work was partially supported by the Statistical Inference Generates kNowledge for Artificial Learners (SIGNAL) program at Pacific Northwest National Laboratory (PNNL), as well as by the Mathematics for Artificial Reasoning in Science (MARS) initiative via the Laboratory Directed Research and Development (LDRD) Program at PNNL. PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL0-1830.

S. Banerjee and A. D. Sarwate are with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, Piscataway, NJ 08854, USA. Email: {sb1977, anand.sarwate}@rutgers.edu. T. Marrinan and R. Cannon are with the Pacific Northwest National Lab, Richland, WA 99352, USA. Email: {timothy.marrinan, reilly.cannon}@pnnl.gov. T. Chiang is affiliated with the Pacific Northwest National Laboratory and ARPA-H, Bethesda, Maryland, 20892. Email: tony.chiang@arpa-h.gov.

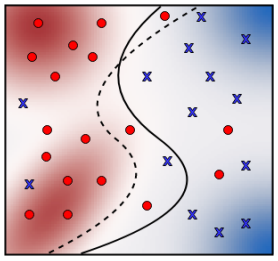

Fig. 1. Hypothetical decision boundaries corresponding to two models with the same accuracy. The shaded regions represent the underlying data distribution.

two models differ in their predictions on the same test point. These summary statistics only focus on the *decisions* made by predictive (classification) models and do not directly assess differences in the functions learned by these models.

In this paper, we assess DNN training variability over random seeds using the *network outputs used to make the decision*. Figure 1 illustrates the difference: the solid and dashed lines represent two decision boundaries between red (circle) and blue (cross). Accuracy measures incorrect decisions, and the churn is determined by the region between the two curves. If we think of the training algorithm as generating a random sample from a function space, we can use other tools to understand model variability. As a first step, we can examine the distribution of the pre-thresholded outputs (the *logit gap*) from functions learned by different runs of a fixed DNN architecture. To that end, we adopt a non-parametric hypothesis testing framework to measure the similarity of functions learned by DNN models that vary in random seeds.

II. Problem Setup

Notation: For a positive integer n, let $[n] = \{1, 2, \ldots, n\}$. Let $\mathbb{1}(.)$ denote the indicator function, such that $\mathbb{1}(x \le t) = 1$ if $x \le t$, and 0 otherwise. Random variables will be denoted in boldface, with realizations being non-bolded, so $\boldsymbol{\theta}$ is a random variable and θ is a realization. We consider training a DNN for predicting an output taking values in a set $\mathcal{Z}$ with input from a (feature vector) x taking values in a space $\mathcal{X}$. A DNN with a given architecture is specified by a set of parameters (e.g. the weights) θ taking values in parameter space, Θ.

A. The learning task

We will restrict attention to *binary classification* using a DNN with parameters θ. Given a label space $\mathcal{Y}$, a training algorithm takes a training set, $\mathcal{D}_{\text{train}} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} \colon i \in [N_{\text{train}}]\}$, estimates the weights θ that (approximately) minimize the empirical loss $\hat{R}(\theta; \mathcal{D}_{\text{train}})$ over the training data, and assigns a label $y \in \mathcal{Y}$ using a *softmax* operation. We model this by assuming the DNN takes an input x and computes functions $m^+(x \mid \theta)$ and $m^-(x \mid \theta)$, with the predicted label $\hat{y}(x; \theta)$ being $+1$ if $m^+(x \mid \theta) \ge m^-(x \mid \theta)$ and -1 otherwise. A Bayesian interpretation of this rule assumes the data is generated according to an (unknown) distribution $\pi(x, y)$, with a likelihood function $\bar{\pi}(x|y)$, and a uniform prior $\text{Unif}[\mathcal{Y}]$. The functions $m^+(x \mid \theta)$ and $m^-(x \mid \theta)$ for the positive and negative classes can be converted into posterior probability estimates:

$$\hat{\pi}(y = 1 \mid x) = \frac{\exp(m^+(x \mid \theta))}{\exp(m^+(x \mid \theta)) + \exp(m^-(x \mid \theta))}. \quad (1)$$

$$\hat{\pi}(y = -1 \mid x) = \frac{\exp(m^-(x \mid \theta))}{\exp(m^+(x \mid \theta)) + \exp(m^-(x \mid \theta))}. \quad (2)$$

The prediction function is then the maximum *a posteriori* (MAP) estimate:

$$\begin{aligned} \hat{y}(x; \theta) &= \text{sgn}(\hat{\pi}(y = 1 \mid x) - \hat{\pi}(y = -1 \mid x)) \\ &= \text{sgn}\big(m^+(x \mid \theta) - m^-(x \mid \theta)\big). \end{aligned} \quad (3)$$

We can then assume the learned function is $m(x; \theta) = m^+(x \mid \theta) - m^-(x \mid \theta)$ which is an approximation to the (unknown) log-likelihood ratio $\log \frac{\bar{\pi}(x|y=1)}{\bar{\pi}(x|y=-1)}$. We therefore refer to $m^+(x \mid \theta)$ and $m^-(x \mid \theta)$ as *logits* and $m(x; \theta)$ as the *logit gap*. In our setting, for a given $\theta \in \Theta$, the network computes the function $m(x; \theta)$ (logit gap function), and the DNN defines a family of functions $\mathcal{M} = \{m(x; \theta) : \mathcal{X} \to \mathcal{Z} \colon \theta \in \Theta\}$. The goal of *training* a DNN is to find a "good" setting for the parameters θ or, equivalently, to find a "good" function $m \in \mathcal{M}$.

Almost all DNN training algorithms use *stochastic optimization*, making them approximate in two ways. First, they will generally converge to a local minimum because the risk minimization problem is non-convex. Second, randomization means the estimated parameters $\boldsymbol{\theta}$ are random variables. Two runs of the same training algorithm on the same training set $\mathcal{D}_{\text{train}}$ can produce different functions. One natural question is to ask how different these functions are. We can try to answer this using a *test set*, $\mathcal{D}_{\text{test}} = \{(x_j, y_j) \colon j \in [N_{\text{test}}]\}$. If we assume $\mathcal{D}_{\text{test}}$ is drawn i.i.d. from the data distribution π, then given a trained model $m(x; \theta)$, the set of values $\{m(x_j; \theta) \colon j \in [N_{\text{test}}]\}$ is also an i.i.d. sample from a distribution on $\mathbb{R}$ induced by π.

B. Trained models and reference functions

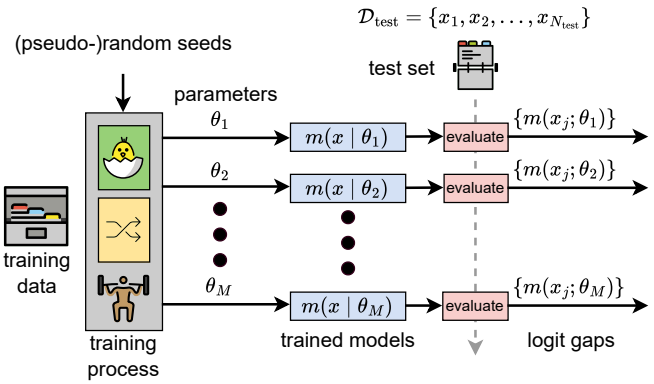

Fig. 2. The experiment design. Randomness is used within the training algorithm for initialization and batch selection. We train (or fine-tune) M models independently using the same training data. Each model is then evaluated on the test set. The resulting values are used to form empirical CDFs.

We consider the experiment shown in Figure 2: we train M models independently by varying the random seed and using

the same training set. We then evaluate the corresponding models on the test set. Let τ be the distribution of $\boldsymbol{\theta}$ corresponding to the randomness in the training algorithm run on data set $\mathcal{D}_{\text{train}}$. Then $\mathcal{D}_{\text{param}} = \{\theta_1, \theta_2, \ldots, \theta_M\}$ is sampled i.i.d. $\sim \tau$. Let $\mathcal{P}$ be the set of cumulative distribution functions (CDFs) on $\mathbb{R}$. Given a fixed DNN architecture, the training set $\mathcal{D}_{\text{train}}$, and a stochastic training algorithm, the M i.i.d. samples of parameters induce M i.i.d. samples of functions $\{m(x;\theta_k) : k \in [M]\}$, taking values in $\mathcal{M}$.

Given a realization of a parameter θ_k and $\mathbf{x} \sim \pi$, we define the CDF of the model $m(\mathbf{x};\theta_k)$ by

$$G_k(t) = \mathbb{E}_{\pi|\theta_k}[\mathbb{1}(m(\mathbf{x};\theta_k) \leq t)] \tag{4}$$

Now, suppose $\boldsymbol{\theta} \sim \tau$ and $\mathbf{x} \sim \pi$ are drawn independently. The expected CDF over both is

$$F_{\tau\times\pi}(t) = \mathbb{E}_{\tau\times\pi}[\mathbb{1}(m(\mathbf{x};\boldsymbol{\theta}) \leq t)]. \tag{5}$$

Since we do not know $\tau \times \pi$, we can consider an approximate empirical version of $F_{\tau\times\pi}$ by conditioning (5) on the sampled parameters $\mathcal{D}_{\text{param}}$. The result is the following function:

$$\begin{aligned}\bar{F}_{\pi|\mathcal{D}_{\text{param}}}(t) &= \mathbb{E}_{\pi|\mathcal{D}_{\text{param}}}\left[\frac{1}{M}\sum_{k=1}^{M}\mathbb{1}(m(\mathbf{x};\theta_k) \leq t)\right] \\ &= \frac{1}{M}\sum_{k=1}^{M}\mathbb{E}_{\pi|\mathcal{D}_{\text{param}}}[\mathbb{1}(m(\mathbf{x};\theta_k) \leq t)], \\ &= \frac{1}{M}\sum_{k=1}^{M}\mathbb{E}_{\pi|\theta_k}[\mathbb{1}(m(\mathbf{x};\theta_k) \leq t)], \end{aligned} \tag{6}$$

by the linearity of expectation. Note that the terms inside the sum are the CDFs in (4).

C. Empirical CDFs, reference functions, and ensembles

We can only access data distribution π through samples from $\mathcal{D}_{\text{test}}$. In our subsequent hypothesis tests, we will use empirical cumulative distribution functions (eCDFs) in the test statistics. Given N samples from π we can compute the eCDF for a model with parameter θ_k

$$\hat{G}_k(t) = \frac{1}{N}\sum_{j=1}^{N}\mathbb{1}(m(x_j;\theta_k) \leq t). \tag{7}$$

We can average the eCDFs from M models to form what we call the *reference function*

$$\hat{\bar{G}}(t) = \frac{1}{M}\sum_{k=1}^{M}\left(\frac{1}{N}\sum_{j=1}^{N}\mathbb{1}(m(x_j,\theta_k) \leq t)\right). \tag{8}$$

In our framework, we will also consider *ensemble models* in which we average the logit gaps. If we take a subset $(\theta'_1, \theta'_2, \ldots, \theta'_{M_{\text{ens}}})$ of M_{ens} models from our M models, the corresponding ensemble is

$$\bar{m}(x_j) = \frac{1}{M_{\text{ens}}}\sum_{k=1}^{M_{\text{ens}}} m(x_j,\theta'_k). \tag{9}$$

The eCDF of this ensemble model is

$$\hat{\bar{H}}'(t) = \frac{1}{N}\sum_{j=1}^{N}\mathbb{1}(\bar{m}(x_j) \leq t). \tag{10}$$

We are interested in testing whether a new model with parameter θ_0 is close to the expected model from the training process. We formulate this using hypothesis tests that compare G_0 corresponding to θ_0 as defined in (4) with the average in $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$. By switching on or off different uses of randomization in training, we induce different distributions τ on the parameters. For example, we can use deterministic initialization or fixed batch ordering. Under these scenarios, we can generate M models and analyze the variability of these models using the reference function (8), which captures the consensus of the collection of trained models.

Remark 1: While we have described the problem in this section for binary classifiers and the logit gap, note that this formulation can be used for any machine learning model by taking a single scalar measurement function $m\colon \mathcal{X} \times \Theta \to \mathbb{R}$ applied to $\mathcal{D}_{\text{test}}$. For binary classifiers, the logit gap is a natural choice and makes the comparison to the validation accuracy more interpretable.

III. Robust and non-parametric testing

Ideally, we want to estimate the variability of trained models by measuring the discrepancy between a candidate model, G_0, and the expected CDF $F_{\tau\times\pi}$ defined in (5). Without access to $\tau \times \pi$, we instead consider the CDF, $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$, averaged over $\mathcal{D}_{\text{param}}$ as defined in (6). We formulate this as a a one-sided hypothesis test: the null hypothesis is that G_0 is the same as $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$.

A. Classical one- and two-sample KS tests

We can use non-parametric hypothesis testing to formulate the comparison between G_0 and $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$. Given N samples, $\{x_j : j \in [N]\} \sim \pi$, we can compute the logit gaps, $\{m(x_j;\theta_0)\colon j \in [N]\}$. The null hypothesis for the one-sided test is

$$\mathcal{H}_0^{\text{KS1}}\colon \{m(x_j;\theta_0)\colon j \in [N]\} \sim \bar{F}_{\pi|\mathcal{D}_{\text{param}}}. \tag{11}$$

That is, is the candidate, G_0, the same as $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$? The classical Kolmogorov-Smirnov (KS) test uses the eCDF, $\hat{G}_0$, defined in (7) to compute the test statistic, $\|\bar{F}_{\pi|\mathcal{D}_{\text{param}}} - \hat{G}_0\|_\infty$.

The KS test statistic cannot be evaluated without a closed-form expression for $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$. However, we can still employ a two-sample KS test, which leads us to the null hypothesis

$$\mathcal{H}_0^{\text{KS2}}\colon G_0 = \bar{F}_{\pi|\mathcal{D}_{\text{param}}}. \tag{12}$$

Unfortunately, this hypothesis also relies on explicit knowledge of $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$. To get around this, we propose using $\hat{\bar{G}}$ as a proxy for $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$. In this case we draw $2N$ samples $\{x_j : j \in [2N]\} \sim \pi$ and use half to compute the reference function $\hat{\bar{G}}$ in (8) and half to compute $\hat{G}_0$ and use the test statistic $\|\hat{\bar{G}} - \hat{G}_0\|_\infty$. The following result shows that $\hat{\bar{G}}$ is a reasonable proxy for $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$.

Theorem 1: Let $\hat{\bar{G}}$ and $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$ be given by (8) and (6). Then for any $\delta_b > 0$,

$$\mathbb{P}_{\pi|\mathcal{D}_{\text{param}}}\left(\left\|\bar{F}_{\pi|\mathcal{D}_{\text{param}}} - \hat{\bar{G}}\right\|_\infty > \delta_b\right) \leq \epsilon_b, \tag{13}$$

where $\epsilon_b = 2M\exp(-2N\delta_b^2)$.

The proof is in Section A of the Appendix and follows from the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality [13] and a union bound over $\{\theta_j\}$.

We can use the two-sample DKW inequality [14] to set a threshold for the test. The DKW inequality shows that

$$\mathbb{P}_{\pi|\mathcal{D}_{\text{param}}}\left(\left\|\hat{G}_0 - \hat{\bar{G}}\right\|_\infty > \delta_a\right) \leq \epsilon_a, \tag{14}$$

where $\epsilon_a = C\exp(-N\delta_a^2)$. Given a target ϵ_a we can set the threshold for the test as

$$\delta_a = \sqrt{\frac{1}{N}\ln\frac{C}{\epsilon_a}}. \tag{15}$$

Here $C = e$ for general N and for $N \geq 458$ we can take $C = 2$ [14, Theorem 1]. Using the the triangle inequality, we have the following corollary.

Corollary 1: Under the assumptions of Theorem 1, for any $\delta_a > 0$, $\delta_b > 0$ and sample size $N \geq 458$, we have

$$\begin{aligned}\mathbb{P}_{\pi|\mathcal{D}_{\text{param}}}&\left(\left\|\bar{F}_{\pi|\mathcal{D}_{\text{param}}} - \hat{G}_0\right\|_\infty \leq \delta_a + \delta_b\right)\\ &\geq 1 - 2M\exp\left(-2N\delta_b^2\right) - 2\exp(-N\delta_a^2).\end{aligned} \tag{16}$$

Provided we have enough samples, we can set thresholds for this test to achieve a guaranteed false alarm (type I error) probability. However, in large sample settings, the KS-test often rejects the null because even small changes in the sample can result in a significant shift in the L_∞-norm [15, p.245].

B. Robust statistics and trimming

Modern machine learning models are complex and often have many more parameters than training points. Even if we use a very large number of models, M, to compute $\hat{\bar{G}}$ and estimate $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$, we may not have a good approximation to the expected CDF, $F_{\tau\times\pi}$. We want to represent this epistemic uncertainty and alleviate the over-sensitivity of the KS test in a more structured way. We can capture some of this uncertainty in the null hypothesis by replacing the null hypothesis with a composite hypothesis: this is the approach taken in *robust statistics* [16].

We describe the approach for general distributions and later specialize it to our setting. Consider a given distribution P_0 and define a new null hypothesis as a set of distributions close to P_0. Given a metric or divergence $d(\cdot,\cdot)$, radius α, and distribution P_0, define the d-ball of radius α by $\mathcal{B}_d(P_0,\alpha) = \{P \in \mathcal{P} : d(P_0,P) \leq \alpha\}$. While we could take $d(\cdot,\cdot)$ to be any metric or divergence between probability distributions, the classical approach in robust statistics [16] uses an L_1 ball corresponding to a *contamination neighborhood*

$$\mathcal{B}_1(P_0,\alpha) = \{P : P = (1-\alpha)P_0 + \alpha Q,\ Q \in \mathcal{P}\}. \tag{17}$$

The interpretation is that up to an α-fraction of samples may come from an unknown "outlier" distribution Q.

In this paper, we use the L_1 ball not only because it is the default in robust statistics but also because of the connection to *impartial trimming* [17]. Given a distribution $P_1 \in \mathcal{P}$ and a scalar $\alpha \in [0,1]$, the set of α-trimmings of P_1 is defined by,

$$\mathcal{R}_\alpha(P_1) = \left\{P \in \mathcal{P}\colon P \ll P_1,\ \frac{dP}{dP_1} \leq \frac{P_1}{(1-\alpha)}\ \text{a.s.}\right\}, \tag{18}$$

where $P \ll P_1$ denotes that P is absolutely continuous with respect to P_1 [18]. Trimmings are related to contamination neighborhoods [19]:

$$P_1 \in \mathcal{B}_1(P_0,\alpha) \Leftrightarrow P_0 \in \mathcal{R}_\alpha(P_1), \tag{19}$$

A recent work by del Barrio, Inouzhe, and Matrán [20] formulated a robust hypothesis test using the connection between contamination and trimming. Given a sample generated from an unknown distribution P, their null hypothesis[1] is:

$$\mathcal{H}_0^{\text{BIM}}\colon P \in \mathcal{B}_1(P_0,\alpha), \tag{20}$$

where P_0 is a known distribution. Because of the connection to α-trimming in (19), this hypothesis is equivalent to stating $P_0 \in \mathcal{R}_\alpha(P)$. They, therefore, take the following as their null hypothesis:

$$\mathcal{H}_0^{\text{BIM}}\colon \inf_{\tilde{P}\in\mathcal{R}_\alpha(P)} \|P_0 - \tilde{P}\|_\infty = 0. \tag{21}$$

Given a sample of N points and its empirical eCDF $\hat{P}_N$, they propose the test statistic:

$$T(\hat{P}_N) = \inf_{\tilde{P}\in\mathcal{R}_\alpha(\hat{P}_N)} \|P_0 - \tilde{P}\|_\infty. \tag{22}$$

This test statistic involves finding the closest L_∞ approximation to P_0 in the set of α-trimmings of the eCDF $\hat{P}_N$. This change is important because test statistics using the contamination-based null in (20) could entail an optimization over the L_1 ball $\mathcal{B}_1(P_0,\alpha)$, which would be infinite dimensional problem, wheras optimization over the set of α-trimmings in (22) is a finite dimensional problem.

Computing the optimizer of (22) in the set of trimmings of $\hat{P}_N$ involves finding a reweighting of the samples such that downplaying the importance of a small fraction of contaminated samples (from the test set) would allow a KS-test to not reject the null hypothesis. We point the readers to Section B-A, and B-B in the Appendix for a detailed description.

C. A new robust two-sample test

To apply the trimming-based approach in our problem, we could take $P_0 = \bar{F}_{\pi|\mathcal{D}_{\text{param}}}$ and test using the eCDF $\hat{P}_N = \hat{G}_0$ computed from N samples. Because $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$ is not known, we cannot compute the test statistic in (22). In what follows, we assume that the support S of the distributions is bounded[2] with length $|S|$: this assumption will let us set the threshold in our test.

[1]The superscript is the initials of del Barrio, Inouzhe, and Matrán [20].

[2]While logit gaps can in general take any values in $(-\infty,\infty)$, in practice, logit gaps of well-trained models are not too large. Large logit gaps would indicate a poorly calibrated model or the model being overconfident in one class. In our experiments we clip the logit gaps.

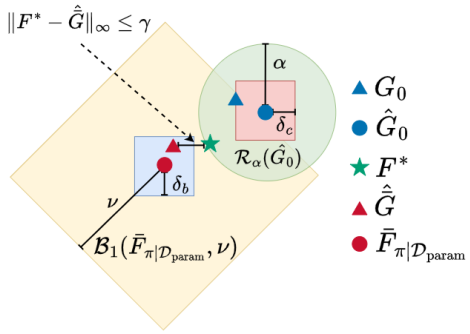

Fig. 3. Illustration of the parameters in the trimming-based two-sample test. The null hypothesis uses the L_1-contamination ball, $\mathcal{B}_1(\bar{F}_{\pi|\mathcal{D}_{\text{param}}}, \nu)$. The test will accept if the L_∞ distance between $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$ and the set of α-trimmings of $\hat{G}_0$, $\mathcal{R}_\alpha(\hat{G}_0)$, is small.

We propose using the following null hypothesis in a two-sample test:

$$\mathcal{H}_0 \colon G_0 \in \mathcal{B}_1(\bar{F}_{\pi|\mathcal{D}_{\text{param}}}, \nu). \tag{23}$$

As before, we need to find a test statistic and threshold using $2N$ samples from π. We use N samples to compute $\hat{\tilde{G}}$ and N samples to compute $\hat{G}_0$. The DKW inequality shows that for any $\delta_c > 0$,

$$\|G_0 - \hat{G}_0\|_\infty \leq \delta_c \tag{24}$$

with probability $1 - 2e^{-2N\delta_c^2}$.

From the bounds in (24) and (13) and our assumption on the support, we have the following bounds:

$$\|G_0 - \hat{G}_0\|_1 \leq |S|\delta_c \tag{25}$$

$$\|\bar{F}_{\pi|\mathcal{D}_{\text{param}}} - \hat{\tilde{G}}\|_1 \leq |S|\delta_b \tag{26}$$

with probability $1-2e^{-2N\delta_c^2}$ and $1-2Me^{-2N\delta_b^2}$, respectively. The decision rule we propose is to accept the null if

$$\min_{\tilde{F} \in \mathcal{R}_\alpha(\hat{G}_0)} \|\tilde{F} - \hat{\tilde{G}}\|_\infty \leq \gamma. \tag{27}$$

This decision rule finds the best L_∞ approximation of the reference $\hat{\tilde{G}}$ in the α-trimmings of the candidate eCDF $\hat{G}_0$.

We want to understand the relationship between the contamination level ν of the null, the trimming level α, and the threshold γ. Let F^* be the minimizer of (27):

$$F^* = \operatorname*{argmin}_{\tilde{F} \in \mathcal{R}_\alpha(\hat{G}_0)} \|\tilde{F} - \hat{\tilde{G}}\|_\infty. \tag{28}$$

Suppose the null is accepted by (27) so that $\|F^* - \hat{\tilde{G}}\|_\infty \leq \gamma$. This implies

$$\|F^* - \hat{\tilde{G}}\|_1 \leq |S|\gamma. \tag{29}$$

By (19), $F^* \in \mathcal{R}_\alpha(\hat{G}_0)$ means $\hat{G}_0 \in \mathcal{B}_1(F^*, \alpha)$, so

$$\|F^* - \hat{G}_0\|_1 \leq \alpha. \tag{30}$$

Thus by the triangle inequality,

$$\|\hat{G}_0 - \hat{\tilde{G}}\|_1 \leq \alpha + |S|\gamma. \tag{31}$$

Combining this with the two high probability inequalities from (25) and (26) we have

$$\|\bar{F}_{\pi|\mathcal{D}_{\text{param}}} - G_0\|_1 \leq \alpha + |S|(\gamma + \delta_b + \delta_c). \tag{32}$$

with probability $1 - \varepsilon$, where

$$\varepsilon \leq 2e^{-2N\delta_c^2} + 2Me^{-2N\delta_b^2}. \tag{33}$$

For a fixed threshold γ, we can estimate the trimming level, α, needed for the L_∞ distance in (27) to be less than or equal to the threshold. Consequently, for a fixed test threshold γ, and with enough samples to make δ_b and δ_c small, we can estimate the contamination level ν, in the contamination neighborhood $\mathcal{B}_1(\bar{F}_{\pi|\mathcal{D}_{\text{param}}}, \nu)$ of $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$, for which the robust two-sample hypothesis test would fail to reject the null with high probability ϵ. We use this in the next section to propose a new measure for analyzing model variability.

IV. Metrics to analyze model variability

One of our goals is to use the framework from the robust two-sample test to design a measure of model discrepancy/variability. As we show in the experiments, this new measure can be more informative than existing measures.

A. A new trimming-based metric

In this section, we introduce how our trimming-based two-sample test can be used to define a new measure of discrepancy between a model with parameter θ_0 and the reference model formed using $\theta_1, \theta_2, \ldots, \theta_M$. In Algorithm 1, we estimate the smallest α such that the test in (27) accepts. We do this by running the test B times using bootstrap resampling from $\mathcal{D}_{\text{test}}$. In each resampling we increase the trimming level α until the test accepts. This is equivalent to finding a contamination level ν for the null such that the test accepts. The output $\hat{\alpha}$ is the average over the α values from each resampling and is our proposed measure of discrepancy between model $\hat{G}_0$ and $\hat{\tilde{G}}$.

We compute $\hat{\tilde{G}}$ and $\hat{G}_0$ by sampling uniformly from $\mathcal{D}_{\text{test}}$ of size $N_{\text{test}} = 2N$. Let $\mathcal{Z} = \{z_j\}_{j=1}^{2N}$ be the combined ordered arrangement of $2N$ logit values from $\hat{\tilde{G}}$ and $\hat{G}_0$. For a fixed α, we calculate the α-trimming of $\hat{G}_0$ that minimizes the L_∞-distance to $\hat{\tilde{G}}$. We then check to see if this trimming is within the L_∞ ball around $\hat{\tilde{G}}$ of radius γ:

$$\max_{z \in \mathcal{Z}} \min_{\tilde{F} \in \mathcal{R}_\alpha(\hat{G}_0)} \left| \tilde{F}(z) - \hat{\tilde{G}}(z) \right| \leq \gamma, \tag{34}$$

We set our threshold to be $\gamma = \delta_a$ by fixing the probability ϵ_a in (15), which follows from the two-sample DKW inequality introduced in (14).

A small $\hat{\alpha}$ for a candidate model G_0 implies the level of trimming needed for it to be not rejected, meaning that it is close to $\hat{\tilde{G}}$, which in turn is close to $\hat{G}_0$. In Section V, we compare $\hat{\alpha}$ to metrics like accuracy and churn that are commonly used to analyze model variability. We generate multiple trained models from the randomization in the training process. We can look at each trained model as a potential candidate and use the rest to compute a reference.

Algorithm 1 Estimate $\hat{\alpha}$ measure

Input: Test set $\mathcal{D}_{\text{test}}$, trained models $\{m_k \colon k \in [M]\}$, candidate model m_0, threshold δ_a, trimming levels $\{\alpha_t\}_{t=1}^{T}$, bootstrap sampling number B.
Output: trimming level estimate $\hat{\alpha}$.
  for $b = 1$ to B **do**
    Compute $\hat{\tilde{G}}$ in (8) using $\{m_k \colon k \in [M]\}$, and $\hat{G}_0$ in (7) using m_0 from two sets of N samples resampled from $\mathcal{D}_{\text{test}}$.
    Set $\mathcal{Z} = \{z_j\}_{j=1}^{2N}$ be the ordered set of logit values from $\hat{\tilde{G}}$ and $\hat{G}_0$.
    Reject $\leftarrow 1$, $\hat{\alpha} \leftarrow 0$, $t \leftarrow 1$
    while Reject $= 1$ and $t \leq T$ **do**
      $\alpha \leftarrow \alpha_t$
      Use $\mathcal{Z}$ to compute (34)
      if Test (34) accepts **then**
        Reject $\leftarrow 0$,
        $\hat{\alpha}_b \leftarrow \alpha$
      else
        $t = t + 1$
      end if
    end while
  end for
  $\hat{\alpha} \leftarrow \frac{1}{B} \sum_{b=1}^{B} \hat{\alpha}_b$

B. Other metrics for model variability

The *accuracy* of a model is

$$A(\theta) = \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}(\hat{y}(x_j; \theta) = y_j). \tag{35}$$

The *churn* is defined by

$$C(\theta_1, \theta_2) = \sum_{j=1}^{N} \mathbb{1}(\hat{y}(x_j; \theta_1) \neq \hat{y}(x_j; \theta_2)), \tag{36}$$

which is the number of test points where the models disagree. Both accuracy and churn focus on the *predictions* made by models and do not use information about the logit gap function $m(x; \theta_k)$ beyond its sign. Looking at $m(x; \theta_k)$ directly gives us other approaches to assess whether models are similar or not: two models may have similar accuracy and low churn but can have very different logit gaps.

We also consider a third metric for the qualitative assessment of these models, the Expected Calibration Error (ECE) [21]. The ECE is a summary statistic of model calibration which measures the difference in accuracy and expected confidence and is obtained by partitioning the predictions into R equally spaced bins B_r,

$$\text{ECE}(\theta) = \sum_{r=1}^{R} \frac{|B_r|}{N} |A(B_r; \theta) - \text{CONF}(B_r; \theta)|, \tag{37}$$

where

$$A(B_r; \theta) = \frac{1}{|B_r|} \sum_{j=1}^{|B_r|} \mathbb{1}(\hat{y}(x_j; \theta) = y_j) \tag{38}$$

$$\text{CONF}(B_r; \theta) = \frac{1}{|B_r|} \sum_{j=1}^{|B_r|} \hat{\pi}(y_j = \hat{y}(x_j; \theta) | x_j) \tag{39}$$

A perfectly calibrated model will have $A(B_r; \theta) = \text{CONF}(B_r; \theta)$. Although Niculescu-Mizil et al. [22] noted that while DNNs are well-calibrated on binary classification tasks, it is impossible for a model to achieve perfect calibration in reality, making this another useful metric to assess the quality of models produced through different random seeds in conjunction with accuracy.

V. Experiments

Our proposed measure of model closeness/discrepancy offers new insight into questions around neural network training. In this section, we perform a series of illustrative experiments to demonstrate the utility of this measure:

- In Section V-A1, we show that the eCDF of a deep ensemble model reliably approximates the reference as the size of the pool of candidate models in the ensemble increases. Establishing this relationship at the very onset allows us to qualitatively compare candidate models with the deep ensemble substitute of our reference function, while comparing candidate models to the reference function through the robust hypothesis test.
- In Section V-A2, we use our proposed measure of model closeness $\hat{\alpha}$ to show the minimum number of candidate models needed to form a reliable deep ensemble that has less variability in different performance metrics and approximates our reference function well.
- In Section V-B, and V-C, we connect the proposed discrepancy measure to other metrics used to measure network variability. We show how $\hat{\alpha}$ is more informative than validation accuracy alone through two case studies. The first uses a small CNN to perform a binary classification task on the CIFAR-10 dataset. The second uses a ViT variant, pre-trained on ImageNet, to perform the same task on the same dataset. The former case study allowed us to use neural networks with relatively few parameters, so we can test the limits of the expressivity of the proposed test and the latter to show how the same test applies to large-scale pre-trained models as well.

For experiments in Section V-A, and V-B, we chose the following experimental setup. We used a small convolutional neural network with two convolutional layers (having 32 and 16 features, respectively, with a 3×3 kernel size) followed by one hidden layer of 64 units and a final layer of 2 units that output the raw logits of the network. Details of the network architecture and parameters are included in Section C of Appendix. We train this network on a subset of the CIFAR-10 dataset [23], under the following settings:

- Of the 10 classes, we use 8 to create a binary classification problem by merging them into two super groups.

Class 1 is comprised of airplane, automobile, ship, truck, and class -1 is comprised of bird, dog, frog and horse.
- The training size is $N_{\text{train}} = 40000$ and test size is $N_{\text{test}} = 8000$.
- We fix all hyperparameters other than the random seeds by training all models for 50 epochs, with a fixed learning rate of 0.001, and fix the batch size to 32.

We chose a small example to allow us to train many models so that we can explore model training in different scenarios. The authors acknowledge the use of high-performance computing resources provided by the Office of Advanced Research Computing (OARC), at Rutgers, for running the experiments in this paper [24]. Code to compute the test statistic and $\hat{\alpha}$ metric of the two-sample version of the robust KS-test has been made available online [25].

A. Comparing the reference function and a deep ensemble

This section demonstrates how the eCDF of a deep ensemble model $\hat{\bar{H}}'$, defined in (10), closely approximates our reference function $\hat{\bar{G}}$, defined in (8). Establishing this relation will allow us to compare candidate models with the deep ensemble substitute of the reference function using different performance metrics like validation accuracy, churn w.r.t. a deep ensemble, and the ECE, while comparing the eCDFs of candidate models to the reference function through the robust KS-test to analyze model variability. A lower ECE indicates a better-calibrated model, while lower churn w.r.t. an ensemble indicates less disagreement between candidate models and an ensemble. We point the readers to Section IV-B for a formal definition of these metrics. We train a total of 1600

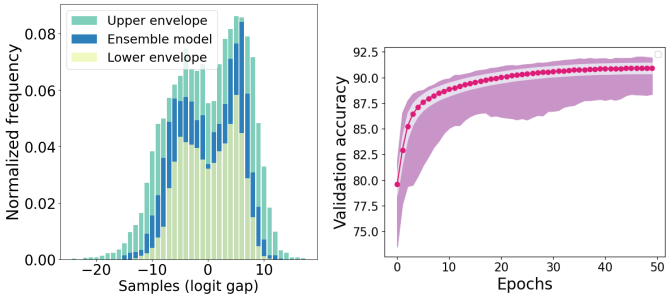

Fig. 4. (Left) Histogram of logit gaps from the ensemble model with the upper and lower envelopes representing the maximum and minimum probability attained in each bin among individual candidate models. (Right) A plot showing the evolution of validation accuracy of CNN models over different epochs. The solid red dots represent the mean validation accuracy over 800 seeds at each epoch, the light-colored area denotes one standard deviation, and the purple area represents the maximum and minimum values at that epoch.

models by randomly fixing a seed for each model. The seeds control both initialization and batch order during SGD. We use the first $M = 800$ models to create a reference function $\hat{\bar{G}}$ and set the rest $M' = 800$ models to create the eCDF of a deep ensemble $\hat{\bar{H}}'$. We choose different ensemble sizes, $M_{\text{ens}} \in [3, 5, 10, 20, 30, 70, 100, 150, 200]$, to observe the variability of ensembles in different metrics and to understand how large of an ensemble we need to approximate the reference function well. For each value of M_{ens}, we sample M_{ens} models without replacement from the remaining $M' = 800$ models and compute $\hat{\bar{H}}'$. We repeat this experiment 500 times to get 500 ensemble models using M_{ens} components for each value of M_{ens}. Deep ensemble predictors have been widely used in the literature to reduce variability in DNN models [26], [27]. Candidate models will have varying degrees of certainty on individual test points. Taking the average of model "confidences" across independent training runs makes them closer to their expected values, lowering variability. Although ensembling through averaging over softmax probabilities is common practice in the literature, averaging over logits has also been investigated to address the shortcomings of probability averaging [28], [29].

Figure 4 (Left) shows how the logit gap samples obtained from ensembling $M_{\text{ens}} = M'$ candidate models compare with candidate models in the pool. The ensemble model produces fewer samples with small logit gaps (samples with higher uncertainty) and large logit gaps (overconfident samples).

Figure 4 (Right) shows the evolution of validation accuracy of candidate models over epochs. The solid red dots in the plot correspond to the mean accuracy over M' seeds at each epoch, the light-colored region corresponds to one standard deviation, and the purple region corresponds to the minimum and maximum accuracy at that epoch. As observed in the plot, validation accuracy stops increasing from epoch 30 onwards, hence the decision to stop training at epoch 50, which is well past this optimization convergence. The same strategy was adopted by Picard [6, Section 4.1] who discusses it in more depth.

1) Closeness of a deep ensemble to the reference function: The first question we want to answer is **how quickly does the eCDF of an ensemble model approximate the chosen reference?** In other words, as M_{ens} approaches M' how quickly does $\left\|\hat{\bar{G}} - \hat{\bar{H}}'\right\|_\infty$ decrease? To answer this, we compute the L_∞-distance between $\hat{\bar{G}}$ and $\hat{\bar{H}}'$ and compare it to the threshold set by the test in (14). To set the threshold δ_a in (15), we fix $\epsilon_a = 0.01$.

Figure 5 demonstrates this convergence behavior. The eCDF of all ensemble models $\hat{\bar{H}}'$, produced with $M_{\text{ens}} = 100, 150,$ and 200 candidate models have L_∞-distances from $\hat{\bar{G}}$ less than the threshold set for our KS-test. This empirical evidence indicates that a deep ensemble eCDF more closely resembles the reference function $\hat{\bar{G}}$ as M_{ens} approaches M'. Therefore, we expect any candidate model close to our reference function $\hat{\bar{G}}$, formed with M candidate models, to be also close to a deep ensemble, formed with M_{ens} candidate models, in terms of different performance metrics, provided M_{ens} approaches $M' = M$.

Since the benefits of ensembling are observed even for pool sizes as small as $M_{\text{ens}} = 5$ [26], it might be tempting to focus on a few random seeds to generate a deep ensemble model due to computational constraints. This brings us to the next question: **Does averaging over a small number of candidate models to create an ensemble model result in a reliable representative of the training procedure?** We try to understand this qualitatively. We investigate three metrics of ensemble models as M_{ens} approaches M': validation accuracy,

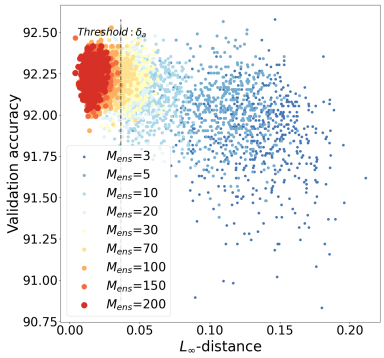

Fig. 5. A plot showing L_∞-distance of the eCDF of ensemble models $\hat{\bar{H}}'$ (formed with M_{ens} models) w.r.t. the reference $\hat{\bar{G}}$ (formed with M models) against validation accuracy of ensemble models. For each value of M_{ens}, we choose M_{ens} candidate models to form one ensemble model and repeat this experiment 500 times through sampling with replacement to create 500 ensemble models. Thus, each dot with a fixed color represents one out of these 500 ensemble models and each color indicates the value of M_{ens} or the number of candidate models in the pool. We chose $\epsilon = 0.01$, to compute the threshold δ_a in (15).

churn w.r.t. a deep ensemble of pool size $M_{\text{ens}} = M$, and ECE. In Figure 6, and Figure 7, we use 2-D scatter plots to visualize the relation among the ensemble models w.r.t. these three metrics. We expect the best ensembles to have high validation accuracy, low churn, and low ECE.

As observed in Figure 6 and 7, as we increase M_{ens}, we notice the ensemble models converge to high validation accuracy, low churn, and low ECE. These "good" ensembles are also the same models whose eCDFs have achieved a smaller L_∞-distance from the reference function $\hat{\bar{G}}$ in Figure 5. Some ensembles created from a small number of models may outperform those created from a large number of models in one particular metric, but it is always at the cost of performance w.r.t. another metric. For instance, one of the highest validation accuracies achieved in Figure 6 is an ensemble model belonging to $M_{\text{ens}} = 3$. However, this model also has a higher churn w.r.t. an ensemble of size $M_{\text{ens}} = M$ compared to ensembles models with large M_{ens} ($M_{\text{ens}} = 100, 150, 200$). This is also reflected by the high L_∞-distance of this model from the reference $\hat{\bar{G}}$ as shown in Figure 5. We conclude that contrary to standard practice, exploring more than 5 seeds is important to create a reliable ensemble that has less variability in different performance metrics. Thus, **ensembling over smaller pool sizes results in a deep ensemble that has significant variability over different performance metrics, and increasing the size M_{ens} of ensemble models reduces this variability.**

2) Selecting the number of models to be ensembled via the robust KS-test: Section V-A1 showed that it is possible to create a good ensemble with enough models that have less variability over different metrics and whose eCDF approximates the reference function well. The more models we ensemble, the better and more consistent the ensembles become. This brings us to our final question: **How many candidate models do we need to create a deep ensemble model that is a reliable representative of the training procedure i.e. has less variability over different performance metrics and approximates the reference function well?** Based on the threshold set by our test on the L_∞-distance and as demonstrated in Figure 5, a classical KS-test will indicate that we need to explore 100 random seeds to create a reliable ensemble, which can then be used as a base model for qualitative comparisons with candidate models. Since the classical KS-test is too sensitive, setting a threshold based on it will require too many models to create the ensemble. We can use the proposed robust KS-test to also measure the similarity between the ensemble function $\hat{\bar{H}}'$ and the reference function $\hat{\bar{G}}$ but with a little wiggle room. In practice, we also may not want to rely on validation accuracy or churn to decide on the number of models we need. So, the proposed test offers a way to estimate this number using only the logits. To that end, we compute $\hat{\alpha}$ through a robust KS-test by measuring the closeness/discrepancy of the eCDFs of ensemble models $\hat{\bar{H}}'$, of different sizes, with the reference function $\hat{\bar{G}}$. Similar to Figure 5, in Figure 8, as M_{ens} approaches M', $\hat{\alpha}$ approaches 0. However, if we set a threshold on the α-trimming level instead of the L_∞-distance, we get a better lower bound on the number of models we need to use for a reliable ensemble. If we consider all ensembles that require $\hat{\alpha} \leq 0.05$, we see that ensembles created from $M_{\text{ens}} = 30$ models or more are all included in this set as shown in column 2 of Table I, while more than 20% of ensembles created from $M_{\text{ens}} = 3$ models have required a higher level of trimming. From Table I, we notice that ensembles achieve better values in all metrics as the size of the ensemble M_{ens} increases and the variability of these three metrics reduces noticeably for ensembles created from $M_{\text{ens}} = 30$ models or more. **Thus, our proposed metric indicates that ensembling over at least $M_{\text{ens}} = 30$ candidate models may be required for an ensemble to be a reliable representative of the training variability.**

TABLE I
ENSEMBLE STATISTICS FOR DIFFERENT VALUES OF M_{ens}

M_{ens}	% of models with $\hat{\alpha} \leq 0.05$	Accuracy mean $\pm$ std	Churn w.r.t. ensemble mean $\pm$ std	ECE mean $\pm$ std
3	77.20	91.85 ± 0.25	190.79 ± 33.90	0.0226 ± 0.0048
5	89.20	92.03 ± 0.18	145.70 ± 28.61	0.0200 ± 0.0037
10	96.80	92.11 ± 0.14	103.57 ± 22.50	0.0185 ± 0.0023
20	99.40	92.17 ± 0.11	76.972 ± 17.23	0.0175 ± 0.0016
30	100.00	92.21 ± 0.10	62.84 ± 12.45	0.0171 ± 0.0014
70	100.00	92.22 ± 0.08	43.45 ± 9.64	0.0168 ± 0.0012
100	100.00	92.23 ± 0.08	38.59 ± 9.08	0.0169 ± 0.0012
150	100.00	92.23 ± 0.07	32.75 ± 7.42	0.0169 ± 0.0011
200	100.00	92.24 ± 0.07	29.3 ± 6.91	0.0169 ± 0.0011

B. Evaluating the proposed metric of model closeness/ discrepancy

We next move on to understand how our proposed measure of model closeness/discrepancy relates to metrics commonly used to understand model variability. We try to understand this for candidate models generated under different sources of randomness (like initialization and batch order). In particular, we considered three scenarios: S_{init} with only random initialization and fixed batches, $\mathsf{S}_{\text{batch}}$ with only random batch selection in SGD and fixed initialization, and S_{all} combining both sources of randomness. We trained 200 models in each of S_{init}, $\mathsf{S}_{\text{batch}}$, and S_{all} with the same hyperparameter settings

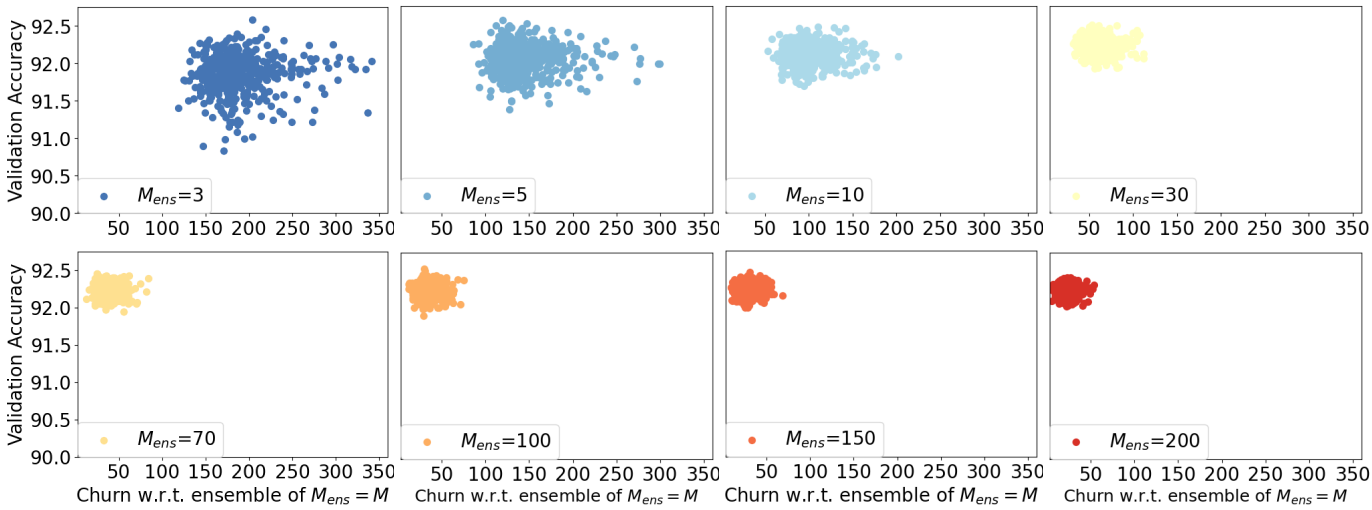

Fig. 6. 2-D scatter plot to visualize the relationship among ensemble models in terms of validation accuracy, and churn w.r.t. an ensemble of size $M_{\text{ens}} = M$. The ensemble models are the same as Figure 5.

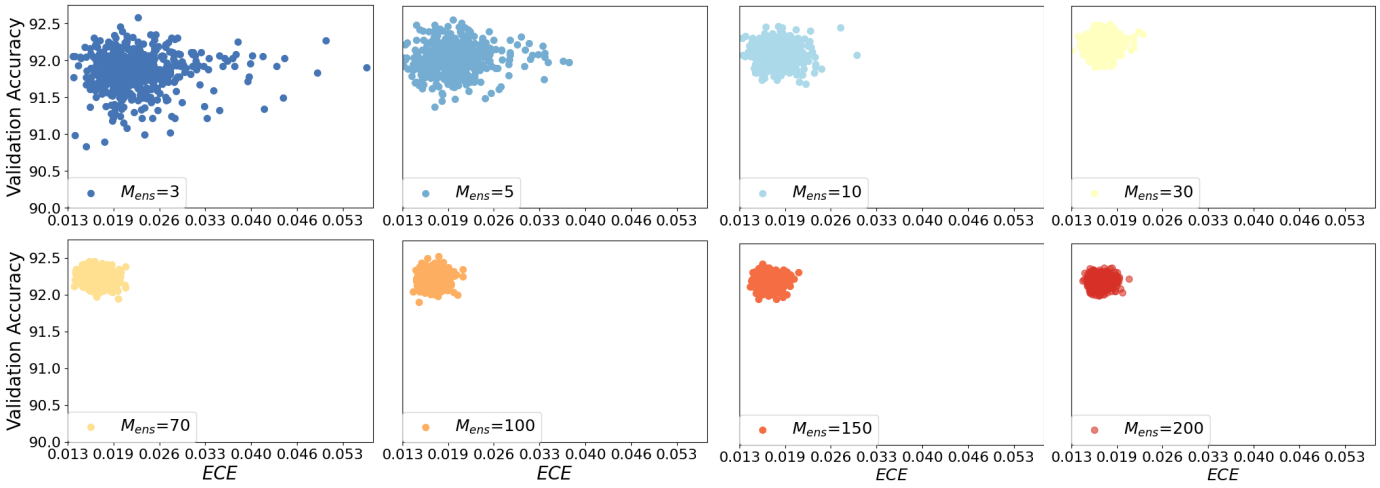

Fig. 7. 2-D scatter plot to visualize the relationship among ensemble models in terms of validation accuracy, and ECE. The ensemble models are the same as Figure 5.

as detailed at the beginning of Section V. For each source of randomness, we use $M = 100$ models to create the reference function $\hat{\bar{G}}$, as defined in (8) and from the rest $M' = 100$ models, we choose our candidate models. For each candidate model, we use its eCDF $\hat{G}_l$, $l \in [M']$, as defined in (7), to compute $\hat{\alpha}$ through a robust KS-test against the reference $\hat{\bar{G}}$.

1) Comparing our proposed discrepancy measure with accuracy: Our first question in this section is: **How does our proposed metric $\hat{\alpha}$ relate to accuracy?** In Figure 9, we plot the relationship between the validation accuracy of candidate models and $\hat{\alpha}$ for candidate models in S_{init}, $\mathsf{S}_{\text{batch}}$, and S_{all}. For each plot, we notice significant variability over random seeds in both $\hat{\alpha}$ and validation accuracy for the same hyperparameter setting. This is evidence that a fixed hyperparameter setting that works well for one random seed can perform poorly for another and thus produce models with very different logit gap functions. We notice more variability in both validation accuracy and $\hat{\alpha}$ among models in $\mathsf{S}_{\text{batch}}$, than models in S_{init}. Details of accuracy statistics for each source of randomness are listed in Table II. The mean accuracy of models in S_{init} is slightly higher than models in S_{all}, and $\mathsf{S}_{\text{batch}}$, while all three categories have achieved similar maximum accuracy as observed in the accuracy range column. The higher variability among models in $\mathsf{S}_{\text{batch}}$ comes from producing models with much poorer validation accuracy than S_{init}, a trend also observed in S_{all}, as reported in the minimum accuracy of the accuracy range for each category. This indicates that in this case study, random batch shuffling has a higher effect on the overall variability

TABLE II
VALIDATION ACCURACY FROM DIFFERENT SOURCES OF RANDOMNESS

Source of randomness	Accuracy mean $\pm$ std	Accuracy range	Ensemble accuracy	Accuracy range of models with $\hat{\alpha} \leq 0.05$
S_{init}	91.101 ± 0.302	[90.422, 91.781]	91.731	[90.422, 91.781]
$\mathsf{S}_{\text{batch}}$	90.819 ± 0.554	[87.852, 91.843]	91.731	[90.409, 91.843]
S_{all}	90.810 ± 0.560	[88.513, 91.769]	92.361	[90.409, 91.769]

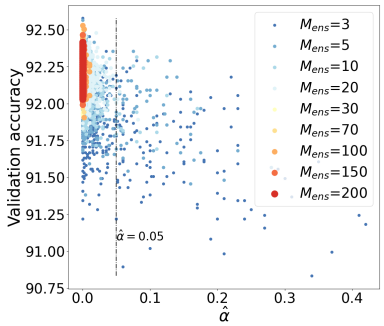

Fig. 8. A plot showing $\hat{\alpha}$ of the eCDF of ensemble models $\hat{\hat{H}}'$ (formed with M_{ens} models) w.r.t. the reference $\hat{\hat{G}}$ (formed with M models) against validation accuracy of ensemble models. We randomly select M_{ens} candidate models for each value of M_{ens} to form an ensemble. We repeat this procedure 500 times, sampling with replacement, to create 500 ensemble models. Thus, each dot with a fixed color represents one out of these 500 ensemble models and each color indicates the value of M_{ens} or the number of candidate models in the pool. The ensemble models are the same as Figure 5.

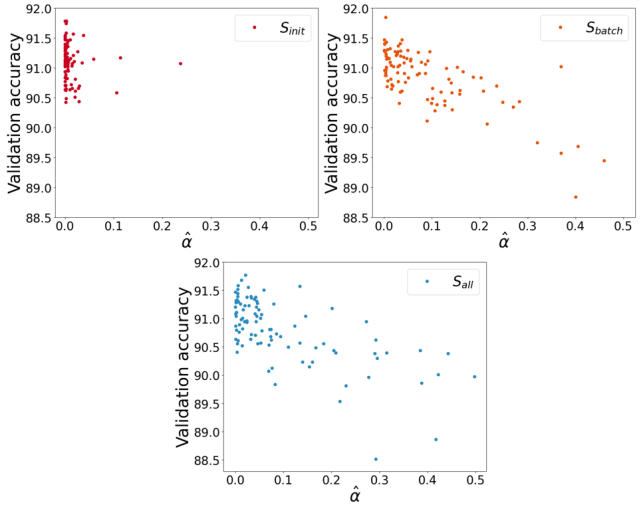

Fig. 9. 2-D Scatter plot to visualize how $\hat{\alpha}$, computed using the robust KS-test against the reference $\hat{\hat{G}}$, relates to validation accuracy for 100 candidate models (each dot in the plots) in S_{init}, $\mathsf{S}_{\text{batch}}$ and S_{all}.

among models in S_{all}, than random initialization. We also notice that although there is more variability among models in $\mathsf{S}_{\text{batch}}$, than S_{init}, the ensemble model has performed the same for both categories. However, the combined variability of S_{init}, and $\mathsf{S}_{\text{batch}}$, reflected among models in S_{all} has led to a better ensemble performance in S_{all}.

We also look at models admitted by small values of $\hat{\alpha}$ in each category (e.g. $\hat{\alpha} \leq 0.05$), and observe the corresponding range of validation accuracy, reported in the fourth column of Table II. For S_{init}, the range of accuracy achieved by models admitted by $\hat{\alpha} \leq 0.05$ covers the full range of accuracy of all models in S_{init}. This range is also close to the range of models admitted by $\hat{\alpha} \leq 0.05$ in S_{all}, and $\mathsf{S}_{\text{batch}}$. The maximum validation accuracy for each source is also admitted by small values of $\hat{\alpha}$. This is because candidate models aggregate to form ensembles that result in a boost over average performance in the group. Since models with lower $\hat{\alpha}$ are close to the reference function in terms of their eCDF, and hence also to the ensemble, they will also have accuracy similar to the ensemble. For each source of randomness, if we consider the range of validation accuracy for models admitted by small values of $\hat{\alpha}$ to be representative of the training variability (as a result of being close to the reference function), then any model that has performed worse than this range will end up with a high value of $\hat{\alpha}$. This can be observed for $\mathsf{S}_{\text{batch}}$, and S_{all} since these two categories have resulted in more models with performance worse than the discussed range. However, the opposite is not true, i.e. models with accuracy within the discussed range will not always have a small $\hat{\alpha}$, and hence will not be good representatives. This shows that validation accuracy alone is not the right metric to assess model quality. A model can end up within the discussed range of validation accuracy but not be a good representative of the training variability over random seeds. We conclude that **smaller values of $\hat{\alpha}$ does not admit poor validation accuracy, but similar validation accuracy does not imply similar $\hat{\alpha}$.**

TABLE III
PAIRWISE CHURN FOR DIFFERENT TYPES OF RANDOMNESS

Source of randomness	Range of pairwise churn	Range of pairwise churn of pairs of models with $(\hat{\alpha_i} \leq 0.05, \hat{\alpha_j} \leq 0.05), i \neq j$
S_{init}	[270, 481]	[270, 481]
$\mathsf{S}_{\text{batch}}$	[218, 1174]	[218, 538]
S_{all}	[327, 1184]	[327, 591]

2) Comparing our proposed discrepancy measure with pairwise churn: **How does our proposed metric $\hat{\alpha}$ relate to pairwise churn?** In Figure 10, we plot the relationship between the pairwise disagreement between candidate models or churn, and $\hat{\alpha}$, for candidate models in S_{init}, $\mathsf{S}_{\text{batch}}$, and S_{all}. We distinguish this from churn w.r.t. a deep ensemble by labeling it as pairwise churn. Similar to the trend observed for validation accuracy, there is more variability in pairwise churn among pairs of models in $\mathsf{S}_{\text{batch}}$, than S_{init}, as observed in the pairwise churn range for each random source reported in column 2 of Table III. Again, most of the variability in pairwise churn in S_{all} is due to $\mathsf{S}_{\text{batch}}$. In Figure 10, for $\mathsf{S}_{\text{batch}}$, and S_{all}, we observe that as the range of pairwise churn increases, the scatter plot becomes less dense in the region where both the models have a low $\hat{\alpha}$. This indicates that pairs of models with low values of $\hat{\alpha}$ for both models achieve pairwise churn in the smaller range among its group. This is less obvious for models in S_{init}, because this group's total range of pairwise churn is low. We see this in column 3 of Table III, where we focus on pairs of models in each category that have achieved pairwise $\hat{\alpha}$ less than 0.05 and report the range of pairwise churn for those models. Since the range of pairwise churn of pairs of models in S_{init} is low, small values of pairwise $\hat{\alpha}$ have admitted this full range. However, for $\mathsf{S}_{\text{batch}}$, and S_{all}, since the total range of pairwise churn is much larger than that of S_{init}, small values of $\hat{\alpha}$ have admitted only those pairs of models that have achieved a pairwise churn in the smaller range of the total range. Any pairs of models with pairwise churn higher than this range will have at least one model in the pair with a high $\hat{\alpha}$. A possible explanation for a pair of models with low pairwise $\hat{\alpha}$ also having a low pairwise churn is that models with low $\hat{\alpha}$

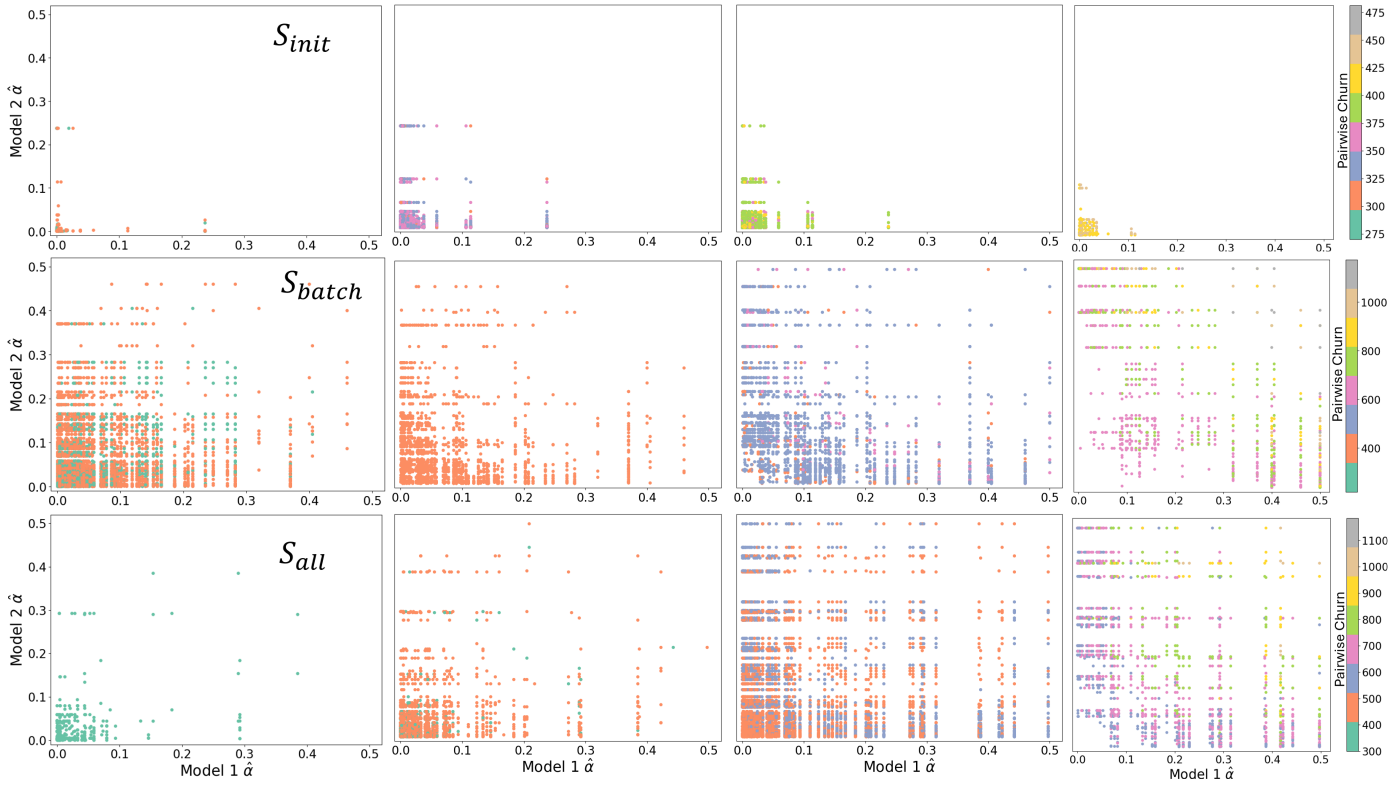

Fig. 10. 2-D Scatter plot to visualize how pairwise values of $\hat{\alpha}$, computed using the robust KS-test against the reference $\hat{\bar{G}}$, relates to pairwise churn for each pair of (100×100) candidate models (each dot in the plots). For each source of randomness (plotted along the rows), the total pairwise churn is divided into 4 ranges (plotted along the columns) indicated by the colorbar. The banded structure of the plot is because our hypothesis test checks for only discrete values of α.

are expected to consist of "good" models that are similar to a good quality ensemble. Since these models achieve validation accuracy within a higher range among their group, the number of points they will make mistakes on will be low. This will also result in a low pairwise churn because these models will only have a limited number of test points to disagree on. As observed for validation accuracy, the opposite is again not true for pairwise churn, i.e., models with low pairwise churn will not necessarily have small values of pairwise $\hat{\alpha}$. To summarize, if both the candidate models in a pair are close to the reference function then their pairwise churn will be low. However, if one candidate model in a pair is close to the reference function, while the other is far away from it, then the pairwise churn between these models can be either high or low. We conclude that **smaller values of pairwise $\hat{\alpha}$ does not admit high pairwise churn but low pairwise churn does not imply smaller pairwise $\hat{\alpha}$.**

3) Our proposed discrepancy measure is more informative than accuracy: We now provide evidence that $\hat{\alpha}$ is more informative than validation accuracy for model comparison. Figure 11 and Figure 12 show the relationship of $\hat{\alpha}$ of candidate models in S_{all} with three metrics: validation accuracy, churn w.r.t. the ensemble of candidate models, and their ECE, in 2-D and 3-D respectively. If the eCDF of a candidate model is close to the eCDF of the reference function, i.e. for smaller values of $\hat{\alpha}$, the variability in each of the three metrics is confined to a range that includes the best value in each metric, with the worst value not differing too much from the best value. For

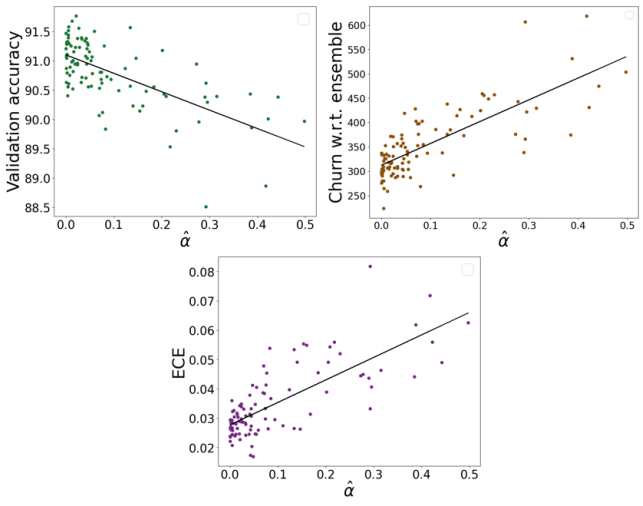

Fig. 11. 2-D scatter plot to visualize the relationship of $\hat{\alpha}$ of candidate models in S_{all} with their average cdf, $\hat{\bar{G}}$, in terms of (Left) Validation accuracy, (Middle) Churn w.r.t. their ensemble and (Right) ECE.

each metric, we can think of the range of values corresponding to a small $\hat{\alpha}$ as the minimum variability observed in models that are close to their expected distribution. Any model that has achieved a value worse than this range will also have a high $\hat{\alpha}$. Models with low $\hat{\alpha}$ are within a higher accuracy range among their group and have ECE and churn w.r.t. ensemble within a low range. However, the converse is not true, i.e.

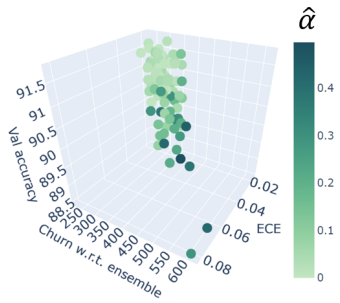

Fig. 12. 3-D scatter plot to visualize how $\hat{\alpha}$ compares to other metrics for 100 candidate CNN models performing a binary classification task on CIFAR-10.

TABLE IV
TABLE SHOWING $\hat{\alpha}$ VALUES FOR CNN MODELS WITH SIMILAR VALIDATION ACCURACY AND THEIR CORRESPONDING VALUES ACHIEVED IN OTHER METRICS.

	ECE	Churn w.r.t. ensemble	Validation accuracy	Average churn	$\hat{\alpha}$
Model 1	0.028	297	91.108	472.15	0.002
Model 2	0.021	280	91.519	458.61	0.005
Model 3	0.026	265	91.581	459.37	0.005
Model 4	0.030	309	91.157	475.91	0.016
Model 5	0.023	318	91.769	476.95	0.021
Model 6	0.030	371	91.382	482.45	0.024
Model 7	0.029	307	91.556	476.35	0.027
Model 8	0.026	376	91.569	**513.83**	**0.134**
Model 9	**0.038**	**425**	91.182	**552**	**0.201**

high accuracy or low churn w.r.t. ensemble or low ECE alone does not imply low $\hat{\alpha}$. These models can achieve a value that is within a good range of values in one metric but can perform poorly in another metric, and this is reflected in the corresponding $\hat{\alpha}$ value. To see this, we focus on models that have achieved a similar high validation accuracy and look at other metrics like ECE, average churn for each model w.r.t. all other models, and churn w.r.t. their ensemble in Table IV. $\hat{\alpha}$ is low if all three metrics fall within the previously discussed good range of values. An increase in $\hat{\alpha}$ (denoted in bold) in Table IV indicates a reduction in quality in one or more of these metrics. We can see that most models with similar validation accuracy in Table IV have needed only a small trimming level ($\hat{\alpha} \leq 0.05$) to not reject the null. But for model 8 and model 9, we see that despite having similar validation accuracy to other models in the group, $\hat{\alpha}$ values are large due to a reduction in quality in other metrics (denoted in bold).

C. Application in Transfer Learning

We conclude our experiments by demonstrating the usefulness of our framework in transfer learning. A common transfer learning setup is when a model trained on one task (usually on a large dataset) is fine-tuned to perform a different, often related task, using a smaller dataset. The smaller dataset (target domain) is assumed to be from the same distribution as the larger dataset (source domain). The pre-trained models in transfer learning are typically very large, designed to capture a wide range of useful features from a vast amount of data, allowing them to generalize well to a variety of tasks.

We use a Vision Transformer(ViT) variant [30], pre-trained on ImageNet, to perform a downstream binary classification

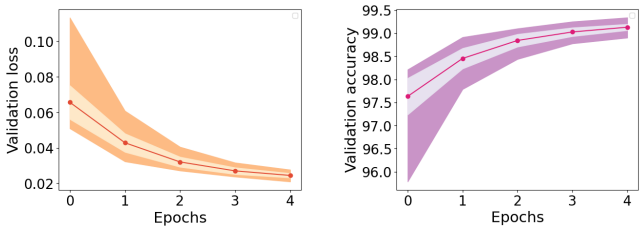

Fig. 13. (Left) A plot showing the evolution of validation loss of pre-trained ViT models, over different epochs. The solid red dots represent the mean of validation loss over 45 seeds at each epoch, the light-colored area represents one standard deviation, and the orange area represents the maximum and minimum values at that epoch. (Right) A plot showing the evolution of validation accuracy of pre-trained ViT models, over different epochs. The solid red dots represent the mean of validation accuracy over 45 seeds at each epoch, the light-colored area represents one standard deviation, and the purple area represents the maximum and minimum values at that epoch.

task on CIFAR-10 (as described in Section V). The pre-trained model is available as part of the Hugging Face Community [31]. We only train the final task-specific classification layer for 5 epochs (till close to convergence as visualized in Figure 13) and rely on already fine-tuned hyperparameters and the pre-trained weights of the upstream task to "fine-tune" the downstream binary classification task over random seeds only. Thus, keeping the fine-tuning regime and pre-trained weights fixed, we only vary the random seed that controls different sources of randomness in the training procedure (initialization and batch order during SGD), generate 90 models, and observe the variability over random seeds for this fixed setup. As observed by Picard et al. [6], there is less variability in validation accuracy over random seeds when using pre-trained models.

We create our reference function $\hat{\bar{G}}$ using the first 45 models and treat the remaining 45 models as our candidate models $\hat{G}_l$, $l \in M'$, for comparison with the reference. We conduct the same experiment as in Section V-B3, where we run our robust hypothesis test between the reference function $\hat{\bar{G}}$ and candidate models $\hat{G}_l$ to compute our proposed discrepancy measure $\hat{\alpha}$. Following similar reasoning to Section V-B3, in Figure 14, we observe how our proposed metric is more informative than validation accuracy. The measure $\hat{\alpha}$ is indicative of the quality of models in terms of other metrics like ECE, churn w.r.t. an ensemble, and average churn of candidate models w.r.t. all other models in the pool alongside accuracy. A higher $\hat{\alpha}$ is usually accompanied by a reduction in quality in one or more of these three metrics, denoted in bold in Table V. Although pre-trained large-scale models provide us with very high validation accuracy and less variability in different metrics, our experiment demonstrates how smaller values of $\hat{\alpha}$ are still helpful in identifying the best representatives of the training procedure.

For the next set of experiements, we use the first $M = 45$ models to create a reference function $\hat{\bar{G}}$ and set the rest $M' = 45$ models to create the eCDF of a deep ensemble $\hat{\bar{H}}'$. We choose different ensemble sizes, $M_{\text{ens}} \in [3, 5, 10, 30]$, to observe the variability of ensembles in different metrics. Figure 15, and Figure 16 show the relationship among different metrics for ensembles of the pre-trained candidate models.

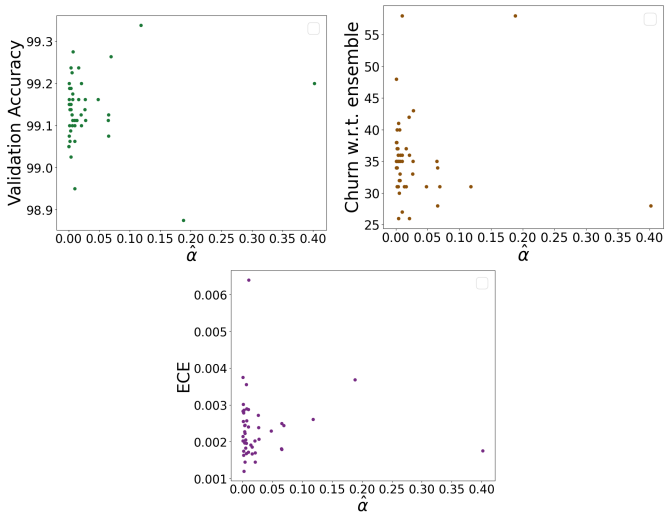

Fig. 14. 2-D Scatter plot to visualize how $\hat{\alpha}$, computed using the robust KS-test against the reference $\hat{\tilde{G}}$, relates to validation accuracy for 45 candidate ViT models (each dot in the plots) differing in random seeds.

TABLE V
TABLE SHOWING $\hat{\alpha}$ VALUES FOR PRE-TRAINED VIT MODELS WITH SIMILAR VALIDATION ACCURACY AND THEIR CORRESPONDING VALUES ACHIEVED IN OTHER METRICS.

	Average churn	**Churn w.r.t. ensemble**	**Validation accuracy**	**ECE**	$\hat{\alpha}$
Model 1	44.82	26	99.2	0.002	0.004
Model 2	45.88	24	99.2	0.001	0.007
Model 3	48.33	30	99.3	0.002	0.026
Model 4	47.97	**34**	99.2	0.001	**0.069**
Model 5	**49.44**	28	99.25	**0.003**	**0.118**

Since all the individually trained models achieved very high validation accuracy and low variability, the ensembles did not result in significant performance gains over their constituents. This low variability also resulted in a lower $\hat{\alpha}$ for most pre-trained candidate models, as seen in Figure 14. If all models performed similarly, ensembling them would not make much difference. Thus, moderate variability in model performance may be required for the pool of candidate models to benefit from ensembling, which implies that $\hat{\alpha}$ for all candidate models cannot be too low.

The low variability observed in pre-trained ViT models in the above setting does not generalize to all types of pre-trained models. Dodge et al. demonstrated that finetuning pre-trained language models, like BERT, over weight initialization, and data ordering, results in significant improvement in performance metrics, indicating the instability of the training process in large language models [11]. They provide evidence that some data orders and initializations are better than others and emphasize the need for more rigorous reporting of benchmark model performance for comparisons across different architectures. However, like most previous works, they also use accuracy as their metric to analyze fine-tuning variability. Our metric $\hat{\alpha}$ can add an extra layer of reliability by identifying "good" seeds that closely represent the expected training behavior.

Through these experiments, we highlight the importance of

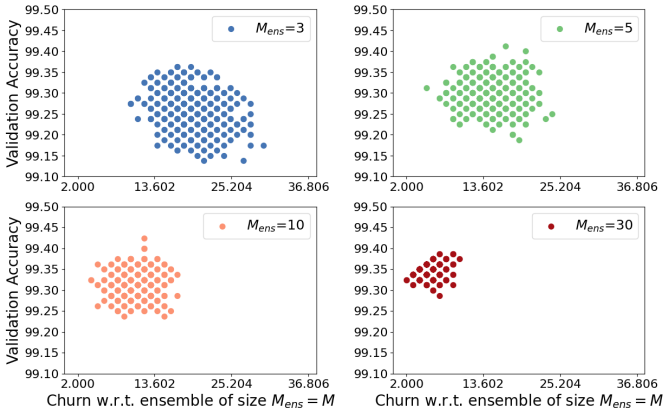

Fig. 15. 2-D scatter plot to visualize the relationship among ViT ensemble models in terms of validation accuracy, and churn w.r.t. an ensemble of size $M_{\text{ens}} = M$. For each value of M_{ens}, we choose M_{ens} candidate models to form one ensemble model and repeat this experiment 500 times through sampling with replacement to create 500 ensemble models. Thus, each dot with a fixed color represents one out of these 500 ensemble models and each color indicates the value of M_{ens} or the number of candidate models in the pool. Several ensemble models have achieved the same accuracy or churn resulting in fewer than 500 visible models.

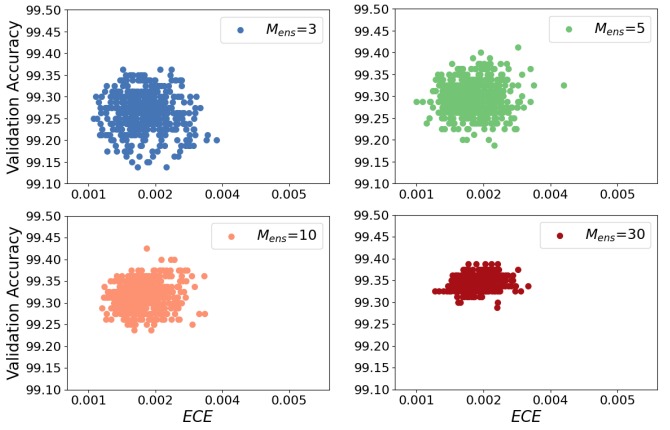

Fig. 16. 2-D scatter plot to visualize the relationship among ViT ensemble models in terms of validation accuracy, and ECE. We follow the experiment in Figure 15 to create the ensemble models.

treating the seeds that control random elements in a DNN as a separate hyperparameter that needs "tuning" and propose a framework to reliably select seeds that do not rely solely on commonly used performance metrics like validation accuracy. When good seed selection is needed to account for the variability caused by random seeds, we recommend using a rule of thumb such as exploring at least 30 seeds and choosing the seed that generates a model with high validation accuracy and small $\hat{\alpha}$.

VI. CONCLUSION AND FUTURE WORK

In this work, we propose a framework for random seed selection of a DNN with a fixed architecture and a fixed hyperparameter setting. Our proposed framework is based on a robust two-sample hypothesis testing problem that uses trimming of the eCDF of samples obtained from the logit gap function. Our test assesses the closeness of candidate models

in a pool with their expected eCDF by down-weighting that part of the data that has a greater influence on the dissimilarity. We also provide some evidence that our new measure, the trimming level $\hat{\alpha}$, could be a more informative metric to assess model performance than commonly used test/validation accuracy. This allows us to perform random seed selection in a more principled fashion instead of relying solely on trial-and-error methods or metrics like validation accuracy. Although our paper focused on the influence of the random seed on model variability, our method can be extended to investigate the influence of any hyperparameter.

Since the usefulness of our methodology relies on ensembles achieving significant performance gains over their constituents, we investigate the behavior of the ensemble as the number of models in the ensemble pool increases. Our proposed metric $\hat{\alpha}$ can also be used as a stopping criterion for the number of different seeds that need to be explored for an ensemble model to reliably approximate the expected eCDF of the logit gap distribution of candidate models, and to have less variability in different performance metrics. Although ensembling by model averaging in the output space is quite straightforward and usually results in performance gains, this comes at the cost of reduced interpretability, since such ensembles do not learn any parameters or features. By selecting models that are close to their ensembles in the output space of a DNN, instead of the ensemble itself, our framework leaves room for model interpretability while at the same time maintaining high accuracy compared to their counterparts. Our methodology is useful in critical application areas like credit risk assessment where a single model is often preferred over the improved accuracy of an ensemble due to interpretability concerns [32].

There is a common belief in the ensemble learning literature that increasing diversity among ensemble members will improve the quality of the ensembles [33] [34]. If all models were to perform exactly the same then the ensemble of these models are not going to perform any better than a single model in the ensemble. As observed from the experimental results in the transfer learning application, low diversity in models would indicate lower values of $\hat{\alpha}$ for all candidate models. However, imposing too much diversity among ensemble members can also be detrimental to the ensembles performance as increasing diversity can sacrifice overall model performance. In this scenario, we would likely find many models with large $\hat{\alpha}$. Through our test, $\hat{\alpha}$ will select models that are closer to a poor performing ensemble, which may lead to selecting random seeds that achieve poor accuracy. Thus, in scenarios where ensembling by model averaging might not lead to significant performance gains over candidate models in the pool or hurt performance, future extensions of this work include investigating when candidate models form effective ensembles.

We are also interested in extensions to multi-class classification and exploring robust two-sample hypothesis testing based on other distance metrics like the Wasserstein metric [35]. In this work, we focused on samples from the logit gap function as our probe to understand deep net variability. We can look at other functions of the trained models, such as the Jacobian or the Neural Tangent Kernel of functions learned by these models [36]. The eigen-distribution of these matrices for instance can inform us how well individual candidate models can generalize to test data.

APPENDIX A
PROOF OF THEOREM 1

Proof: Both $\bar{F}_{\pi|\mathcal{D}_{\text{param}}}$ and $\hat{\bar{G}}$ take averages over M parameters. Define

$$A_k(t) \doteq \mathbb{E}_{\pi|\theta_k}[\mathbb{1}(m(\mathbf{x},\theta_k) \leq t)], \text{ and} \tag{40}$$

$$B_k(t) \doteq \frac{1}{N}\sum_{j=1}^{N} \mathbb{1}(m(x_j;\theta_k) \leq t) \tag{41}$$

Then we can upper bound $\left\|\bar{F}_{\pi|\mathcal{D}_{\text{param}}} - \hat{\bar{G}}\right\|_\infty$ by

$$\left\|\bar{F}_{\pi|\mathcal{D}_{\text{param}}} - \hat{\bar{G}}\right\|_\infty \leq \frac{1}{M}\sum_{k=1}^{M} \|A_k - B_k\|_\infty. \tag{42}$$

For each θ_k, B_k is the eCDF of samples drawn from a distribution with CDF A_k. By the DKW inequality [13],

$$\mathbb{P}_{\pi|\theta_k}(\|A_k - B_k\|_\infty > \delta_b) \leq 2\exp(-2N\delta_b^2). \tag{43}$$

Taking a union bound over $\mathcal{D}_{\text{param}} = \{\theta_k : k \in [M]\}$ we get,

$$\begin{aligned}\mathbb{P}_{\pi|\mathcal{D}_{\text{param}}}(\exists k \in [M] \text{ s.t. } \|A_k - B_k\|_\infty > \delta_b) \\ \leq 2M\exp(-2N\delta_b^2).\end{aligned} \tag{44}$$

Therefore, each term on the right-hand side of (42) is smaller than δ_b w.h.p. and as desired,

$$\begin{aligned}\mathbb{P}_{\pi|\mathcal{D}_{\text{param}}}\left(\left\|\bar{F}_{\pi|\mathcal{D}_{\text{param}}} - \hat{\bar{G}}\right\|_\infty \leq \delta_b\right) \\ \geq 1 - 2M\exp(-2N\delta_b^2).\end{aligned} \tag{45}$$

■

APPENDIX B
MORE DETAILS ON TRIMMING

A. Impartial trimming

In Section III-B, we introduce a robust version of the KS-test, where we allow outliers in samples from the null hypothesis using α-trimming of distributions. For completeness, we present the concepts on α-trimming of a distribution w.r.t. a reference distribution as introduced by Alvarez-Esteban et al. [18]. Trimming allows us to assume some outliers in samples that did come from the null distribution and helps quantify the fraction of these outliers.

Let k be the number of trimmed observations, and let α be the trimming fraction, which implies $k \leq n\alpha$. Given n i.i.d samples $\{x_i\}_{i=1}^n$ with probability distribution P_1, the empirical measure can be defined as $\frac{1}{n}\sum_{i=1}^n \delta^{\text{Dirac}}(x_i)$, where $\delta^{\text{Dirac}}(x)$ is the Dirac measure. To not reject the null, we can remove outliers by assigning a weight of 0 to the bad observations in the sample and adjusting the weight of good observations by $\frac{1}{n-k}$. However, we may not want to completely get rid of samples from the feasible set and instead downplay the importance of bad observations by modifying the empirical

measure to be $\frac{1}{n}\sum_{i=1}^{n} w_i\delta_{x_i}$, where $0 \leq w_i \leq \frac{1}{(1-\alpha)}$, and $\frac{1}{n}\sum_{i=1}^{n} w_i = 1$. This is called an *impartial trimming* of the probability measure P_1. For the remainder, we use the term impartial trimming interchangeably with trimming.

An α-trimming of the distribution P_1, denoted by $R_\alpha(P_1)$, is defined as [18, Definition 1],

$$\mathcal{R}_\alpha(P_1) = \{P \in \mathcal{P} : P \ll P_1, \frac{dP}{dP_1} \leq \frac{1}{(1-\alpha)} P_1 \text{ a.s.}\} \quad (46)$$

where P_1, P are probability measures on $\mathbb{R}$, $0 \leq \alpha \leq 1$, and we say P is an α-trimming of P_1 if P is absolutely continuous w.r.t. P_1 and satisfies the above definition. The set of $R_\alpha(P_1)$, the α-trimmings of P_1, can be characterized in terms of the trimming function h. The function h determines which zones in the distribution P are downplayed or removed.

Let $\mathcal{C}_\alpha$ be the class of absolutely continuous functions $h : [0,1] \to [0,1]$, such that $h(0) = 0$, and $h(1) = 1$, with derivative h', such that $0 \leq h' \leq \frac{1}{1-\alpha}$. For any real probability measure P_1, the following holds [18, Proposition 1 a.],

$$\mathcal{R}_\alpha(P_1) = \{P \in \mathcal{P} : P(-\infty, t] = h(P_1(-\infty, t]), h \in C_\alpha\}. \quad (47)$$

B. Trimmed Kolmogorov-Smirnov distance

Given two distribution functions, $F_{\tau\times\pi}$ and G_0, as defined in Section II-B, the KS distance is given by,

$$d(F_{\tau\times\pi}, G_0) = \sup_{x\in\mathbb{R}} |F_{\tau\times\pi}(x) - G_0(x)|. \quad (48)$$

We similarly define the α-trimmed KS distance functional as,

$$d(F_{\tau\times\pi}, R_\alpha(G_0)) = \min_{\tilde{F}\in\mathcal{R}_\alpha(G_0)} d(F_{\tau\times\pi}, \tilde{F}). \quad (49)$$

The plug-in estimator for $d(F_{\tau\times\pi}, R_\alpha(G_0))$ is $d(F_{\tau\times\pi}, R_\alpha(\hat{G}_0))$, where $\hat{G}_0$, defined in Section II-C, is the empirical distribution function based on a sample of N independent random variables with common distribution function G_0, and $F_{\tau\times\pi}$ is our null distribution. A practical computation for $d(F_{\tau\times\pi}, \mathcal{R}_\alpha(\hat{G}_0))$ uses function $F_{\tau\times\pi} \circ G_0^{-1}$ to express $d(F_{\tau\times\pi}, R_\alpha(G_0))$. If $F_{\tau\times\pi}$ is continuous and x (generated samples) has distribution function G_0, $F_{\tau\times\pi} \circ G_0^{-1}$ is the quantile function associated with the random variable $Y = F_{\tau\times\pi}(x)$. This gives rise to the following lemma and theorem for computing the trimmed KS distance between a theoretical distribution and an eCDF [20],

Lemma 1 ([37], Lemma 2.4): If $G_0, F_{\tau\times\pi}$ are continuous distribution functions and G_0 is strictly increasing then,

$$d(F_{\tau\times\pi}, \mathcal{R}_\alpha(G_0)) = \min_{h\in C_\alpha} ||h - F_{\tau\times\pi} \circ G_0^{-1}||, \quad (50)$$

$$d(F_{\tau\times\pi}, \mathcal{R}_\alpha(\hat{G}_0)) = \min_{h\in C_\alpha} ||h - F_{\tau\times\pi} \circ \hat{G}_0^{-1}||. \quad (51)$$

Theorem 2 ([37], Theorem 2.5): Suppose $\Gamma : [0,1] \to [0,1]$ is a continuous non-decreasing function and let

$$B(t) = \Gamma(t) - \frac{t}{1-\alpha}, \quad (52)$$

$$U(t) = \sup_{t\leq s\leq 1} B(s), \quad (53)$$

$$L(t) = \inf_{0\leq s\leq t} B(s), \text{ and} \quad (54)$$

$$\tilde{h}_\alpha(t) = \max\left(\left(\min\left(\frac{U(t)+L(t)}{2}\right), 0\right)\frac{-\alpha}{(1-\alpha)}\right). \quad (55)$$

Then

$$h_\alpha \triangleq \tilde{h}_\alpha + \frac{(\cdot)}{(1-\alpha)} \quad (56)$$

is an element of C_α and $\min_{h\in C_\alpha}||h-\Gamma|| = ||h_\alpha - \Gamma|| = ||\tilde{h}_\alpha - \Gamma||$, with assumptions on Γ holding for $\Gamma = F_{\tau\times\pi} \circ G_0^{-1}$.

In our application we don't have access to $F_{\tau\times\pi}$, and as discussed in Section III-C, our robust hypothesis test is against the reference function $\hat{\bar{G}}$. Since this function is an average of eCDFs and not a continuous function, for practical computation of $d(\hat{\bar{G}}, \mathcal{R}_\alpha(\hat{G}_0))$, we consider the linearly interpolated CDF of $\hat{\bar{G}}$, which we denote as $\hat{G}^l$. Thus, running the test in (34) is equivalent to running the following test,

$$\max_z \min_{\tilde{F}\in\mathcal{R}_\alpha(\hat{G}_0)} |\hat{G}^l(z) - \tilde{F}(z)| \underset{\tilde{\mathcal{H}}_0}{\overset{\tilde{\mathcal{H}}_1}{\gtrless}} \delta_a + \frac{1}{N}, \quad (57)$$

where we arrive at the threshold on the right-hand side by considering the inequality $\left\|\hat{\bar{G}} - \hat{G}^l\right\|_\infty \leq \frac{1}{N}$ and then applying the triangle inequality.

APPENDIX C
ADDITIONAL EXPERIMENTAL DETAILS

Figure 17 shows the CNN architecture and parameters used in Section V.

CNN Summary

path	module	inputs	outputs	params
	CNN	float32[39873,32,32,3]	- float32[39873,2] - float32[39873,64] - float32[39873,64] - float32[39873,16,16,16] - float32[39873,16,16,32] - float32[39873,32,32,32]	
CONV1	Conv	float32[39873,32,32,3]	float32[39873,32,32,32]	bias: float32[32] kernel: float32[3,3,3,32] **896 (3.6 KB)**
CONV2	Conv	float32[39873,16,16,32]	float32[39873,16,16,16]	bias: float32[16] kernel: float32[3,3,32,16] **4,624 (18.5 KB)**
DENSE1	Dense	float32[39873,1024]	float32[39873,64]	bias: float32[64] kernel: float32[1024,64] **65,600 (262.4 KB)**
DENSE2	Dense	float32[39873,64]	float32[39873,2]	bias: float32[2] kernel: float32[64,2] **130 (520 B)**
			Total	**71,250 (285.0 KB)**

Total Parameters: 71,250 (285.0 KB)

Fig. 17. CNN Architecture

REFERENCES

[1] O. E. Gundersen, K. Coakley, C. Kirkpatrick, and Y. Gil, "Sources of irreproducibility in machine learning: A review," *arXiv preprint arXiv:2204.07610v2*, 2023.

[2] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[3] N. Reimers and I. Gurevych, "Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging," *arXiv preprint arXiv:1707.09861*, 2017.

[4] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," *arXiv preprint arXiv:1707.05589*, 2017.

[5] X. Bouthillier, C. Laurent, and P. Vincent, "Unreproducible research is reproducible," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 2019, pp. 725–734.

[6] D. Picard, "Torch. manual_seed (3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision," *arXiv preprint arXiv:2109.08203*, 2021.

[7] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," *arXiv preprint arXiv:1912.02757*, 2019.

[8] X. Bouthillier, P. Delaunay, M. Bronzi, A. Trofimov, B. Nichyporuk, J. Szeto, N. Mohammadi Sepahvand, E. Raff, K. Madan, V. Voleti, S. Ebrahimi Kahou, V. Michalski, T. Arbel, C. Pal, G. Varoquaux, and P. Vincent, "Accounting for variance in machine learning benchmarks," *Proceedings of Machine Learning and Systems*, vol. 3, 2021.

[9] C. Summers and M. J. Dinneen, "Nondeterminism and instability in neural network optimization," *arXiv preprint arXiv:2103.04514*, 2021.

[10] K. Jordan, "Calibrated chaos: Variance between runs of neural network training is harmless and inevitable," *arXiv preprint arXiv:2304.01910*, 2023.

[11] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *arXiv preprint arXiv:2002.06305*, 2020.

[12] M. Milani Fard, Q. Cormier, K. Canini, and M. Gupta, "Launch and iterate: Reducing prediction churn," in *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[13] A. Dvoretzky, J. Kiefer, and J. Wolfowitz, "Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator," *The Annals of Mathematical Statistics*, pp. 642–669, 1956.

[14] F. Wei and R. M. Dudley, "Two-sample Dvoretzky–Kiefer–Wolfowitz inequalities," *Statistics & Probability Letters*, vol. 82, no. 3, pp. 636–644, 2012.

[15] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference (Sixth Edition)*. Boca Raton, FL, USA: CRC Press, 2021.

[16] P. J. Huber and E. M. Ronchetti, *Robust Statistics, Second Edition*. Hoboken, NJ, USA: John Wiley & Sons, 2009.

[17] A. Gordaliza, "Best approximations to random variables based on trimming procedures," *Journal of Approximation Theory*, vol. 64, no. 2, pp. 162–180, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002190459190072I

[18] P. C. Álvarez-Esteban, E. del Barrio, J. A. Cuesta-Albertos, and C. Matrán, "Trimmed comparison of distributions," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 697–704, 2008.

[19] ——, "Uniqueness and approximate computation of optimal incomplete transportation plans," in *Annales de l'IHP Probabilités et statistiques*, vol. 47, no. 2, 2011, pp. 358–375.

[20] E. del Barrio, H. Inouzhe, and C. Matrán, "On approximate validation of models: a Kolmogorov–Smirnov-based approach," *Test*, vol. 29, no. 4, pp. 938–965, 2020.

[21] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 29, no. 1, 2015.

[22] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 625–632.

[23] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[24] Rutgers Office of Advanced Research Computing. Amarel - Office of Advanced Research Computing. [Online]. Available: https://oarc.rutgers.edu/resources/amarel/

[25] Sinjini Banerjee. Two sample robust Kolmogorov-Smirnov test. [Online]. Available: https://github.com/Sinjini77/RobustKSTest

[26] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 6405–6416.

[27] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, "Why M heads are better than one: Training a diverse ensemble of deep networks," *arXiv preprint arXiv:1511.06314*, 2015.

[28] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[29] D. Wood, T. Mu, A. M. Webb, H. W. Reeve, M. Lujan, and G. Brown, "A unified theory of diversity in ensemble learning," *Journal of Machine Learning Research*, vol. 24, no. 359, pp. 1–49, 2023.

[30] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[31] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[32] R. Florez-Lopez and J. M. Ramon-Jeronimo, "Enhancing accuracy and interpretability of ensemble strategies in credit risk assessment. a correlated-adjusted decision forest proposal," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5737–5753, 2015.

[33] T. Abe, E. K. Buchanan, G. Pleiss, and J. P. Cunningham, "Pathologies of predictive diversity in deep ensembles," *arXiv preprint arXiv:2302.00704*, 2023.

[34] R. Theisen, H. Kim, Y. Yang, L. Hodgkinson, and M. W. Mahoney, "When are ensembles really effective?" *Advances in neural information processing systems*, vol. 36, pp. 15 015–15 026, 2023.

[35] P. C. Álvarez-Esteban, E. del Barrio, J. A. Cuesta-Albertos, and C. Matrán, "Similarity of samples and trimming," *Bernoulli*, vol. 18, no. 2, pp. 606 – 634, 2012.

[36] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[37] E. del Barrio, H. Inouzhe, and C. Matrán, "Box-constrained monotone approximations to lipschitz regularizations, with applications to robust testing," *Journal of Optimization Theory and Applications*, vol. 187, no. 1, pp. 65–87, 2020.