Deep anytime-valid hypothesis testing

Teodora Pandeva
University of Amsterdam

Patrick Forré
University of Amsterdam

Aaditya Ramdas
Carnegie Mellon University

Shubhanshu Shekhar
Carnegie Mellon University

Abstract

We propose a general framework for constructing powerful, sequential hypothesis tests for a large class of nonparametric testing problems. The null hypothesis for these problems is defined in an abstract form using the action of two known operators on the data distribution. This abstraction allows for a unified treatment of several classical tasks, such as two-sample testing, independence testing, and conditional-independence testing, as well as modern problems, such as testing for adversarial robustness of machine learning (ML) models. Our proposed framework has the following advantages over classical batch tests: 1) it continuously monitors online data streams and efficiently aggregates evidence against the null, 2) it provides tight control over the type I error without the need for multiple testing correction, 3) it adapts the sample size requirement to the unknown hardness of the problem. We develop a principled approach of leveraging the representation capability of ML models within the *testing-by-betting* framework, a game-theoretic approach for designing sequential tests. Empirical results on synthetic and real-world datasets demonstrate that tests instantiated using our general framework are competitive against specialized baselines on several tasks.

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 INTRODUCTION

We consider an abstract class of nonparametric hypothesis testing problems characterized by the action of two known operators (denoted by $\mathcal{T}_1$ and $\mathcal{T}_2$) on the data generating distribution. Under the null, it is assumed that the transformed distributions resulting from the action of these two operators are the same, while under the alternative, it is assumed that the two transformed distributions are different. Formally, suppose the observations $Z_1, Z_2, \ldots$ are $\mathcal{Z}$-valued observations drawn i.i.d. from P_Z, and $\mathcal{T}_i : \mathcal{Z} \to \mathcal{W}$ (for some space $\mathcal{W}$ possibly different from $\mathcal{Z}$) for $i = 1, 2$ denote a pair of operators acting on the observation space. Then, we want to test

$$H_0 : \mathcal{T}_1(Z) \overset{d}{=} \mathcal{T}_2(Z) \quad \text{vs.} \quad H_1 : \mathcal{T}_1(Z) \overset{d}{\neq} \mathcal{T}_2(Z). \quad (1)$$

This abstract formulation allows for a unified treatment of several classical and modern problems ranging from two-sample and independence testing, to certifying adversarial robustness and group fairness of machine learning models (details in Section 2).

In this paper, we propose a deep learning-based strategy for designing powerful sequential tests for (1). Compared to classical batch tests, well-designed sequential tests have several benefits: they are valid under optional continuation, can consolidate evidence from possibly dependent experiments, and are computationally cheaper than permutation tests. Our results further enhance these advantages by presenting a principled approach to harness the representational power of deep neural networks (DNNs) in the context of sequential testing. As a result, our tests are particularly well-suited for handling complex data types, such as images and videos.

Contributions. Our main contribution is a unified data-driven strategy for designing powerful se-

quential tests for (1). Since this abstract formulation models a large class of practical applications, our approach effectively yields new sequential tests for all these problems in one shot. This contrasts with some recent works in this area that propose specialized sequential tests for these individual problems.

Our design strategy is guided by the principle of "*testing by betting*" (Shafer, 2021). This principle translates the task of designing sequential tests into that of increasing the wealth of a fictitious bettor in repeated betting games that are fair under the null. Some recent works using this principle (details in Section 3) decouple this task (of setting up and betting on fair games) into separate *betting* and *payoff-design* problems, mainly due to analytical tractability. Our work is motivated by the observation that this decoupling is unnecessary, and we instead develop a class of tests based on joint learning of both the payoff and the bets using deep learning models. In other words, unlike related sequential tests, the deep learning models in our framework are trained to directly optimize the growth rate of the wealth of the bettor, without the separation into betting and payoff design.

Building on this basic idea, we develop a general sequential test for the abstract testing problem in Section 4 and its extension to randomization hypothesis testing in Section 5. This test relies on incrementally updated DNN (or more generally, any machine learning) models on batches of observations. We show in Proposition 4.3 that this test provides tight non-asymptotic type-I-error control under the null, and is consistent (i.e., rejects the null almost surely) against arbitrary fixed alternatives, under very mild conditions on the learning algorithm.

Finally, in Section 6, we instantiate and empirically evaluate our general test for several important applications, such as two-sample testing, conditional independence testing, group invariance testing and certifying adversarial robustness. Our empirical results show that the proposed framework offers tests that are competitive and often superior to state-of-the-art tests tailored to the specific tasks.

2 MOTIVATING APPLICATIONS

We now illustrate the utility of studying the abstract testing problem (1), by showing that it models several important applications in a unified manner.

Example 2.1 (Paired Two-sample testing)**.** Given a stream of paired observations: $\{(X_t, Y_t) : t \geq 1\}$ drawn i.i.d. from a distribution $P_X \times P_Y$ on a product space $\mathcal{X} \times \mathcal{X}$, our goal is to decide between the null, $H_0 : P_X = P_Y$, against the alternative $H_1 : P_X \neq P_Y$. This is a nonparametric testing problem with a composite null and a composite alternative. The null hypothesis class, however, has an interesting symmetry: the joint distribution of (X, Y) is the same as the joint distribution of (Y, X). We can formally state this as $H_0 : (X, Y) \stackrel{d}{=} \mathcal{T}_{\text{swap}}((X, Y))$, where $\mathcal{T}_{\text{swap}} : \mathcal{X} \times \mathcal{X} \to \mathcal{X} \times \mathcal{X}$, and $\mathcal{T}_{\text{swap}}((x, y)) = (y, x)$.

Example 2.2 (Conditional independence testing)**.** Given observations $\{(U_t, V_t, W_t) : t \geq 1\}$ drawn i.i.d. from P_{UVW}, we want to test whether $U \perp\!\!\!\perp V|W$ or not. This problem is fundamentally impossible without further assumptions (Shah and Peters, 2020), and a common structural assumption is that the conditional $P_{U|W}$ is known (the *model-X* assumption (Candes et al., 2018)). We can now reframe this problem as follows:

- Given (U, V, W), generate a new $\widetilde{U} \sim P_{U|W}(\cdot|W)$, and let Z denote $\big((U, V, W), (\widetilde{U}, V, W)\big)$.
- Let $\mathcal{T}_1$ and $\mathcal{T}_2$ denote the coordinate projections: $\mathcal{T}_1(Z) = (U, V, W)$ and $\mathcal{T}_2(Z) = (\widetilde{U}, V, W)$.

With these definitions, conditional independence (CI) testing falls under the abstract framework defined in (1).

Testing for invariance under group actions, such as rotations, is another instantiation of (1).

Example 2.3 (Rotation invariance testing)**.** Given a stream of observations $\{(X_t, Y_t) : t \geq 1\}$, where the X_t's denote images of the (handwritten) digit "6", while dataset Y_t's are images that, at a glance, represent the digit "9". However, these may essentially be the digit "6" but rotated. We aim to determine the statistical relationship between X_t and $\mathcal{T}_{180}(Y_t)$: the 180 degree rotations of Y_t. Essentially, we want to decide whether $(Y_t)_{t\geq 1}$ are merely rotated versions of "6", or truly represent the digit "9". Using the swap operator from Example 2.1, we define two distinct operators: $\mathcal{T}_1 = (\mathcal{T}_{180}, \mathcal{T}_{\text{id}}) \circ \mathcal{T}_{\text{swap}}$, and $\mathcal{T}_2 = (\mathcal{T}_{\text{id}}, \mathcal{T}_{180})$, with $\mathcal{T}_{\text{id}}$ being the identity mapping. Then, the above-defined test is equivalent to testing $H_0 : \mathcal{T}_1(Z) \stackrel{d}{=} \mathcal{T}_2(Z)$, where $Z = (X, Y)$.

Our general framework is not restricted to simple operators with closed-form expressions as in the examples above. In fact, the operators involved can even be general function approximators, such as large neural networks.

Example 2.4 (Adversarial Examples)**.** We now consider the problem of certifying the robustness of a trained machine learning model h to adversarial perturbations (Szegedy et al., 2014). In particular, let $\mathcal{T}_{\text{adv}}$ denote the adversarial attack that maps an input X to its adversarially perturbed version $\widetilde{X}$. Furthermore, let $\mathcal{T}_h$ denote the output of a specific layer (for example, a bottleneck layer) of the model. Then, our goal is to decide if the distributions of $Y = \mathcal{T}_h(X)$ and $\widetilde{Y} = \mathcal{T}_h(\widetilde{X})$ are equal or not. In other words, the null states that the distribution of X after applying $\mathcal{T}_h$ and $\mathcal{T}_h \circ \mathcal{T}_{\text{adv}}$ is the same.

Further examples of (1), such as tests for group fairness and independence, are available in Section 8.

3 RELATED WORK

The abstract testing problem studied in this paper is motivated by the general task of testing for invariance to the action of finite groups, stated as the *randomization hypothesis* by Lehmann and Romano (2022, Definition 17.2.1). In fact, the ideas we develop can also be extended to deal with the formulation of Lehmann and Romano (2022) (See Section 5).

From a methodological perspective, our techniques are related to the growing body of recent work on safe anytime-valid inference (SAVI), surveyed by Ramdas et al. (2022). Our design strategy follows the principle of *testing by betting* (Shafer, 2021), which states that the evidence against a null can be precisely characterized in terms of the gain in wealth of a (fictitious) bettor, who repeatedly bets on the observations in betting games with odds that are fair (or sub-fair) under H_0. This principle has been used by several authors, such as Shekhar and Ramdas (2023); Podkopaev et al. (2023); Podkopaev and Ramdas (2023); Shaer et al. (2023), to transform different hypothesis testing problems into that of designing relevant betting strategies and payoff functions. For example, Shekhar and Ramdas (2023) considered the two-sample testing problem (Example 2.1), and defined the wealth process of the bettor as follows for $t \geq 1$:

$$W_t = W_{t-1} \times \big(1 + \lambda_t\big(g_t(X_t) - g_t(Y_t)\big)\big)\,,$$

for some $[-1/2, 1/2]$-valued payoff functions $(g_t)_{t\geq 1}$, and for bets $\lambda_t \in [-1, 1]$ and $W_0 = 1$. By construction, the process $(W_t)_{t\geq 0}$ satisfies the requirement of fair payoffs under the null, as it is a non-negative martingale. The term λ_t is a predictable bet, whose absolute value denotes the fraction of the accumulated wealth that is placed at stake in round t. Hence, the approach of Shekhar and Ramdas (2023), reduces the problem of two-sample testing into that of developing appropriate strategies for selecting $(g_t)_{t\geq 1}$ (the *prediction strategy*) and $(\lambda_t)_{t\geq 1}$ (the *betting strategy*). There exist off-the-shelf betting strategies in the literature on online learning, such as the online Newton step (ONS) strategy (Hazan et al., 2007), that ensure exponential growth of the wealth process for arbitrary $(g_t)_{t\geq 1}$ under the alternative. For the prediction strategy, Shekhar and Ramdas (2023) suggested selecting $(g_t)_{t\geq 1}$ that approximate the *witness function* (g^*) associated with statistical distance metrics with variational representations (such as Kolmogorov-Smirnov metric, kernel MMD, f-divergence, etc). A similar approach was followed by Podkopaev et al. (2023) and Shaer et al. (2023) for the problems of independence and conditional independence testing.

Machine learning models (classifiers or regressors) have proven highly effective in developing tests for complex data structures; see (Kim et al., 2021) and references therein for more details. Even within the SAVI framework, some prior works such as Podkopaev and Ramdas (2023); Pandeva et al. (2022); Lhéritier and Cazals (2018) propose two-sample and independence tests based on classifiers. Unlike these works, our approach is geared towards a general class of problems modeled by (1) and uses models trained directly to optimize the statistical test performance.

4 METHODOLOGY

We study the testing problem described in (1), under the assumption that we observe a stream of datapoints arriving in mini-batches $(B_t)_{t\geq 1}$, where $B_t = \{Z_{(t-1)b+1}, \ldots, Z_{tb}\}$ denotes the t-th mini-batch consisting of b i.i.d. observations. Our objective is to design a procedure to continuously monitor the stream, aggregate evidence against the null, and stop and reject the null as soon as sufficient evidence is collected. Formally, such procedures are called *sequential tests of power one*, following Darling and Robbins (1968), and we state their definition below.

Definition 4.1. Given a significance level $\alpha \in (0, 1)$, and a stream of mini-batches, $\{B_t : t \geq 1\}$, consisting of i.i.d. samples drawn from a distribution P_Z, consider the testing problem introduced in (1). A level-α sequential test of power one for this problem is a stopping time, γ, adapted to the natural filtration $(\mathcal{F}_t)_{t\geq 0}$, with $\mathcal{F}_t = \sigma(B_1, \ldots, B_t)$, satisfying

$$\mathbb{P}_{H_0}(\gamma < \infty) \leq \alpha, \text{ and } \mathbb{P}_{H_1}(\gamma < \infty) = 1.$$

In other words, γ denotes a data-driven stopping rule at which the data-analyst stops collecting more data, and rejects the null. It is required that if the null holds, the probability that the test stops, i.e. it rejects the null, is bounded by α. In contrast, if the alternative is true this probability should be 1 which guarantees the consistency of the test.

4.1 Oracle Sequential Test

We begin by formulating an 'oracle' sequential test for (1), that assumes full knowledge of the true distribution P_Z. While this test is impractical, it provides the template for designing our practical data-driven sequential test.

Let $\mathcal{G} = \{g_\theta : \theta \in \Theta\}$ denote a class of machine learning models parameterized by Θ. For example, $\mathcal{G}$ might represent a class of deep learning models, with the parameter set Θ specifying the architecture. We make the following assumptions on this function class in this paper.

Assumption 1. *The parametrized function class $\mathcal{G} = \{g_\theta : \theta \in \Theta\}$ satisfies the following properties:*

- *Every $g \in \mathcal{G}$ satisfies $|g(x)| \leq q$ for all $x \in \mathcal{W}$, for some $q \in (0, 1/2)$.*
- *If a function g belongs to $\mathcal{G}$, then so does $c \cdot g$, for every $c \in [-1, 1]$.*

It is easy to verify that both these conditions can be satisfied by modifying any class of neural networks by using an appropriate activation function to restrict the output values in the required interval.

Since DNNs satisfy the universal approximation property for sufficiently large choices of the architecture (Hornik, 1991; Cybenko, 1989), for two distinct distributions $P \neq Q$ on $\mathcal{Z}$ with $P \circ \mathcal{T}_1^{-1} \neq P \circ \mathcal{T}_2^{-1}$, we can infer that

$$\sup_{\theta \in \Theta} \mathbb{E}\left[\widetilde{g}_\theta(Z, \mathcal{T}_1, \mathcal{T}_2)\right] > 0, \tag{2}$$

where $\widetilde{g}_\theta(z, \mathcal{T}_1, \mathcal{T}_2) := g_\theta \circ \mathcal{T}_1(z) - g_\theta \circ \mathcal{T}_2(z)$. In other words, if the class of "test-functions" $\mathcal{G}$ is rich enough, we can use it to distinguish between any two distinct transformed distributions $P \circ \mathcal{T}_1^{-1}$ and $P \circ \mathcal{T}_2^{-1}$. We now record a simple consequence of the assumptions made on the function class $\mathcal{G}$.

Proposition 4.2. *Under Assumption 1, the condition* (2) *is equivalent to*

$$\sup_{\theta \in \Theta} \mathbb{E}\left[\log\left(1 + \widetilde{g}_\theta(Z, \mathcal{T}_1, \mathcal{T}_2)\right)\right] > 0. \tag{3}$$

This statement is proven in Section 10.1 and is fundamental to the construction of the proposed test.

In the sequel, we will drop the $\mathcal{T}_1$ and $\mathcal{T}_2$ dependence of $\widetilde{g}$, and simply write $\widetilde{g}_\theta(z) \equiv \widetilde{g}_\theta(z, \mathcal{T}_1, \mathcal{T}_2)$. Using the above proposition, we can define the 'oracle' parameter, $\theta^* \equiv \theta^*(P_Z, \mathcal{T}_1, \mathcal{T}_2)$ as follows:

$$\theta^* \in \arg\max_{\theta \in \Theta} \mathbb{E}_{P_Z}\left[\log\left(1 + \widetilde{g}_\theta(Z)\right)\right].$$

Thus, θ^* represents the log-optimal function in $\mathcal{G}$, and we can use it to define an *oracle sequential test*

$$\gamma^* = \inf\{t \geq 1 : W_t^* \geq 1/\alpha\}, \tag{4}$$

where $W_t^* = \prod_{l=1}^t \prod_{Z \in B_l} (1 + \widetilde{g}_{\theta^*}(Z))$. It is easy to verify that γ^* is a sequential test according to Definition 4.1. In particular, it ensures the control of type-I error at level-α under H_0, and is finite almost surely under the alternative.

The test defined above is not practical, as it depends on the 'oracle' parameter θ^*, which is a function of P_Z. To construct a practical test, we instead use predictable empirical estimates of θ^*[1]. This is explained in detail in the following.

4.2 Practical sequential test

For the practical test, we propose to replace $(W_t^*)_{t \geq 0}$ in (4) with a data-driven process $(W_t)_{t \geq 0}$, that we refer to as the *wealth process* following the standard convention as discussed in Section 3. We set $W_0 = 1$, denoting the bettor's initial investment, and update W_t to $W_{t-1} \times S_t$ for $t \geq 1$, with S_t representing the gain (or loss) made while betting on the t-th batch of observations. We refer to this increment S_t as the *betting score* following Shafer (2021).

Algorithm 1 provides a detailed pseudocode describing the steps involved in the construction of our sequential test. The inputs to this algorithm include the stream of mini-batches $(B_t)_{t \geq 1}$, test-specific operators ($\mathcal{T}_1$ and $\mathcal{T}_2$), significance level $\alpha \in (0, 1)$, the maximum time horizon $T_{\max}$, and a deep learning (or any other machine learning) model initialized at θ_0. The algorithm then proceeds by repeating the following steps for all $t \geq 1$: it observes the next mini-batch B_t, computes the betting score S_t by calling the `ComputeScore` subroutine, and updates the model to θ_t by calling the `UpdateModel` subroutine. The updated wealth W_t is obtained by using the betting score S_t, and the algorithm stops

[1] As a warm-up, we also construct and theoretically analyze a practical batch test based on sample-splitting in Section 9 of the appendix.

and rejects the null if W_t exceeds the threshold $1/\alpha$. This general approach is illustrated for two-sample testing with paired observations in Figure 1.

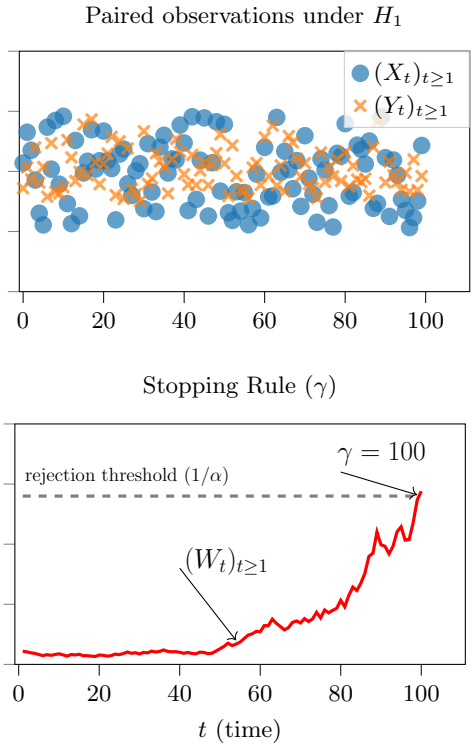

Figure 1: Illustration of our general strategy for the problem of two-sample testing with paired observations. The top figure shows the stream of observations under the alternative, while the bottom figure plots the variation of the wealth process $(W_t)_{t\geq 1}$. As we can see, after an initial period ($\approx t \leq 50$), the wealth process grows rapidly and exceeds the required threshold at $t = 100$, at which point we stop collecting more observations and reject the null.

To complete the description of our scheme, we now present the details of the two subroutines.

Compute Betting Score (`ComputeScore`). In round $t \geq 1$, this subroutine takes the inputs:

- $\mathcal{D}_{t-1} = \cup_{i=1}^{t-1} B_i$: the data observed so far.
- The model θ_{t-1}, trained on $\mathcal{D}_{t-1}$.
- $B_t = \{Z_{(t-1)b+1}, \ldots, Z_{tb}\}$: the new mini-batch.
- $\mathcal{T}_1, \mathcal{T}_2$: the operators defining the null.

Using these inputs, this subroutine computes and returns the next multiplicative increment, or betting score, (S_t) of the wealth process, which is defined as

$$S_t = \prod_{j=1}^{b} \left(1 + \widetilde{g}_{\theta_{t-1}}(Z_{(t-1)b+j})\right). \tag{5}$$

With this, we update the wealth process $W_t \leftarrow W_{t-1} \times S_t$. We reject the null if $W_t \geq \alpha^{-1}$, otherwise, we proceed to the next step:

Model Update (`UpdateModel`). This subroutine updates and returns a model θ_t on the data set $\mathcal{D}_t = \mathcal{D}_{t-1} \cup B_t$, i.e. θ_t maximizes the objective

$$\theta_t \in \arg\max_{\theta\in\Theta} \sum_{l=1}^{t} \sum_{Z\in B_l} \log(1 + \widetilde{g}_\theta(Z)) \tag{6}$$

The input `training_params` refers to all training parameters needed for model refinement. In particular, this includes parameters that are required for the optimization (such as the learning rate for the optimizer) and those that set the criteria for early stopping to prevent model overfitting. Importantly, our framework is versatile enough to incorporate any learning and model selection process, including methods such as cross-validation. For implementation details of this step see Sections 6 and 11.

Consistency. The stopping time constructed by Algorithm 1 can be formally defined as

$$\gamma = \inf\{t \geq 1 : W_t \geq 1/\alpha\}.$$

As mentioned earlier, γ represents the time at which the data-analyst stops collecting more observations (i.e., mini-batches), and declares the null to be not true. We now show that γ is finite under the null with a probability no larger than α, and under the alternative with probability 1 (assuming $T_{\max}$ is large enough). In simpler terms, the following theoretical result confirms that the sequential test from Algorithm 1 is consistent while maintaining non-asymptotic type I error control.

Proposition 4.3. *Suppose the learning algorithm satisfies the condition*

$$\liminf_{t\to\infty} \frac{\mathbb{E}[\log(1 + \widetilde{g}_{\theta_t}(Z))|\mathcal{F}_t]}{2c\sqrt{\log(t)/t}} \overset{a.s.}{>} 1, \quad \textit{under } H_1$$

for a universal constant c. Then, we have

$$\mathbb{P}_{H_0}(\gamma < \infty) \leq \alpha, \ \textit{and}\ \mathbb{P}_{H_1}(\gamma < \infty) = 1.$$

In words, γ is a sequential level-α test of power one.

The proof is provided in the appendix (Section 10.3).

Remark 4.4. The condition required of the learning algorithm by Proposition 4.3 for consistency of γ under H_1 is very mild. Informally, we only require $\mathbb{E}[\log(1+\widetilde{g}_{\theta_t}(Z))|\mathcal{F}_t]$ to be larger than $2c\sqrt{\log(tb)/tb}$ for large t, and in particular, this value can even converge to 0. In practice, most models converge to a local optimum θ_∞ with $\mathbb{E}[\log(1+\widetilde{g}_{\theta_\infty}(Z))] > 0$, which is much stronger than the condition required above. Such strong performance guarantees on learning algorithms can lead to stronger statistical properties of the test (such as bounds on $\mathbb{E}_{H_1}[\gamma]$). We leave such extensions to future work.

Algorithm 1: Sequential Test

Input: $\{B_t\}_{t\geq 1}$ (batch stream), $\mathcal{T}_1, \mathcal{T}_2$ (operators), $T_{\max}$ (maximum rounds of observations), α (size of the test), θ_0 (a trainable model).

$W_0 \leftarrow 1$, $\mathcal{D}_0 \leftarrow \emptyset$.
Initialize the model to an arbitrary value θ_0.
for $t \leftarrow 1$ **to** $T_{\max}$*:* **do**
  Observe the next batch $B_t = \{Z_{(t-1)b+1}, \ldots, Z_{tb}\}$;
  Compute the multiplicative increment: $S_t \leftarrow$ `ComputeScore` $(B_t, \theta_{t-1}, \mathcal{T}_1, \mathcal{T}_2, \sigma)$;
  Update the wealth process: $W_t \leftarrow W_{t-1} \times S_t$;
  Check for stopping condition:
  if $W_t \geq 1/\alpha$ **then**
    Stop and reject the null
  Increment Data: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup B_t$;
  Update the model: $\theta_t \leftarrow$ `UpdateModel`$(\mathcal{D}_t, \theta_{t-1},$ `training_params`$)$;

5 EXTENSION TO RANDOMIZATION HYPOTHESIS TESTING

The abstract problem (1) tests whether the data distributions, after being transformed by operators $\mathcal{T}_1$ and $\mathcal{T}_2$, are the same or not. We now study a generalization of this in which $\mathcal{T}_1$ and $\mathcal{T}_2$ could be one among finite disjoint classes of operators. This extension is motivated by the so-called randomization hypothesis assumption of Lehmann and Romano (2022, § 17.2.1). More formally, we consider testing problems with null defined as

$$H_0 : \mathcal{T}_1(Z) \stackrel{d}{=} \mathcal{T}_2(Z), \ \forall \mathcal{T}_1 \in \mathcal{O}_1, \forall \mathcal{T}_2 \in \mathcal{O}_2 \qquad (7)$$

for finite disjoint sets of operators $\mathcal{O}_1 \subset \mathcal{W}^{\mathcal{Z}}$ and $\mathcal{O}_2 \subset \mathcal{W}^{\mathcal{Z}}$. This formulation is highly flexible and covers a wide range of complex hypotheses, from those testing invariance under permutations to diverse group actions, as illustrated in Example 8.4. By using the general strategy we developed in Section 4, we can construct a sequential test for this problem, by simply modifying the training objective from (6) of the machine learning model as follows:

$$\theta_t \in \arg\max_{\theta\in\theta} \sum_{l=1}^{t} \sum_{Z\in B_t} \log\left(1 + \mathbb{E}_{\mathcal{T}_1,\mathcal{T}_2}\left[\widetilde{g}_\theta(Z, \mathcal{T}_1, \mathcal{T}_2)\right]\right),$$

$$\mathbb{E}_{\mathcal{T}_1,\mathcal{T}_2}\left[\widetilde{g}_\theta(Z, \mathcal{T}_1, \mathcal{T}_2)\right] = \frac{\sum_{\mathcal{T}_1,\mathcal{T}_2} g_\theta \circ \mathcal{T}_1(Z) - g_\theta \circ \mathcal{T}_2(Z)}{|\mathcal{O}_1||\mathcal{O}_2|}.$$

Essentially, the model aims to maximize the growth rate by calculating the average payoffs across all operator pairs. If there is a noticeable difference in one of the components $g_\theta \circ \mathcal{T}_1(Z) - g_\theta \circ \mathcal{T}_2(Z)$ it suggests the alternative hypothesis might hold. Similar to the previous section, the model can be incrementally refined with each new data batch. At the same time, the wealth process can be monitored with betting scores at time t defined as

$$S_t = \prod_{j=1}^{b} (1 + \mathbb{E}_{\mathcal{T}_1,\mathcal{T}_2}\left[\widetilde{g}_{\theta_{t-1}}(Z_{(t-1)b+j}, \mathcal{T}_1, \mathcal{T}_2)\right]).$$

Following the same arguments as Proposition 4.3, we can show that the resulting sequential test for (7) provides finite-sample type I error control and consistency guarantees.

6 EXPERIMENTS

In this section, we instantiate our general strategy, which we refer to as **DAVT**, to a wide range of tasks: two-sample testing, rotation invariance testing, robustness to adversarial attacks, and conditional independence under the model-X assumption. We compare DAVT to popular nonparametric baselines, including *sequential methods* such as the two-sample tests: E-C2ST (Lhéritier and Cazals, 2018; Pandeva et al., 2022) and Seq-IT with a batch-wise ONS betting strategy (Podkopaev and Ramdas, 2023) and the conditional independence test ECRT (Shaer et al., 2023). In this group of baselines, we include *permutation-based nonparametric tests* such as the MMD test (Gretton et al., 2012) and the classifier two-sample test (S-C2ST) (Kim et al., 2021; Lopez-Paz and Oquab, 2017). The latter two techniques have a "non-sequential" decision rule based on a p-value computed on a single batch. Although this approach may not be entirely appropriate, in our experiments we apply these tests in a sequential manner. More precisely, with each new batch

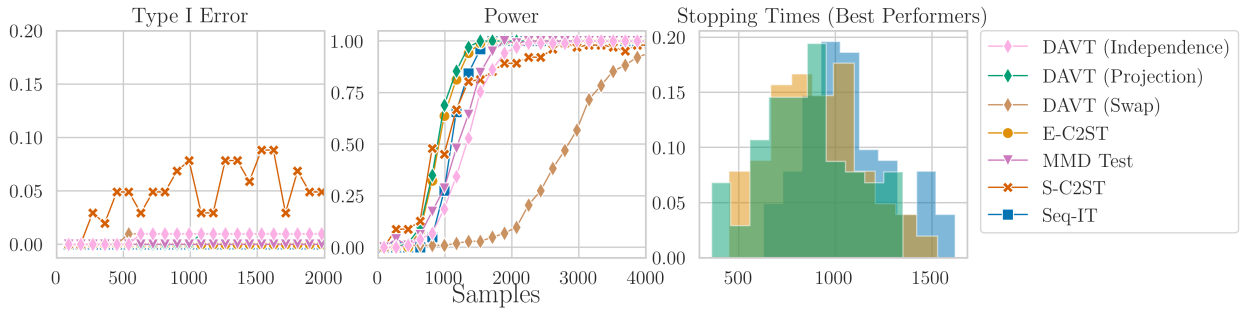

Figure 2: Power and type I error analysis for the Blob dataset. We compared three variations of our method (DAVT (Independence), DAVT (Swap), DAVT (Projection)) to sequential baselines (E-C2ST, Seq-IT) and non-sequential ones (S-C2ST, MMD Test). We fixed the batch size to be 90. DAVT (our method) with the projection operators (in green) and E-C2ST show the best performance, followed by Seq-IT. This order in performance is also confirmed by the histogram of the stopping times.

of data, we calculate a p-value to guide our testing process. For S-C2ST, this involves retraining the classifier on the previously collected data. However, unlike the sequential tests considered, we do not incorporate a stopping rule for the non-sequential S-C2ST and MMD tests.

For a fair comparison, we implement all considered data-driven tests using the same net architectures. Details of the selected DNNs, and their training procedures can be found in Section 11. The code to reproduce the experiments is provided at `https://github.com/tpandeva/deep-anytime-testing`. We evaluate all tests by monitoring the rejection rates (power and type I error) over time from 100 independent runs, performed at a significance level of $\alpha = 0.05$. In our empirical evaluation on standard benchmark datasets, DAVT shows competitive, if not superior, results to its specialized counterparts.

6.1 Two-Sample Testing

The two-sample testing problem can be modeled in several ways using distinct operators within our framework. For instance, in Example 2.1, we modeled it using the operators $\mathcal{T}_1 := \mathcal{T}_{\text{swap}}$ and $\mathcal{T}_2 := \mathcal{T}_{\text{id}}$. Alternatively, the operators $\mathcal{T}_1(x, y) = x$ and $\mathcal{T}_2(x, y) = y$ also characterize the two-sample testing problem which we refer to as DAVT-Projection. Finally, another option is to formulate two-sample testing as an instance of independence testing. This is achieved by introducing a binary variable L and a variable W such that $P(W|L = 1) = P(X)$ and $P(W|L = 0) = P(Y)$. Then, the two-sample test transforms into testing the independence of W and L, using the operator defined in Example 8.1.

Building on this observation, we evaluate the above two-sample tests along with the baselines on the Blob dataset (Chwialkowski et al., 2015). This dataset contains two classes of data, X and Y, both representing nine Gaussians on a two-dimensional grid that differ in their variances (see Figure 7). The results, summarized in Figure 2, show the superior performance of DAVT-Projection (our method) (shown in green), closely followed by the sequential methods E-C2ST and Seq-IT. Conversely, other DAVT variants did not deliver comparable performance. For example, using only the swap operator (DAVT-Swap) results in poor test performance. One possible explanation of this could be the inherent attempt of the neural network to find correlations between X and Y, which contradicts the problem setup that X and Y are independent. We explore the test performance for dependent X and Y in Section 11.3.3. Moreover, DAVT-Independence is not as powerful as the top sequential methods on this task, but it achieves maximum power faster than the non-sequential methods.

Overall, our two-sample experiment highlights the strengths of sequential methods over batch tests. By construction, well-designed sequential tests continuously monitor the data stream to accumulate evidence against the null. Thus, instead of setting a fixed sample size, using sequential methods allows for dynamically tailoring the sample size to the complexity of the task in a data-driven manner. This adaptability is illustrated in Figure 2, showing the stopping times distribution of the top three performers from the power experiment. Here, DAVT-Projection and E-C2ST reject the null more quickly than Seq-IT, indicating a more efficient use of data.

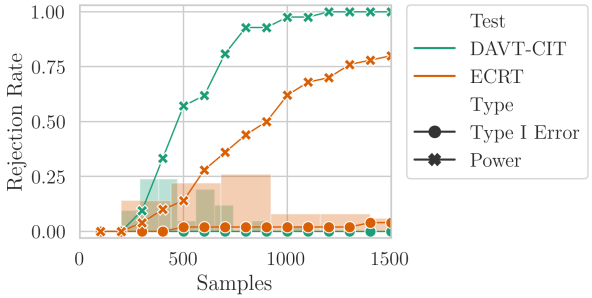

Figure 3: CIT Power and Type I error control. DAVT-CIT (ours) increases power faster than ECRT while keeping a very low Type I error.

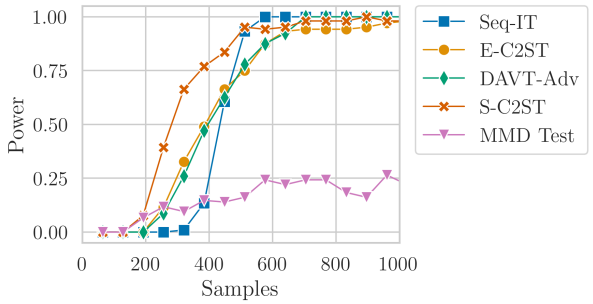

Figure 4: Power analysis for testing the adversarial robustness of ResNet50 trained on CIFAR-10. While S-C2ST initially outperforms in scenarios with limited data, DAVT-Adv (our method) and Seq-IT catch up and accelerate to reach maximum performance at a faster rate, leaving DAVT-Adv as the second best-performing method.

6.2 Conditional Independence Testing under Model-X Assumption

We use the synthetic example of (Shaer et al., 2023) with the following data generation model. We construct mini-batches containing 100 observations of the random variables U, V, W for which we test $H_0 : V \perp\!\!\!\perp U|W$. These triples come from the model $W \sim \mathcal{N}(0, I_d)$ and $U|W = w \sim \mathcal{N}(a^\top w, 1)$, where under the alternative hypothesis, we use $V|W = w, U = u \sim \mathcal{N}((b^\top w)^2 + 3w, 1)$, while under the null hypothesis we use $V|W = w, U = u \sim \mathcal{N}((b^\top w)^2, 1)$. We recall that the model-X assumption implies that at testing time t, we have access to the true data distribution $U|W$.

We instantiate our conditional independence test (called the DAVT-CIT) using the operators defined in Example 2.2, and benchmark its performance against the sequential method ECRT (Shaer et al., 2023). Note that we have tailored ECRT to fit our framework, that is, both ECRT and DAVT-CIT employ the same network architecture for model fitting and use a single sample of $\tilde{U}$, at each step t to estimate the betting scores S_t.

The type-I error and power of these two tests over 100 trials is plotted in Figure 3. While both methods control type-I error at the required level $\alpha = 0.05$, our test (DAVT-CIT) requires significantly fewer observations (on average) to reject the null under H_1. The histograms of the two stopping times in Figure 3 further demonstrate the better sample efficiency of DAVT-CIT compared to ECRT.

6.3 Adversarial Attacks on ResNet50

In this experiment, we evaluate the robustness of a ResNet50 model (He et al., 2016), refined on the CIFAR-10 dataset (Krizhevsky and Hinton, 2009), against adversarial attacks as explained in Example 2.4. We employ the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) for generating adversarial samples. Within this context, let X represent the true CIFAR-10 images and $\mathcal{T}_{adv}(X)$ the biased ones produced with FGSM. Moreover, let the operator $\mathcal{T}_h : \mathcal{X} \to \mathbb{R}^{10}$ map an image to ResNet50 last layer[2]. In this example, we consider ResNet50 to be robust against FGSM adversarial attacks if $\mathcal{T}_h(X)$ and $\mathcal{T}_h \circ \mathcal{T}_{adv}(X)$ have the same distribution. In other words, we will test:

$$H_0 : \mathcal{T}_h(X_t) \overset{d}{=} \mathcal{T}_h \circ \mathcal{T}_{adv}(X).$$

We run 100 power experiments on batches of paired original and FGSM-altered images, with a sample size of 64. We compare DAVT-Adv with the two-sample test baselines: E-C2ST, Seq-IT, S-C2ST, and the MMD test. Figure 4 shows that of the sequential methods, DAVT-Adv (ours) outperforms E-C2ST and Seq-IT until it reaches 60% power. After that, Seq-IT achieves maximum power faster, leaving our method as the second best. Nevertheless, the non-sequential S-C2ST initially shows greater power than all other methods, but this advantage declines with increasing sample size.

6.4 Rotation Invariance Testing

Here, we consider a test similar to Example 2.3 and extend it to the more general setting described in

[2]Any intermediate layer of the network can be considered while defining an adversarial operator.

Section 5. More precisely, we consider that at each time t we are given batches of rotated images X following the generative model $P_X = 0.5p \cdot P_{90} + 0.5(1-p) \cdot P_{180} + 0.5p \cdot P_{270} + 0.5(1-p) \cdot P_{360}$, where p is the mixing weight. Here the distributions P_{90}, P_{180}, P_{270} and P_{360} represent the distribution of the randomly rotated "6" at angles in the set $\{90, 180, 270, 360\}$ degrees. This generative model is initially unknown to the practitioner, who wants to determine whether the distribution of X is invariant to rotations of 90, 180, or 270 degrees. Thus, by applying ideas from Section 5 we can form the null hypothesis:

$$H_0 : \mathcal{T}_i(X) \overset{d}{=} X \text{ for all } i \in \{90, 180, 270\}$$

where the operators correspond to the specified rotations. This test can be easily adapted to our framework by specifying $Z := X$ and defining the operator set $\mathcal{O} := \{\mathcal{T}_i : i \in \{90, 180, 270, 360\}\}$.

For constructing the two distributions, we use the MNIST dataset (LeCun et al., 2010). We conduct experiments for $p = 0.3, 0.4$, and 0.5 in batches of 32 samples each. Figure 5 presents the power results for $p = 0.3$ and $p = 0.4$, and the type I error rate when $p = 0.5$. The test successfully controls the type I error and shows reduced power in the more challenging case of $p = 0.4$, compared to $p = 0.3$.

We create a baseline test by applying S-C2ST to each hypothesis, each based on a single operator. This process involves training a different neural net for every hypothesis and then computing its associated p-value during testing. We then consolidate the three derived p-values using the Bonferroni correction. Figure 5 shows the rejection rates of this method for $p = 0.3, 0.4, 0.5$ over time and highlights the lack of power of the method. This is a common effect when multiple correction procedures are used which vindicates the use of sequential tests.

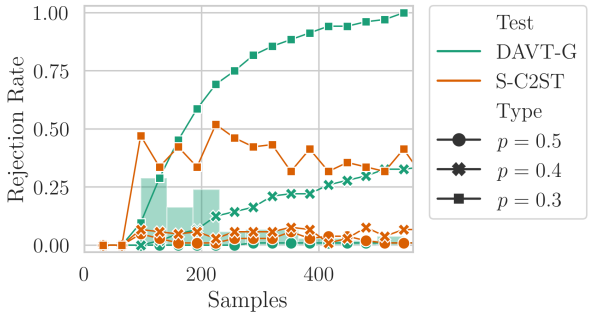

Figure 5: Power/Type I error analysis for the rotated 6 MNIST images for different mixing proportions $p = 0.3, 0.4, 0.5$. DAVT(ours) shows better power performance than the baseline S-C2ST combined with multiple testing corrections.

7 DISCUSSION

In this paper, we developed a unified deep learning-based approach for constructing sequential tests for an abstract class of nonparametric testing problems. This class of problems includes various important applications ranging from two-sample testing and independence testing to certifying adversarial robustness and group fairness of machine learning models. Our sequential test provides tight control over the type-I error, and is consistent under very mild conditions on the learning algorithm. Through extensive empirical evaluation, we show that our general testing strategy, when instantiated to several practical applications, performs competitively with existing sequential tests specifically designed for those tasks.

Our work opens up several interesting directions for future work. For example, from a theoretical perspective, obtaining stronger guarantees on the performance of our test, perhaps by leveraging recent advances in deep learning theory (Jacot et al., 2018), is an important question. On the practical front, some interesting topics include designing improved rules for computing the mini-batch betting scores (see Section 12 for more details), and identifying appropriate regularizations or modifications of the training objective functions to learn better joint betting and payoff strategies. Furthermore, our experiments suggest that an important direction for future research is to develop principled strategies for integrating network architecture search, operator design, and hyperparameter tuning schemes into our general sequential testing framework.

References

E. Candes, Y. Fan, L. Janson, and J. Lv. Panning for Gold: 'Model-X' Knockoffs for High Dimensional Controlled Variable Selection. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 80(3):551–577, 2018.

K. P. Chwialkowski, A. Ramdas, D. Sejdinovic, and A. Gretton. Fast Two-Sample Testing with Analytic Representations of Probability Measures. *Advances in Neural Information Processing Systems*, 28, 2015.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

D. A. Darling and H. Robbins. Some nonparametric sequential tests with power one. *Proceedings of the National Academy of Sciences*, 61(3):804–809, 1968.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2014.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A Kernel Two-Sample Test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

P. Grünwald, R. de Heide, and W. M. Koolen. Safe Testing. *Journal of the Royal Statistical Society: Series B (to appear with discussion)*, 2023.

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69:169–192, 2007.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

A. Jacot, F. Gabriel, and C. Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. *Advances in neural information processing systems*, 31, 2018.

I. Kim, A. Ramdas, A. Singh, and L. Wasserman. Classification accuracy as a proxy for two-sample testing. *The Annals of Statistics*, 49(1):411–434, 2021.

A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

E. L. Lehmann and J. P. Romano. *Testing Statistical Hypotheses*. Springer, 4th edition, 2022.

A. Lhéritier and F. Cazals. A Sequential Non-Parametric Multivariate Two-Sample Test. *IEEE Transactions on Information Theory*, 64(5):3361–3370, 2018.

D. Lopez-Paz and M. Oquab. Revisiting Classifier Two-Sample Tests. In *International Conference on Learning Representations*, 2017.

T. Pandeva, T. Bakker, C. A. Naesseth, and P. Forré. E-Valuating Classifier Two-Sample Tests. *arXiv preprint arXiv:2210.13027*, 2022.

A. Podkopaev and A. Ramdas. Sequential Predictive Two-Sample and Independence Testing. *Advances in neural information processing systems*, 2023.

A. Podkopaev, P. Blöbaum, S. Kasiviswanathan, and A. Ramdas. Sequential Kernelized Independence Testing. In *International Conference on Machine Learning*, pages 27957–27993. PMLR, 2023.

A. Ramdas, P. Grünwald, V. Vovk, and G. Shafer. Game-Theoretic Statistics and Safe Anytime-Valid Inference. *arXiv preprint arXiv:2210.01948*, 2022.

S. Shaer, G. Maman, and Y. Romano. Model-X Sequential Testing for Conditional Independence via Testing by Betting. In *International Conference on Artificial Intelligence and Statistics*, pages 2054–2086. PMLR, 2023.

G. Shafer. Testing by betting: A strategy for statistical and scientific communication. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 184(2):407–431, 2021.

R. D. Shah and J. Peters. The hardness of conditional independence testing and the generalised covariance measure. *The Annals of Statistics*, 48 (3):1514–1538, 2020.

S. Shekhar and A. Ramdas. Nonparametric two-sample testing by betting. *IEEE Transactions on Information Theory*, 2023.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

J. Ville. Etude Critique de la Notion de Collectif. *Gauthier-Villars, Paris.*, 1939.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes, the details of our methodology are in Section 4]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, empirical evaluation of the sample size efficiency is provided for every experiment in Section 6, the other requirements are not applicable because they are linked to the complexity of the task not the testing procedure.]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes, we provide a link to the code `https://github.com/tpandeva/deep-anytime-testing`.]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes, the proofs are in Section 10 of the Appendix.]
   (c) Clear explanations of any assumptions. [Yes, see Remark 4.4.]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes, the code is available as a URL.]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes it is provided in Section 11 in the appendix]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes, it is explained in Sections 6 and 11)]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes, available in the Appendix (Section 11)]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Deep anytime-valid hypothesis testing: Supplementary Materials

8 MORE EXAMPLES

Example 8.1 (Independence Testing)**.** Independence testing is another well-studied problem in statistics, where given observations $\{(X_i, Y_i) : 1 \leq i \leq n\}$ drawn i.i.d. from a distribution P_{XY} on a product space $\mathcal{X} \times \mathcal{Y}$, we want to test whether $P_{XY} = P_X \times P_Y$ or not. By working with two pairs of observations at a time, we can again describe the null as being invariant to an operator. In particular, let given $Z_1 = (X_1, Y_1)$ and $Z_2 = (X_2, Y_2)$, let $\mathcal{T}$ denote the operator that maps (Z_1, Z_2) to (Z_1', Z_2'), with $Z_1' = (X_1, Y_2)$ and $Z_2' = (X_2, Y_1)$. Clearly, the distribution of (Z_1, Z_2) is the same as that of (Z_1', Z_2') under the null, while this invariance to $\mathcal{T}$ is broken under the alternative.

Example 8.2 (Symmetry testing)**.** In the simplest version of this problem, we consider real-valued observations (that is, $\mathcal{Z} = \mathbb{R}$), and the operators $\mathcal{T}_2 = \mathcal{T}_{\text{id}}$, and $\mathcal{T}_1 = \mathcal{T}_{\text{flip}}$, where the operator $\mathcal{T}_{\text{flip}}$ simply flips the observations about the origin; that is, $\mathcal{T}_{\text{flip}}(x) = -x$. The resulting null hypothesis asserts that P_Z is symmetric about the origin. The same formulation also covers other kinds of symmetry, such as rotational invariance, or invariance to horizontal or vertical flips in the case of images.

Example 8.3 (Group Fairness)**.** Group fairness, sometimes referred to as demographic or statistical fairness, is a research area in machine learning that focuses on how machine learning models perform across different demographic groups. The main goal is to ensure that a model's performance is consistent across predefined groups, avoiding situations where the model may disproportionately benefit or harm a particular group.

A typical application in this context would be testing whether an ML model is racially biased. For example, suppose a trained ML recommendation model h is used to predict which candidates are most likely to succeed in a job. The company running this model wants to ensure that it is not racially biased. To achieve this, they categorize applicants into p ethnic groups and then evaluate whether the model h produces consistent results across all p groups. Let Y be the categorical random variable indicating the demographic group and X be a random vector collecting the rest of the applicant's covariates. The associated statistical test is

$$H_0 : h(X) \perp\!\!\!\perp Y$$

Thus, by using Example 8.1, we can design a test with respect to the defined null based on our framework.

Example 8.4 (Group invariance testing)**.** Suppose we have a collection of images Z containing equilateral triangles. Each edge of these triangles is colored either blue or green. A practitioner would like to find out if the edges of the triangles are colored without any particular pattern, or if some hidden rule controls their coloring. To do this, we will examine whether the triangles remain the same when rotated 120 or 240 degrees. We will therefore introduce a set of operators that represent the aforementioned rotations: $\mathcal{T}_{120}$ and $\mathcal{T}_{240}$. Next, we formulate the following *composite* null hypothesis:

$$H_0 : \mathcal{T}_{120}(Z) = Z \text{ and } \mathcal{T}_{240}(Z) = Z$$

Thus, we want to test whether the distribution of Z remains invariant with respect to the two operators $\mathcal{T}_{120}$ and $\mathcal{T}_{240}$.

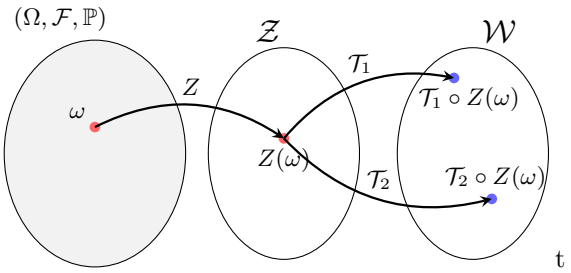

Figure 6: Let Z denote a $\mathcal{Z}$-valued random variable on an underlying probability space $(\Omega, \mathcal{F}, \mathbb{P})$. By definition, the distribution of the Z is equal to $P_Z = \mathbb{P} \circ Z^{-1}$. The two black curved lines from $\mathcal{Z}$ to $\mathcal{W}$ denote the operators $\mathcal{T}_i$ for $i \in \{1, 2\}$, used to characterize the class of null distributions. In particular, the distribution of the resulting $\mathcal{W}$-valued random variables are $\mathbb{P} \circ (\mathcal{T}_i \circ Z)^{-1} = \mathbb{P} \circ Z^{-1} \circ \mathcal{T}_i^{-1} = P_Z \circ \mathcal{T}_i^{-1}$, and the null hypothesis of our abstract testing problem states that the two distributions, $P_Z \circ \mathcal{T}_1^{-1}$ and $P_Z \circ \mathcal{T}_2^{-1}$, are the same.

9 BATCH TEST BASED ON SAMPLE-SPLITTING

We can also construct a batch test (also called a fixed sample-size test) for the abstract testing problem (1) using the idea of sample splitting. In particular, let $\mathcal{D} = \{Z_i : 1 \leq i \leq n\}$ denote the set of observations, which are then split into two equal halves, $\mathcal{D}_1 = \{Z_1, \ldots, Z_{n/2}\}$ and $\mathcal{D}_2 = \{Z_{n/2+1}, \ldots, Z_n\}$. We use the first split, $\mathcal{D}_1$, to train an ML model (usually a DNN) with the objective of maximizing the growth rate:

$$\widehat{\theta} \in \operatorname*{arg\,min}_{\theta \in \Theta} \; -\frac{2}{n} \sum_{Z_i \in \mathcal{D}_1} \log\left(1 + \widetilde{g}_\theta(Z_i)\right).$$

Next, we use the learned parameter on the second split to construct the test statistic

$$E_n = \prod_{Z_i \in \mathcal{D}_2} \left(1 + \widetilde{g}_{\widehat{\theta}}(Z_i)\right).$$

We expect this statistic E_n to be small under the null, and thus we can use it to define a test for (1) that rejects the null for large values of E_n. Our next result analyzes the performance of such a test.

Proposition 9.1. *Suppose the learning algorithm ensures that*

$$\liminf_{n \to \infty} \frac{\mathbb{E}[\log(1 + \widetilde{g}_{\widehat{\theta}}(Z)) | \mathcal{D}_1]}{4c\sqrt{\log n / n}} \overset{a.s.}{>} 1, \quad \textit{under } H_1,$$

where c is a universal constant. Then, the test $\Psi(Z^n) = \mathbb{1}_{E_n \geq 1/\alpha}$, that rejects the null if E_n exceeds $1/\alpha$, satisfies the following properties:

$$\mathbb{E}_{H_0}\left[\Psi(Z^n)\right] \leq \alpha, \quad \textit{and} \quad \lim_{n \to \infty} \mathbb{E}_{H_1}\left[\Psi(Z^n)\right] = 1.$$

That is, Ψ is a consistent, level-α test for (1).

The proof of this result is in Section 10.2 of the appendix.

Note that the test statistic E_n is an *e-variable* (Grünwald et al., 2023); which is a nonnegative random variable with an expected value no larger than 1 under the null. As a result, the test Ψ is valid under *optional continuation*. That is, suppose we compute E_n using a dataset $\mathcal{D}$, and its value turns out to be smaller than $1/\alpha$. Since the test Ψ based on $\mathcal{D}$ is inconclusive, we may decide to collect m further observations, $\mathcal{D}'$, and use it to compute E'_m. We can combine the evidence from the two experiments easily by simply rejecting the null if $E_n \times E'_m$ exceeds $1/\alpha$, without violating type-I error guarantees. This is a simple consequence of the fact that $\mathbb{E}[E_n E'_m] = \mathbb{E}[E_n \mathbb{E}[E'_m | \mathcal{D}]] \leq \mathbb{E}[E_n] \leq 1$.

10 DEFERRED PROOFS

In this section, we present the proofs of Proposition 4.2, along with the two results analyzing the performance of our batch-test Ψ (Proposition 9.1), and the sequential test γ (Proposition 4.3).

10.1 Proof of Proposition 4.2

We need to prove that (2) $\iff$ (3); that is,

$$A \coloneqq \sup_{\theta\in\Theta} \mathbb{E}\left[\widetilde{g}_\theta(Z)\right] > 0 \quad \iff \quad B \coloneqq \sup_{\theta\in\Theta} \mathbb{E}\left[\log\left(1+\widetilde{g}_\theta(Z)\right)\right] > 0.$$

Proof of ($\Rightarrow$). Consider any ϵ such that $0 < \epsilon < A$. Then, by the definition of supremum, there exists a $\theta \in \Theta$ such that $\mathbb{E}[\widetilde{g}_\theta(Z)] \geq \epsilon$. Now, due to the second part of Assumption 1, we have

$$B \geq \sup_{c\in[-1,1]} \mathbb{E}\left[\log\left(1+c\,\widetilde{g}_\theta(Z)\right)\right].$$

The first part of Assumption 1 implies that $|c\widetilde{g}_\theta(z)| < 1$ for all $z \in \mathcal{Z}$. Furthermore, for $x \leq 0.68$, we know that $\log(1+x) \geq x - x^2/2$. Using these facts, we obtain

$$\begin{aligned} B &\geq \sup_{c\in[-1,1]} \mathbb{E}\left[\log\left(1+c\,\widetilde{g}_\theta(Z)\right)\right] \\ &\geq \sup_{c\in[-0.68,0.68]} \mathbb{E}\left[c\,\widetilde{g}_\theta(Z)\right] - \frac{1}{2}\mathbb{E}\left[\left(c\,\widetilde{g}_\theta(Z)\right)^2\right] \\ &\geq \sup_{c\in[-0.68,0.68]} c\epsilon - \frac{bc^2}{2}, \end{aligned}$$

where $b = \mathbb{E}\left[\left(\widetilde{g}_\theta(Z)\right)^2\right]$. If $b = 0$, then we trivially have $B \geq 0.68\epsilon > 0$. Now, if $b > 0$, the function $f(c) = c\epsilon - c^2 b/2$ has two real roots at $c = 0$ and $c = \epsilon/b > 0$, which implies that there exists a $c \in [0, \min\{0.68, \epsilon/b]$ at which $f(c) > 0$. In other words, there exists a function $g \in \mathcal{G}$ such that $\mathbb{E}[\log(1+\widetilde{g}(Z))] > 0$, as required.

Proof of ($\Leftarrow$). Consider any ϵ such that $0 < \epsilon < B$. Then, by the definition of supremum, there exists a $\theta \in \Theta$ such that $\mathbb{E}[\log(1+\widetilde{g}_\theta(Z))] \geq \epsilon$. This implies that

$$\epsilon \leq \mathbb{E}\left[\log\left(1+\widetilde{g}_\theta(Z)\right)\right] \overset{(i)}{\leq} \log\left(1+\mathbb{E}[\widetilde{g}_\theta(Z)]\right).$$

The inequality (i) above follows from an application of Jensen's inequality due to the concavity of the $\log(\cdot)$ function. From the above inequality, we can obtain the required result as follows:

$$0 < e^\epsilon - 1 \leq \mathbb{E}[\widetilde{g}_\theta(Z)] \leq \sup_\theta \mathbb{E}[\widetilde{g}_\theta(Z)] = A.$$

10.2 Proof of Proposition 9.1

Type-I error control. The proof of the type-I error control follows from an application of Markov's inequality. More specifically, the expected value of E_n can be expressed as follows:

$$\begin{aligned} \mathbb{E}[E_n] = \mathbb{E}[\mathbb{E}[E_n|\mathcal{D}_1]] &= \mathbb{E}\left[\mathbb{E}\left[\prod_{Z_i\in\mathcal{D}_2}\left(1+\widetilde{g}_{\widehat{\theta}}(Z_i)\right) \mid \mathcal{D}_1\right]\right] \\ &= \mathbb{E}\left[\prod_{Z_i\in\mathcal{D}_2}\left(1+\mathbb{E}\left[\widetilde{g}_{\widehat{\theta}}(Z_i) \mid \mathcal{D}_1\right]\right)\right], \end{aligned} \tag{8}$$

where $\mathcal{D}_1$ and $\mathcal{D}_2$ denote the two splits of the dataset $\mathcal{D}$. Recall that the parameter $\widehat{\theta}$ trained on the first split $\mathcal{D}_1$, and thus $\widetilde{g}_{\widehat{\theta}}$ can be treated as a constant function in the conditional expectation above. The equality in (8) uses the fact that dataset $\mathcal{D}$, and hence the split $\mathcal{D}_2$ consists of i.i.d. data-points. Finally, we have

$$\mathbb{E}\left[\widetilde{g}_{\widehat{\theta}}(Z_i) \mid \mathcal{D}_1\right] = 0,$$

under H_0 for any $Z_i \in \mathcal{D}_2$, which implies that

$$\mathbb{E}[E_n] = 1, \quad \text{under } H_0.$$

This immediately implies the type-I error guarantee of our test Ψ, since

$$\mathbb{P}_{H_0}\left(\Psi(Z^n) = 1\right) = \mathbb{P}_{H_0}\left(E_n \geq 1/\alpha\right) \leq \frac{\mathbb{E}_{H_0}[E_n]}{1/\alpha} = \frac{1}{1/\alpha} = \alpha.$$

The inequality above is due to Markov's inequality.

Consistency. For proving the consistency of our test, we need some additional notation:

$$v_i = \log\left(1 + \widetilde{g}_{\widehat{\theta}}(Z_{n/2+i})\right), \text{ for } i \in \{1, \ldots, n/2\}, \quad \text{and} \quad A_n \coloneqq \mathbb{E}\left[\widetilde{g}_{\widehat{\theta}}(Z)|\mathcal{D}_1\right],$$

where $Z \sim P_Z$ is independent of $\mathcal{D}_1$. Now, observe the following

$$\begin{aligned}\mathbb{P}\left(\Psi(Z^n) = 1\right) &= \mathbb{P}\left(\frac{2\log E_n}{n} \geq \frac{2\log(1/\alpha)}{n}\right) = \mathbb{P}\left(\frac{2}{n}\sum_{i=1}^{n/2} v_i \geq \frac{2\log(1/\alpha)}{n}\right) \\ &= \mathbb{P}\left(A_n + \frac{2}{n}\sum_{i=1}^{n/2}(v_i - A_n) \geq \frac{2\log(1/\alpha)}{n}\right). \end{aligned} \tag{9}$$

Now, we introduce the event $G_n = \{|(2/n)\sum_{i=1}^{n/2} v_i - A_n| \leq c\sqrt{4\log n/n}\}$, for $c = 2\log(1/(1-2q))$ and $q \in (0, 1/2)$ is the upper bound on $|g_\theta(x)|$ for all x, θ assumed in Section 4. Note that for each i, the random variable $v_i - A_n$ is bounded in $[-c/2, c/2]$, with mean 0. This means that, by an application of Hoeffding's inequality, we get

$$\mathbb{P}\left(G_n^c\right) \leq \frac{2}{n^2}, \quad \text{which implies that} \quad \sum_{n=2}^{\infty} \mathbb{P}(G_n^c) < \infty. \tag{10}$$

Returning to (9), we now get

$$\begin{aligned}\mathbb{P}\left(\Psi(Z^n) = 1\right) &\geq \mathbb{P}\left(\left\{A_n + \frac{2}{n}\sum_{i=1}^{n/2}(v_i - A_n) \geq \frac{2\log(1/\alpha)}{n}\right\} \cap G_n\right) \\ &\geq \mathbb{P}\left(\left\{A_n \geq \frac{2\log(1/\alpha)}{n} + 2c\sqrt{\frac{\log n}{n}}\right\} \cap G_n\right).\end{aligned}$$

The second inequality above uses the fact that under the event G_n, the term $(2/n)\left(\sum_{i=1}^{n/2} v_i - A_n\right)$ is lower bounded by $-2c\sqrt{\log n/n}$.

For large enough values of n, the term $2\log(1/\alpha)/n$ is smaller than $2c\sqrt{\log n/n}$. Using this fact, we obtain

$$\mathbb{P}(\Psi(Z^n) = 1) \geq \mathbb{P}\left(H_n \cap G_n\right) = \mathbb{E}[\mathbb{1}_{H_n}\mathbb{1}_{G_n}], \quad \text{with } H_n \coloneqq \left\{B_n \geq 4c\sqrt{\frac{\log n}{n}}\right\}. \tag{11}$$

Now, taking the limiting value of the probability of detection, we get

$$1 \geq \liminf_{n\to\infty} \mathbb{P}\left(\Psi(Z^n) = 1\right) \geq \liminf_{n\to\infty} \mathbb{E}[\mathbb{1}_{H_n}\mathbb{1}_{G_n}] \geq \mathbb{E}[\liminf_{n\to\infty} \mathbb{1}_{H_n}\mathbb{1}_{G_n}],$$

where the last inequality follows by an application of Fatou's Lemma.

To complete the proof, it suffices to show that $\mathbb{1}_{H_n}\mathbb{1}_{G_n} \xrightarrow{a.s.} 1$, which would imply that $\lim_{n\to\infty}\mathbb{P}\left(\Psi(Z^n)=1\right)=1$. We show this in two steps:

- From (10), and an application of (the first) Borel-Cantelli Lemma, we know that

$$\mathbb{P}\left(G_n^c \text{infinitely often}\right) = \mathbb{P}\left(\cap_{n=1}^{\infty}\cup_{m\geq n} G_m^c\right) = 0.$$

  On taking the complement of the event above, we get

$$\mathbb{P}\left(\cup_{n\geq 1}\cap_{m\geq n} G_n\right) = 1, \quad \text{which implies that} \quad \mathbb{1}_{G_n} \xrightarrow{a.s.} 1. \tag{12}$$

- For the final step, we use the assumption about the learning algorithm made in Proposition 9.1. In particular, the assumption that

$$\liminf_{n\to\infty} \frac{A_n}{4c\sqrt{\log n/n}} \overset{a.s.}{>} 1 \quad \text{implies} \quad \mathbb{1}_{H_n} \xrightarrow{a.s.} 1.$$

Together, (11) and (12) imply the required condition that $\mathbb{1}_{G_n}\mathbb{1}_{H_n} \xrightarrow{a.s.} 1$. This completes the proof.

10.3 Proof of Proposition 4.3

Type-I error control. The type-I error control is a consequence of the fact that the process $\{W_t : t \geq 1\}$ is a non-negative martingale with an initial value of 1. As a result, we have

$$\mathbb{P}(\gamma < \infty) = \mathbb{P}\left(\exists t \geq 1 : W_t \geq 1/\alpha\right) \leq \frac{\mathbb{E}[W_0]}{1/\alpha} = \alpha,$$

due to an application of Ville's inequality (Ville, 1939).

Consistency. Recall that we use t to denote the mini-batch counter, and b to denote the size of each mini-batch. To prove the consistency of this test, we begin by observing that

$$\mathbb{P}(\gamma = \infty) = \mathbb{P}\left(\cap_{t\geq 1}\{\gamma > t\}\right) \leq \mathbb{P}\left(\gamma > t\right),$$

for any arbitrary t. Taking the limit, this implies that

$$\mathbb{P}\left(\gamma = \infty\right) \leq \limsup_{t\to\infty} \mathbb{P}\left(\gamma > t\right).$$

To complete the proof, we will show that the RHS above is equal to 0. As in the proof of Proposition 9.1, we introduce the notation

$$v_i = \sum_{Z\in B_i} \log\left(1+\widetilde{g}_{\theta_{i-1}}(Z)\right), \quad \text{and} \quad A_i = \mathbb{E}\left[v_i|\mathcal{F}_{i-1}\right] = b\times\mathbb{E}[\log(1+\widetilde{g}_{\theta_{i-1}}(Z))|\mathcal{F}_{i-1}],$$

where $\mathcal{F}_{i-1} = \sigma\left(\cup_{j=1}^{i-1}B_j\right)$ is the σ-algebra generated by the first $i-1$ batches of observations. Then, we have

$$\mathbb{P}\left(\gamma > t\right) \leq \mathbb{P}\left(\frac{\log W_t}{t} < \frac{\log(1/\alpha)}{t}\right) = \mathbb{P}\left(\frac{1}{t}\sum_{i=1}^{t} v_i - A_i + \frac{1}{t}\sum_{i=1}^{t} A_i < \frac{\log(1/\alpha)}{t}\right). \tag{13}$$

Now, observe that the process $\{v_i - A_i : i \geq 1\}$ is a bounded martingale difference sequence. Hence, an application of Azuma's inequality gives us

$$\mathbb{P}\left(G_t^c\right) \leq \frac{2}{t^2}, \quad \text{with } G_t \coloneqq \left\{\left|\frac{1}{t}\sum_{i=1}^{t} v_i - A_i\right| \leq cb\sqrt{\frac{\log t}{t}}\right\},$$

where we have $c = 2\log(1/(1-2q))$, and $q \in (0, 1/2)$ is the upper bound on $|g_\theta(x)|$ for all x, θ assumed in Section 4. Combining the above result with (13), we get

$$\begin{aligned}
\mathbb{P}(\gamma > t) &\leq \mathbb{P}\left(\left\{\frac{1}{t}\sum_{i=1}^{t} A_i < \frac{\log(1/\alpha)}{t} + \left|\frac{1}{t}\sum_{i=1}^{t} v_i - A_i\right|\right\} \cap G_t\right) + \mathbb{P}(G_t^c) \\
&\leq \mathbb{P}\left(\left\{\frac{1}{t}\sum_{i=1}^{t} A_i < \frac{\log(1/\alpha)}{t} + cb\sqrt{\frac{\log t}{t}}\right\} \cap G_t\right) + \mathbb{P}(G_t^c) \\
&\leq \mathbb{P}\left(\frac{1}{t}\sum_{i=1}^{t} A_i < 2cb\sqrt{\frac{\log t}{t}}\right) + \frac{2}{t^2}. \qquad (14)
\end{aligned}$$

In the last inequality, we used the fact that for sufficiently large t, the term $\log(1/\alpha)/t$ is smaller than $c\sqrt{\log t/t}$, and that $\mathbb{P}(G_t^c) \leq 2/t^2$. By taking the limit in (14), we obtain

$$\mathbb{P}(\gamma = \infty) \leq \limsup_{t\to\infty} \mathbb{P}(\gamma > t) \leq \limsup_{t\to\infty} \mathbb{E}\left[\mathbb{1}_{H_t}\right], \quad \text{where} \quad H_t \coloneqq \left\{\frac{1}{t}\sum_{i=1}^{t} A_i < 2cb\sqrt{\frac{\log t}{t}}\right\}.$$

From the properties of Cesaro means, we know that

$$\liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} A_i \overset{a.s.}{\geq} \liminf_{n\to\infty} A_n,$$

which implies

$$\liminf_{t\to\infty} \frac{\frac{1}{t}\sum_{i=1}^{t} A_i}{2cb\sqrt{\log t/t}} \overset{a.s.}{\geq} \liminf_{t\to\infty} \frac{(A_t/b)}{2c\sqrt{\log t/t}} \overset{a.s.}{>} 1.$$

The last (strict) inequality is due to the assumption on the learning algorithm and noting that $\lim_{t\to\infty}\left(\sqrt{\log t/t}\right) / \left(\sqrt{\log(t-1)/(t-1)}\right) = 1$. This condition implies that $\mathbb{1}_{H_n} \xrightarrow{a.s.} 0$, which by the Bounded convergence theorem (or the continuity of probability) leads to

$$\mathbb{P}(\tau = \infty) \leq \limsup_{t\to\infty} \mathbb{E}[\mathbb{1}_{H_t}] = 0,$$

under the alternative. Hence we have proved the required statement that $\mathbb{P}(\tau < \infty) = 1$ under the alternative.

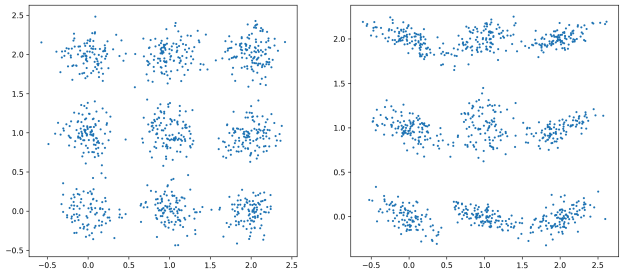

Figure 7: Blob Dataset

11 EXPERIMENTAL DETAILS

We implemented our and the baseline models in pytorch, and the chosen optimizer was Adam. All calculations were performed on a local computing cluster offering 8 TitanX GPU nodes. The average time for each run is 6 min. Thus, the total compute time for the DNN experiments in the main paper is approximately $6 \cdot 33 \cdot 100$ minutes or 330 GPU hours.

In our experiments, we draw mini-batches with consistent sample sizes and then apply Algorithm 1. We trained all models with early stopping coupled with a low patience threshold as a safeguard against potential model overfitting. In this section, we will explain the implementation and training procedure in detail for each experiment: two-sample testing in Section 11.3, adversarial robustness in Section 11.4, group invariance in Section 11.5, CIT in Section 11.6. The hyperparameters are generally selected so the resulting tests have small stopping times. The search space for the learning rate is: $\{0.0005, 0.0001, 0.005, 0.001\}$ and for the patience: $\{2, 5, 10\}$.

11.1 Implementation of DAVT

The training procedure with early stopping always uses the last batch as a validation set. Note that at the beginning of the procedure, the first batches B_1 and B_2 are used for training and validation. We also implement a variation of the loss function that is used for optimization, i.e.

$$\theta_t \in \arg\max_{\theta \in \Theta} \sum_{l=1}^{t} \sum_{Z \in B_l} \log\left(1 + \sigma\left(g_\theta \circ \mathcal{T}_1(Z) - g_\theta \circ \mathcal{T}_2(Z)\right)\right),$$

where $\sigma : \mathbb{R} \to [-1, 1]$ is an monotone increasing function with $\sigma(-x) = -\sigma(x)$.

11.2 Baselines

Here, we provide an overview of the implemented baseline methods.

- E-C2ST (Lhéritier and Cazals, 2018; Pandeva et al., 2022) is a sequential two-sample test based on the M-split likelihood ratio testing where the trained DNN maximizes the data log-likelihood of the previous batches under the alternative.

- Seq-IT (Podkopaev and Ramdas, 2023) extends E-C2ST by formalizing the test in the "testing by betting" framework and thus coupling the payoff to a betting strategy such as ONS. The ONS betting strategy λ_t is computed samplewise in the original paper. To make Seq-IT comparable to our framework, we update the betting strategy batch-wise, resulting in a constant λ_b for the entire batch.

- S-C2ST (Lopez-Paz and Oquab, 2017; Kim et al., 2021) is a non-sequential classifier two-sample test that uses permutation testing for constructing the p-value.

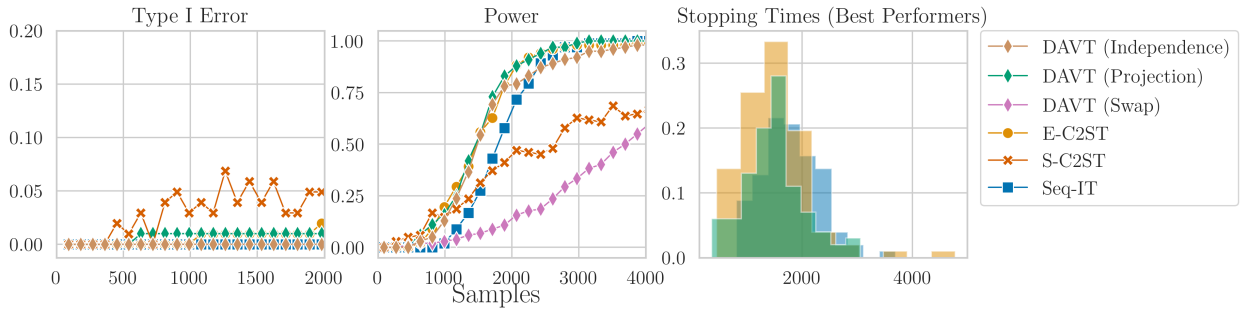

Figure 9: Blob Experiments with lower capacity DNN.

- MMD Test (Gretton et al., 2012) is a kernel-based test utilizing permutation testing. To adapt MMD for testing image data we can use the capabilities of a trained neural network to act as a feature extractor. We then use the extracted features to perform a kernel MMD test.
- ECRT (Shaer et al., 2023) is the only conditional independent test on the list. It is also based on the "testing by betting" paradigm. The difference between DAVT-CIT and ECRT is the presence of a betting strategy linked to the universal portfolio optimization paradigm.

11.3 Two-Sample Testing

The Blob dataset is a synthetic benchmark dataset for two sample tests. It is a challenging dataset due to the overlapping modes of the two classes. Figure 7 provides a visualization of the class distributions.

All trained DNN models follow the network architecture in Table 1. The net contains three linear layers alternating with a LayerNorm and ReLU activation function. All models are trained for a maximum of 500 with early stopping with respect to the loss on the validation set with patience ten and a learning rate of 0.0005. The MMD test bandwidth is set to 0.4.

11.3.1 Type I Error

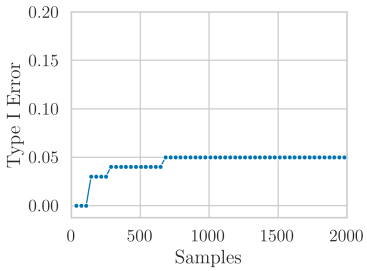

Figure 8: Type I error rate for the Blob dataset.

In this experiment, we demonstrate a scenario where the type I error rate on the two-sample testing with Blob data reaches the chosen significance level of 0.05 (see Figure 8). For this experiment, we fixed the batch size to 36. All other training parameters remain the same as for the experiment showcased in Section 6.

11.3.2 Lower Capacity DNNs

We also performed additional experiments where the chosen DNN has a lower capacity, i.e. the size of the hidden layers is 10 instead of 30 as in Table 1. The results presented in Figure 9 show that the tests' tendencies and ranking in terms of their performance are maintained. Interestingly, the independence test performs better here than in the previous experiments. However, all methods need twice as many samples to achieve maximum power. This illustrates the role of the DNN in the performance of all tests.

11.3.3 Two-sample Test for Dependent X and Y

In the main paper, we compared two different two-sample test operators, and we established that when the samples X and Y are independent, using the swap operator is not very beneficial for

the task. In this experiment, we demonstrate the increasing power of DAVT-Swap as soon as the X and Y become more dependent. We revisit the Blob experiment where we model X_1 and Y_1 (the first dimensions of the dataset) to be dependent and have $\text{corr}(X, Y) = \rho$. We conduct experiments for $\rho = 0.1, 0.2, 1$ and we visualize the results in Figure 10. We can infer that it is easier for DAVT-Swap to reject null the stronger the dependency between X and Y.

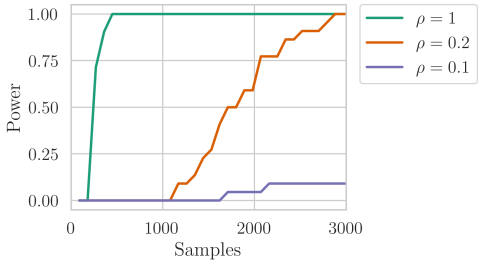

Figure 10: Rejection rate for the Blob two-sample test with swap operator. Here, ρ is the correlation between the first dimensions of X and Y. The stronger the dependence between X and Y the easier it is for DAVT-Swap to reject the null.

11.4 Adversarial Robustness

The output of the trained ResNet50 is an input to a DNN model with the architecture shown in Table 2. It is similar to the one of the Blob data with the difference that the hidden layer size is 32. We trained the models for a maximum of 1000 epochs with a learning rate of 0.0005 with early stopping with patience 5. We normalize the images at the time of loading. The MMD test bandwidth is set to 1.

11.5 Rotated MNIST

The DNN network used here is given in Table 3. The images are normalized before feeding into the network. We traine with l_1 ($\lambda_1 = 0.005$) and l_2 ($\lambda_1 = 0.005$) regularization on the weights. As before, our training procedure is with early stopping with patience 5, learning rate 0.0005, and maximum number of epochs 1000.

11.6 Conditional Independence Testing under Model-X assumption

The trained DNN models follow the network architecture in Table 4. The network includes two linear layers with a dropout(p=0.3) layer and a ReLU activation function between them. All models are trained for up to 500 epochs using early stopping based on validation set loss with a patience of 10 and a learning rate of 0.0005.

Layer (type)	Output Shape
Linear-1	[batch size, 30]
LayerNorm-2	[batch size, 30]
ReLU-3	[batch size, 30]
Linear-4	[batch size, 30]
LayerNorm-5	[batch size, 30]
ReLU-6	[batch size, 30]
Linear-7	[batch size, output size]

Table 1: The network architecture employed in the Blob experiments for all baselines with output size = 2 for S-C2ST and E-C2ST.

Layer (type)	Output Shape
Linear-1	[batch size, 32]
LayerNorm-2	[batch size, 32]
ReLU-3	[batch size, 32]
Linear-4	[batch size, 32]
LayerNorm-5	[batch size, 32]
ReLU-6	[batch size, 32]
Linear-7	[batch size, output size]

Table 2: The network architecture employed in the CIFAR-10 experiments for all baselines with output size = 2 for S-C2ST and E-C2ST.

Layer (type)	Output Shape
Linear-1	[batch size, 128]
ReLU-2	[batch size, 128]
Dropout(p=0.5)-3	[batch size, 128]
Linear-4	[batch size, 64]
ReLU-5	[batch size, 64]
Dropout(p=0.5)-6	[batch size, 64]
Linear-7	[batch size, output size]

Table 3: The network architecture employed in the MNIST experiments.

Layer (type)	Output Shape
Linear-1	[batch size, 128]
ReLU-2	[batch size, 128]
Dropout(p=0.3)-3	[batch size, 128]
Linear-4	[batch size, output size]

Table 4: The network architecture employed in the CIT experiments.

12 PRACTICAL CONSIDERATIONS

The model training and architecture. Different architectures, from convolutional to recurrent models, can affect the performance of all data-driven tests, a discussion alluded to in Section 11.3.2. Therefore, appropriate DNN selection becomes critical to ensure the reliability and robustness of conclusions derived from these tests. While all the considered sequential data-driven statistical tests have a finite-sample type I error control, unlike most classical testing procedures, researchers must be familiar with the specific requirements of their statistical tests in order to select a DNN that maximizes power for a given sample size.

There is another dimension of DNN training worth exploring. Unlike traditional batch mode training, online training is designed for continuous data streams, allowing DNNs to adapt and evolve in real-time. This procedure allows to disregard previous data batches and to update the DNN only from a single batch. This can make sequential testing methods even more appealing for a wide range of applications.

Alternative computation of the betting score. In Section 4, we defined the betting score S_t in (5) by taking the product of $(1 + \widetilde{g}_{\theta_{t-1}}(Z))$ over all Z in the mini-batch B_t. An alternative, and equally valid, way of defining the wealth process is by taking the average; that is,

$$S_t = \frac{1}{b}\left(\sum_{Z \in B_t} 1 + \widetilde{g}_{\theta_{t-1}}(Z)\right) = 1 + \frac{1}{b}\sum_{Z \in B_t} \widetilde{g}_{\theta_{t-1}}(Z). \tag{15}$$

The resulting wealth process has a smaller variance than our proposal, but this comes at the cost of power for very large mini-batches. This suggests that the optimal betting score construction should lie somewhere in between these two extremes to achieve the best bias-variance trade-off. For example, at test time, a mini-batch can be partitioned into sufficiently small ones for which we compute the increments using the new proposed updating scheme. A thorough exploration of the design of optimal betting scores is an interesting question for future work.

Unpaired Data. In the main paper, we mostly presented scenarios where the samples of X_t and Y_t are observed simultaneously. However, our framework is not only limited to paired data. It can also be applied in a more general setting when this assumption does not hold.

This scenario has been discussed in Shekhar and Ramdas (2023). There, the authors propose to use betting scores that align with the proposal in the previous point. More precisely, consider a two-sample test for batches $B_t = \{X_{tb+j}\}_{j=1}^{b_{t1}} \cup \{Y_{tb+j}\}_{j=1}^{b_{t2}}$ consisting of $b_{t1} + b_{t2}$ observations of X and Y. Then, we can define the increments as

$$S_t = 1 + \frac{1}{b_{t1}}\sum_{j=1}^{b_{t1}} g_{\theta_{t-1}}(X_{tb+j}) - \frac{1}{b_{t2}}\sum_{j=1}^{b_{t2}} g_{\theta_{t-1}}(Y_{tb+j})$$

or

$$S_t = 1 + \sigma(\sum_{j=1}^{b_{t1}} g_{\theta_{t-1}}(X_{tb+j}) - \sum_{j=1}^{b_{t2}} g_{\theta_{t-1}}(Y_{tb+j}))$$

where $\sigma : \mathbb{R} \to [-1, 1]$ is an monotone increasing function with $\sigma(-x) = -\sigma(x)$. While this does not quite match our framework, it is a way to model unpaired data.

We can fit this problem into our formalism by considering random variable $L \in \{-1, 1\}$ and W such that $P(W|L = 1) = P(X)$ and $P(W|L = -1) = P(Y)$. We can test whether W and L are independent instead of considering the classical two-sample test. If the samples X_t and Y_t are somewhat dependent, we can apply averaging to obtain the increments (see (15)); otherwise, we can stick to our proposal.