Combinatorial Testing Metrics for Machine Learning

Erin Lanus*, Laura J. Freeman*, D. Richard Kuhn†, Raghu N. Kacker†
*Hume Center for National Security and Technology, Virginia Tech, Arlington, VA, USA {lanus, laura.freeman}@vt.edu
†National Institute of Standards and Technology, Gaithersburg, MD, USA {kuhn, raghu.kacker}@nist.gov

***Abstract*—This paper defines a set difference metric for comparing machine learning (ML) datasets and proposes the difference between datasets be a function of combinatorial coverage. We illustrate its utility for evaluating and predicting performance of ML models. Identifying and measuring differences between datasets is of significant value for ML problems, where the accuracy of the model is heavily dependent on the degree to which training data are sufficiently representative of data encountered in application. The method is illustrated for transfer learning without retraining, the problem of predicting performance of a model trained on one dataset and applied to another.**

***Index Terms*—combinatorial testing, machine learning, operating envelopes, transfer learning, test set selection**

I. Introduction

In software and hardware, component systems are often well designed and tested, but failures occur during integration due to unexpected interactions between components. A study [1] of empirical data found that nearly all failures in software are caused by a limited number of interacting components and concluded that testing interactions of between four and six components could detect all failures in the software systems considered. This result has led to broader adoption of combinatorial testing, because it showed that strong assurance could be achieved without exhaustive testing of software and hardware systems [2].

Conducting testing of systems with embedded machine learning (ML) using conventional software approaches poses challenges due to characteristics of ML such as the large input space, effort required for white box testing, and emergent behaviors apparent only at integration or system levels [3], [4]. CT is a black box approach to testing an integrated system using a pseudo-exhaustive strategy for large input spaces. Thus far, CT has been applied to test case generation for autonomous vehicle systems with embedded ML components [5], testing the internal state space of a neural network [6], feature selection [7], and explainable ML [8].

An ML model is trained on *examples* consisting of *values* assigned to *features* and possibly a *label*, such as the membership class for the example. The data is fundamental to ML model performance. In this paper, we leverage CT for testing ML systems through comparison of datasets to consider how differences between members of two classes lead to classification decisions and if differences between datasets are useful for predicting whether a model trained on one dataset will perform as expected on another. Comparing datasets via combinations of features is possible at three levels of granularity: 1) the count of combinations that are present or absent, 2) which specific combinations are present or absent, and 3) the distribution of combinations.

In this work, we define a new combinatorial coverage metric for comparing ML datasets in § II focusing on the first level of granularity (presence/absence of combinations). We highlight two distinct areas of applications of the metric in § III. The metric's utility based on interpretable features in the data is demonstrated for fault localization and explainable classification. The use of the metric to define a model's operating envelope extends to applications in transfer learning, selection of training and test datasets, and directing data collection and labeling efforts. We discuss problems for future work in § IV.

II. Metrics

We treat features, including the label when available, as *factors* for CT. Continuous-valued factors must be discretized prior to applying CT so that each factor has a corresponding finite set of values. A t-way *value combination* is an assignment of specific values to t of the factors, or a t-tuple of (factor, value) pairs. If there are k factors, each example then contains $\binom{k}{t}$ factor combinations with one value combination each.

Combinatorial coverage, also called total t-way coverage, is a metric from the CT literature [9] to describe the proportion of valid t-way value combinations appearing in a set (Fig. 1). Value combinations that appear in the set are *covered* by the set. Define a universe with k factors and their respective values so that $\mathcal{U}$ is the set of all valid examples, and let $\mathcal{U}_t$ be the set of valid t-way value combinations. If some value combination is invalid, it is a *constraint* and can be removed from $\mathcal{U}_t$. Given a dataset $\mathcal{D} \subseteq \mathcal{U}$, define $\mathcal{D}_t$ as the set of t-way value combinations appearing in $\mathcal{D}$. (We acknowledge a slight abuse of notation as $\mathcal{D}$ may be a multiset. This does not impact the metrics.) Denote set cardinality by $|\mathcal{D}_t|$. The t-way combinatorial coverage [9] of $\mathcal{D}$ is

$$CC_t(\mathcal{D}) = \frac{|\mathcal{D}_t|}{|\mathcal{U}_t|}.$$

Let $\mathcal{S}$ and $\mathcal{T}$ be datasets and define $\mathcal{S}_t, \mathcal{T}_t$ as the set of t-way value combinations appearing in $\mathcal{S}, \mathcal{T}$, respectively. The set difference $\mathcal{T}_t \backslash \mathcal{S}_t$ is the set of value combinations appearing

Research of EL and LJF funded in part by MITRE University Innovation Exchange program.

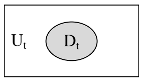

Fig. 1. Venn diagram showing $CC_t(\mathcal{D})$ as the coverage of $\mathcal{U}_t$ by $\mathcal{D}_t$.

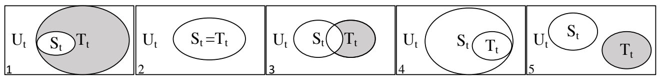

Fig. 2. Venn diagrams of the set theoretic relationships between $\mathcal{S}$ and $\mathcal{T}$.

in $\mathcal{T}_t$ but not in $\mathcal{S}_t$. We define the t-way set difference combinatorial coverage

$$SDCC_t(\mathcal{T} \setminus \mathcal{S}) = \frac{|\mathcal{T}_t \setminus \mathcal{S}_t|}{|\mathcal{T}_t|}$$

as the proportion of t-way value combinations appearing in $\mathcal{T}$ but not $\mathcal{S}$. Constraints need not be explicitly defined as only value combinations present in $\mathcal{T}$ are considered. $SDCC_t$ is a score between 0 and 1 inclusive. The set theoretic relationships (Fig. 2) and corresponding ranges of $SDCC_t$ are:

1) $\mathcal{S}_t \subset \mathcal{T}_t \implies 0 < SDCC_t(\mathcal{T} \setminus \mathcal{S}) < 1$,
2) $\mathcal{S}_t = \mathcal{T}_t \implies SDCC_t(\mathcal{T} \setminus \mathcal{S}) = 0$,
3) $(\mathcal{S}_t \not\subset \mathcal{T}_t) \wedge (\mathcal{T}_t \not\subset \mathcal{S}_t) \wedge (\mathcal{S}_t \bigcap \mathcal{T}_t \neq \emptyset) \implies 0 < SDCC_t(\mathcal{T} \setminus \mathcal{S}) < 1$,
4) $\mathcal{T}_t \subset \mathcal{S}_t \implies SDCC_t(\mathcal{T} \setminus \mathcal{S}) = 0$,
5) $\mathcal{S}_t \bigcap \mathcal{T}_t = \emptyset \implies SDCC_t(\mathcal{T} \setminus \mathcal{S}) = 1$.

Set difference combinatorial coverage is directed; $SDCC_t(\mathcal{T} \setminus \mathcal{S})$ may not be equal to $SDCC_t(\mathcal{S} \setminus \mathcal{T})$. As a difference metric, higher values correspond to a larger difference between the first and second sets.

At the coarsest level of granularity, coverage is represented as a single score or Venn diagram. To provide more information, the value combinations not appearing in $\mathcal{D}_t$ for CC_t and value combinations in the set difference $\mathcal{T}_t \setminus \mathcal{S}_t$ for $SDCC_t$ are listed or plotted as status per value combination. A heatmap of value combination frequency for CC_t and difference in relative frequency for $SDCC_t$ provides the finest granularity.

III. Applications

A. Fault localization

Set differencing of t-way value combinations has been applied to the problem of fault localization. A variety of set theoretic operations can be used in reducing the set of possible failure-triggering value combinations in deterministic software [10]. Running a test set typically results in a large number of passing tests and a small number of failing tests, but only a small subset of value combinations in the failing tests will induce a failure. For P_t = value combinations in passing tests and F_t = value combinations in failing tests and C_t = fault-triggering value combinations, the first step in identifying failure-triggering value combinations is a basic elimination rule: compute $F_t \setminus P_t$, value combinations in failing tests that are not in any passing tests, which for deterministic systems must contain the fault-triggering value combinations C_t. Basic set operations can also be used to further reduce the possible value combinations involved in a failure. For example, a value combination continuity rule says that if a particular t-way value combination in F_t is included in all higher strength value combinations that contain the same t factors, then the t-way value combination is sufficient to detect the error.

B. Explainable Classification

From a certain perspective, the problem of classification in ML is essentially the same as the fault localization problem in CT. We seek to identify a small subset of factors that distinguish the class from examples not in the class. This process could be viewed as generalizing the fault localization problem, where the failing tests are the class and passing tests are non-class members – what value combinations of factor values are unique to the failing tests?

This simple observation leads to a method of producing explanations or justifications of ML classifications [11] to achieve explainable AI (XAI), by computing $C_t \setminus N_t$, the set of t-way value combinations that appear in members of the class C which are not in the non-class members of N, or are more strongly associated with C than N. For example, applying this method in a database of animal characteristics produces seven predicates that are unique to reptiles (within this database): *not aquatic AND not toothed AND four legs*, *egg-laying AND not aquatic AND four legs*, etc. These value combinations have an obvious mapping with simple rules: “if non aquatic AND not toothed AND … ”. No single-factor or 2-way value combinations are uniquely associated with the reptile class, but including 3-way value combinations makes it possible to identify class members.

Previous model induction methods have been developed to reverse engineer an explanation or model from ML output [12], [13], using statistical methods to identify characteristics most closely associated with a class. The combinatorial XAI method extends this approach by producing combinations of characteristics for explanation. This distinction is important because closely associated single factors are not necessarily contained in identifying value combinations. Rule-based expert systems are often considered easy to explain but generally are not as proficient as more opaque methods such as neural networks [14]. The combinatorial approach to XAI provides a natural mapping to clearly understandable diagnostic rules.

C. Model Operating Envelope

Computer vision includes tasks such as detecting or classifying an *object* in an image. The complexity of the domain – all of the variables affecting the production of an image – leads to high likelihood of interaction effects. Consider the problem of detecting a white truck in an image. A white truck against a light background at noon from an overhead view likely presents a more difficult detection scenario than a white truck against the same light background in late afternoon where shadows are present or from profile such that the horizon line breaks up the background. The operating envelope of an ML model describes the *contexts* in which it is expected to perform correctly; deploying to contexts outside

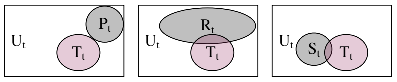

Fig. 3. Set differences used to select a source for a target from a model zoo.

of the envelope can lead to unexpected outcomes. An ML model learns about examples on which it trains, so to perform as expected in each of these contexts, it is anticipated that "enough" representative examples must be included in the training dataset. The challenge is how to define contexts and measure representativeness of the training examples.

One dimension of the operating envelope of a computer vision model is defined by describing the contexts in which the model trained as coverage of value combinations among features present in the dataset. These features may be derived directly from the image data, but there are two benefits of using metadata such as "Time of Day" or "Location" collected along with the image acting as a surrogate for contexts present in the image. Metadata are more understandable by human operators; "Time of Day" as a surrogate for lighting effects in the image is more interpretable than presenting values for luminance and contrast. Metadata may be available when image data is not, such as the case when an event is occurring in the near future in a new deployment environment for which no images have been collected. Expected factors such as "Time of Day" and "Location" can be extracted from the event profile.

When class labels are available, we describe a special way of calculating value combinations. *Label centrism* forces all value combinations to include a label; a label-centric value combination includes the label and $t-1$ of the other features. Label centrism describes the contexts in which objects appear.

Claims of representativeness by a training dataset often rely on randomized selection or counts by object type, but may fail to be representative of larger contexts in the deployment environment. Combinatorial coverage (CC) computed on a training dataset provides a measurement of the contexts on which the model trained via value combinations given the tunable parameter t. In the case of transfer learning, a model trained in one environment is deployed to a new environment, possibly without retraining or fine tuning. Where CC is a measure of coverage by a dataset with respect to some defined universe, the new metric, $SDCC$, describes a directed difference between two datasets and is useful for measuring the distance between a source dataset $\mathcal{S}$ where the model is trained and a target dataset $\mathcal{T}$ where the model is deployed. When multiple source models are available in a *model zoo*, the source dataset $\mathcal{S}$ with the smallest $SDCC_t(\mathcal{T} \setminus \mathcal{S})$ provides the best coverage of contexts in the target by the source (Fig. 3). Additionally, as value combinations in a set difference describe contexts unseen by the trained model, the list of value combinations in the set difference provides a mechanism for directing data collection or labeling efforts to include examples containing these value combinations.

A use case for the set difference application to operating envelopes for transfer learning is demonstrated on the "Planes in Satellite Imagery" Kaggle dataset [15]. The dataset is intended for binary classification and is comprised of images that either have a plane or do not have a plane along with metadata indicating the location as Northern California or Southern California. If a model is trained on the Southern subset of data $\mathcal{S}$, a performance drop occurs when used to make predictions on the Northern subset of data $\mathcal{T}$, indicating a transfer learning problem. The drop is not noted when the direction of transfer is reversed. We apply our metrics to highlight differences between the datasets that might be responsible. Twelve features are derived from the image data (the mean and variance each for the red, green, blue, hue, saturation, and luminance) and values for each feature are discretized by forming three bins encompassing equal-sized ranges. Value combinations are label-centric and $t = 2$. The Southern set contains 21,151 images and the Northern set contains 10,849 images. The $CC_2(\mathcal{S}) = \frac{60}{72} = 0.83$ and $CC_2(\mathcal{T}) = \frac{67}{72} = 0.93$, meaning that the Northern set covers more of the universe than the Southern set despite having half as many images. Fig. 4 plots the coverage of value combinations in the sets side by side.

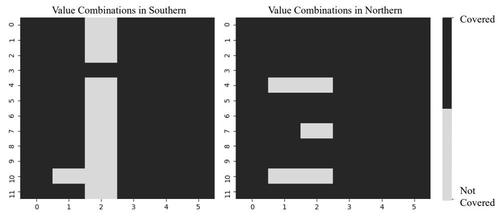

Fig. 4. Coverage of 2-way label-centric value combinations indexed by combination (y-axis) and value combination within a combination (x-axis).

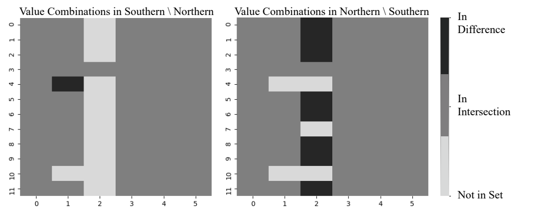

Fig. 5. Set differences of 2-way label-centric value combinations shaded by set membership; $\mathcal{T} \setminus \mathcal{S}$ is dark, $\mathcal{T} \bigcap \mathcal{S}$ is medium, and $\neg T$ is light.

The utility of CC for comparing a source and target pair is limited. Suppose $\mathcal{S}'_t$ contains all value combinations in the left half of a given plot and none in the right half, while $\mathcal{T}'_t = \mathcal{U}_t \setminus \mathcal{S}'_t$ contains the complement. Both have CC_2 values of 0.5. Suppose $\mathcal{S}'' = \mathcal{T}''$ yet $CC_2(\mathcal{S}'') = 0.25$. The relationship between the respective sets is not apparent via CC, which is the limitation for which $SDCC$ is designed. For the Planesnet datasets, $SDCC_2(\mathcal{S} \setminus \mathcal{T}) = \frac{1}{60} = 0.02$ and $SDCC_2(\mathcal{T} \setminus \mathcal{S}) = \frac{8}{67} = 0.12$ (Fig. 5). For this dataset, $SDCC_2$ is correlated with a drop in performance in transfer learning without retraining.

Combinatorial coverage is useful in testing for deterministic failures in software systems where the appearance of a value combination among components in one test is sufficient to cause a failure; if the components will interact to cause a

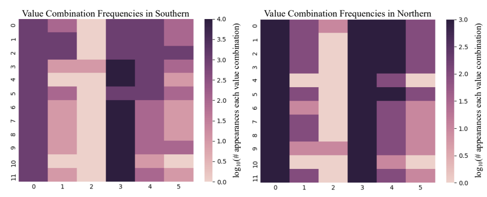

Fig. 6. Frequencies of value combinations provide distributional information.

failure, this is detected by a test suite containing that value combination at least once. Statistical learning does not have this property. We propose the CC and $SDCC$ metrics as tools to identify contexts in the target environment that are not likely to be within the model's operating envelope. However, as models are trained by updating weights each time these contexts are seen, we suspect that distribution of coverage would improve the operating envelope description. Frequently appearing value combinations indicate contexts on which the model was well trained; they could also indicate instances of overfitting. Infrequently appearing value combinations indicate contexts on which the model trained less; they could present contexts in which the model has difficulty making classifications. Our work measures and plots this distribution (Fig. 6).

D. Test Set Design

Datasets are partitioned into training $\mathcal{S}$, validation, and testing $\mathcal{T}$ sets. When datasets are large and random selection is applied, the hope is that the test set is representative of the training set as they are drawn from the same population. Computing $SDCC_t(\mathcal{S} \setminus \mathcal{T})$ and $SDCC_t(\mathcal{T} \setminus \mathcal{S})$ provides assurance against a bad random draw. A simple randomized algorithm makes several random partitions and keeps the one with the lowest $SDCC$ values. This is equivalent to testing within the operating envelope of the model.

Another testing strategy is to identify where the model fails to generalize to new contexts it has not trained by selecting test sets outside of the envelope. In this case, selecting $\mathcal{T}$ so that $SDCC_t(\mathcal{T}\setminus\mathcal{S})$ is close to 1 creates a test set containing many untrained contexts. The importance of the reverse direction for this strategy is not as clear. When $SDCC_t(\mathcal{T}\setminus\mathcal{S}) = 1$, the sets T_t and S_t are disjoint and $SDCC_t(\mathcal{S} \setminus \mathcal{T}) = 1$ necessarily. When $SDCC_t(\mathcal{S} \setminus \mathcal{T}) < 1$, the score depends on $|S_t|$.

IV. Conclusions and Future Work

This work discusses metrics that provide tools for explaining classification outcomes and defining the domain over which an ML model is expected to operate successfully. Future work is needed to test the hypothesis that models trained on source sets with smaller $SDCC_t$ distances to the target perform better in the target environment, as well as explore the usefulness of these metrics across multiple ML domains, the impact of label centrism, and choosing a "good" value combination size t.

Additionally, the sensitivity of these metrics to feature or metadata selection is critical. In the classification application, the features were directly explainable. In the computer vision application, the research had to first hypothesize reasonable features. The process of hypothesizing features, conducting initial screening experiments to select the meaningful features, and confirming results should be codified to ensure that this work is not subject to confirmation biases of the research team or over interpretation of correlations as explanatory variables.

Finally, additional work is needed to exploit the deeper levels of explainability, that is, which specific value combinations are present or absent and the distribution of those value combinations. The specific value combinations present or absent should be explored for potential explanation of how and why models perform well or poorly, potential biases introduced into the models, and predictive capabilities to new operating envelopes. Set difference frequency metrics should be developed and their application to transferability evaluated.

Acknowledgment: We thank Stephen Adams, Sayyed Ahamed, Peter Beling, Tyler Cody, Adam Edwards, and Sachin Shetty for the motivating ML work on the Planesnet dataset. We thank the reviewers for their helpful comments.

References

[1] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE Transactions on Software Engineering*, vol. 30, no. 6, pp. 418–421, 2004.

[2] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys (CSUR)*, vol. 43, no. 2, pp. 1–29, 2011.

[3] D. Marijan, A. Gotlieb, and M. Kumar Ahuja, "Challenges of testing machine learning based systems," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, 2019, pp. 101–102.

[4] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, pp. 1–36, 2020.

[5] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1555–1562.

[6] L. Ma, F. Juefei-Xu, M. Xue, B. Li, L. Li, Y. Liu, and J. Zhao, "Deepct: Tomographic combinatorial testing for deep learning systems," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2019, pp. 614–618.

[7] S. Vilkomir, J. Wang, N. L. Thai, and J. Ding, "Combinatorial methods of feature selection for cell image classification," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2017, pp. 55–60.

[8] R. Kuhn and R. Kacker, "An application of combinatorial methods for explainability in artificial intelligence and machine learning (draft)," National Institute of Standards and Technology, Tech. Rep., 2019.

[9] D. R. Kuhn, I. D. Mendoza, R. N. Kacker, and Y. Lei, "Combinatorial coverage measurement concepts and applications," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, 2013, pp. 352–361.

[10] D. R. Kuhn, R. N. Kacker, and Y. Lei, "Practical combinatorial testing," *NIST special Publication*, vol. 800, no. 142, p. 142, 2010.

[11] D. R. Kuhn, R. N. Kacker, Y. Lei, and D. E. Simos, "Combinatorial methods for explainable AI."

[12] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1135–1144.

[13] F. Shakerin and G. Gupta, "Induction of non-monotonic logic programs to explain boosted tree models using lime," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3052–3059.

[14] D. Gunning, "Explainable artificial intelligence (XAI)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, no. 2, 2017.

[15] Rhammell, "Planes in satellite imagery," Jan 2018. [Online]. Available: https://www.kaggle.com/rhammell/planesnet