Feeling Validated: Constructing Validation Sets for Few-Shot Learning

Anonymous EMNLP submission

Abstract

We study validation set construction via data augmentation in *true* few-shot text classification. Empirically, we show that task-agnostic methods—known to be ineffective for improving test set accuracy for state-of-the-art models when used to augment the training set—are effective for model selection when used to build validation sets. However, test set accuracy on validation sets synthesized via these techniques does not provide a good estimate of test set accuracy. To support better estimates, we propose DAUGSS, a generative method for domain-specific data augmentation that is trained *once* on task-agnostic data and then employed for augmentation on *any* data set, by using provided training examples and a set of guide words as a prompt. In experiments with 6 data sets, both 5 and 10 examples per class, training the last layer weights and full fine-tuning, and the choice of 4 continuous-valued hyperparameters, DAUGSS is better than or competitive with other methods of validation set construction, while also facilitating better estimates of test set accuracy.

1 Introduction

Few-shot learning, i.e., learning with only a handful of training examples, is growing area of machine learning and the subject of significant study. Few-shot learning problems manifest in many practical scenarios, including model adaptation—especially when a single model is being fine-tuned for a number of new domains. Given the typical reliance of state-of-the-art models on large datasets and the high cost of collecting new data, techniques that enable effective few-shot learning are especially valuable. Recent work in few-shot learning employ a variety of techniques include: data augmentation and meta-learning.

While reported few-shot learning results are positive, recent work argues that, by and large, these results were achieved in unrealistic experimental settings (Perez et al., 2021). In particular, most evaluations of few-shot learning systems leverage a validation set (i.e., held-out examples) for model selection. But, in practice, validation data is unavailable in few-shot learning, and thus experiments used to study few-shot learning methods should assume no access to additional examples for validation (Bragg et al., 2021). Perez et al. go on to explore the so-called *true* few-shot setting, in which there are no dedicated validation examples. Their experiments reveal that model selection via either cross-validation or minimum description length—two classic model selection methods that require no dedicated validation set—yield models that perform much worse than those selected using a validation set.

Given the importance of validation data for model selection in few-shot learning, we propose *validation set creation* via *data augmentation*. Data augmentation describes a family of techniques for constructing examples for a wide variety of learning problems. Typically, these methods are invoked to synthesize specific types of examples that are underrepresented in the training set or, to expand a small training set with a wide variety of examples, such as in few-shot learning. Unlike previous work, we employ data augmentation to synthesize examples for model selection. We find that in true few-shot classification (i.e., no dedicated validation set), data augmentation methods are better used for creating validation examples than expanding training sets; even for methods known to provide negligible performance gains (when used for training set expansion) (Longpre et al., 2020). Our experiments also reveal that model selection with synthetic data yields better models than both selection with cross-validation, or with data held out from the (small) few-shot training set.

While synthesized examples can be used effectively for model selection, we find that performance on these examples is not indicative of test set per-

formance. To remedy this, we design DAUGSS, a new generative method for data augmentation designed for use in few-shot classification settings. In DAUGSS, we train a sequence to sequence generation model using generic, publicly available data. After training, the generation model is prompted with available task-specific data in order to generate examples in-domain examples. Importantly, the generation model also takes a set of *guide words* as input, which provide some control over the model's output. Empirically, we show that DAUGSS is best among the data augmentation methods tested in terms of model selection. Moreover, we find that the selected model's performance on the synthetic data provides the most accurate estimate of test set performance, among all methods.

2 Intent Classification

In this work, we study few-shot *intent classification*. Intent classification is the task of classifying natural language *utterances* into *intents*. Typically, utterances are short, ranging from a single word to a few sentences. The number of intents varies by domain: problem instances studied in this work range from having 7 to 150 intents.

In practice, intent classification systems are a basic building block of virtual assistants (Coucke et al., 2018) where few-shot learning ability is critical. Virtual assistants are deployed in a wide variety of domains (e.g., medicine, travel, etc.), and as such, they are often pre-trained by a developer using a large amount of data, and then fine-tuned for each specialized usage. Since each usage requires its own fine-tuning data (and data collection is typically not scalable), virtual assistants must perform well when the number of fine-tuning examples is small. Furthermore, ed virtual assistant must adapt to new domains over time, i.e., handle an ever-growing number of intents, adaptation with only a few additional examples is necessary (Kumar et al., 2019).

3 DAUGSS

Training a state of the art model typically requires a large number of examples for training and validation. In the few-shot setting, i.e., when only a handful of examples are available, one approach is to *generate* more data. However, generating additional data can be challenging, especially when the task at hand lies in a highly specialized domain, and for which no existing data generators exist.

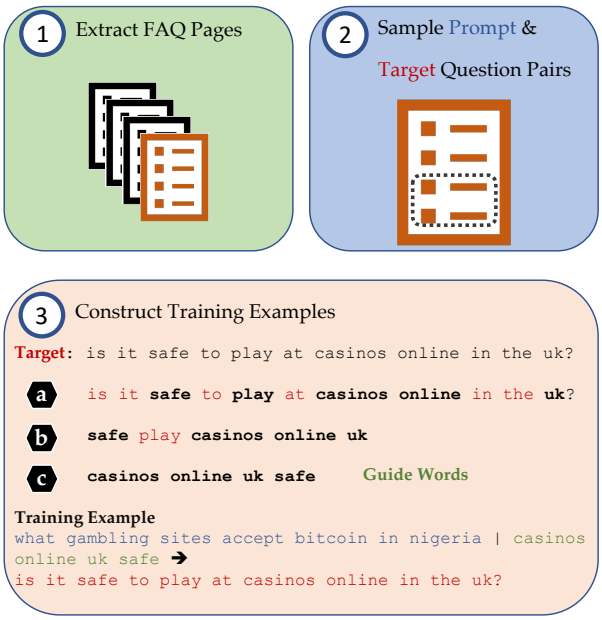

Figure 1: **DAUGSS Training Set Construction.** 1) FAQ pages are extrated from Common Crawl. 2) pairs of questions that appear within the same FAQ are sampled; each pair contains a *prompt* and a *target*. 3) *guide words* are subsampled from the target in three steps: a. all stopwords are removed, b. a small fraction of the remaining tokens are removed, and c. the remaining tokens are shuffled. The guide words are prepended with the prompt followed by a pipe character ("|") and map to the target, thereby constituting a sequence to sequence training example for the generator.

Such cases require a domain specific generator, but training such a model is complicated by the scant in-domain data. It is possible to use a task-agnostic method, like EDA (Wei and Zou, 2019), to construct new examples by perturb existing data, but these methods have been shown to be ineffective when used alongside state-of-the-art transformer models (Longpre et al., 2020).

In this section, we describe DAUGSS, an algorithm for training a generative model for text, intended for us in few-shot, domain-specific settings. Since we assume a very limited amount of available in-domain data, we train the generator on unlabeled, out-of-domain data from Common Crawl. After the generator is trained (on out-of-domain), we use the available in-domain data to prompt the model to generate in-domain data. We begin with a discussion of the generator, its inputs and its outputs. Then, we describe how the generator is trained.

3.1 The Generator

At a high-level, a DAUGSS-trained generator is a model that takes 2 strings and generates a string. The first input—which we call the *prompt*—p, and is a clause that embodies the style and semantic content that the model should generate. The second input is a variable length sequence of *guide words*, w (Pascual et al., 2021). The guide words are tokens that the model is trained to include in the output, and thus provide additional control over the generation. While the guide words appear in the input and output of all of the generator's training examples, unlike previous work, the model's output need not include all guide words provided as input.

In more detail, consider a few-shot, k-way, text classification data set $\mathcal{X} = \{(x_i, y_i)\}_{i=0}^{N}$, where $y \in \{c_0, c_1, \ldots, c_k\}$. Let $X[c] = \{x_j : (x_j, y_j) \in \mathcal{X}, y_j = c\}$ be the subset of utterances in $\mathcal{X}$ of class c. Finally, let g be a DAUGSS-trained generator, $g : p \times \mathbf{w} \rightarrow z$. To generate a new example of a class, c, we must we must choose a prompt, p, and guide words, $\mathbf{w}$. We choose a prompt uniformly at random among utterances of class c, i.e., $p \sim \mathcal{U}(X[c])$. Next, we select guide words. To do so, we begin by building a *per-class token distribution*. That is, for each utterance in $X[c]$, we filter all stop words with `spaCy` (Honnibal et al., 2020), and compute the empirical distribution of the remaining tokens. To sample guide words for a class c, we first sample a length L from the empirical distribution of lengths of examples in $X[c]$, and then sample L guide words independently from the per-class token distribution for c.

Our use of an utterance to prompt the generator is inspired by work on Example Extrapolation (EX2) (Lee et al., 2021). Whereas their work focuses on problems with uneven amounts of data per class, we focus on few-shot learning—where EX2 is inappropriate. Unlike their work, we only provide the generator with a single utterance, but we also provide a sequence of guide words. The guide words yield some control over the generation process, including control in making the outputs diverse. This is especially important when data is scarce for all classes.

3.2 Training

In the DAUGSS algorithm, the generator, g, is trained from a set of triples $\mathcal{Q} = \{(p_i, \mathbf{w_i}, z_i)\}_{i=1}^{M}$, where g must generate z_i from inputs p_i and $\mathbf{w_i}$. Given the assumption of operating in a few-shot setting, DAUGSS is self-supervised and does not require *any in-domain data*. In other words, the generator's training data, $\mathcal{Q}$, does not include any examples in $\mathcal{X}$. Instead, examples in $\mathcal{Q}$ are constructed from a public data source, such as Wikipedia or Common Crawl. A primary benefit of such a generator is that it can be trained *once*, and then employed to generate data for *any number of tasks*.

For any training example, $(p, \mathbf{w}, z)$, the prompt, p, and output, z, should be stylistically and semantically related. This is so that the generator, g, learns to generate in-domain data when the prompt, p, is an in-domain example. Moreover, the guide words, $\mathbf{w}$, should appear in z, since the guide words are intended to be used to control the generated out. In order to accomplish this, we assume access to a (rudimentary) similarity function. Formally, let $\mathcal{J}$ be a collection of utterances (e.g., sentences in Wikipedia) and let $s : J \times J \rightarrow [0, 1]$ be a binary function that returns 1 if its inputs are similar. An example of s is a function that returns 1 when two utterances appear on the same webpage. To construct an example (p, w, z), we select two similar utterances (with respect to s). The first we set to be p; the second, z. The guide words, $\mathbf{w}$, are (a subset of) the non-stopword in z.

In our work, examples in $\mathcal{Q}$ are constructed from a subset of Common Crawl that includes frequently asked questions (FAQ) pages. We set s to be the function that returns 1 if two utterances appear on the same web page (i.e.,in the same FAQ). To construct training examples, we randomly select two questions from the same page to serve as the prompt, p, and output, z, respectively. We only utilize questions (and not answers) because the questions share some stylistic characteristics with typical utterances in intent classification (e.g., both are typically short questions or instructions). The guide words, w, are a randomly selected 95% of the non-stop word tokens in z[1]. We use 95% of the non-stopwords (instead of all non-stopwords) so that the model does not learn that *all* the guide words represent all non-stopwords in the desired output. In our work, g is parameterized by T5 (Raffel et al., 2020), a large-scale, sequence to sequence model, and as such, the inputs p_i and $\mathbf{w_i}$ and concatenated, but delimited by a " | ". See Figure 1 for an example training instance in $\mathcal{Q}$.

[1] w is ordered arbitrarily.

FAQ Extraction from Common Crawl: we extract FAQs from Common Crawl—a large-scale archive of crawled webpages. We use a single dump from January 2021, which includes 3.4 billion webpages. To detect QA contents nested in raw webpages, we leverage structured markup for QA[2] and FAQ[3] pages. This markup is widely used, and facilitates the display of Q & A result previews along with search results (e.g., google search).

Naive search in billions of webpages in costly. Therefore, to extract FAQs, we first perform a fast regex-based search that yields approximately 2.3 million matching pages. After parsing the resulting (HTML) pages, we are able to extract approximately 10 million QA/FAQ data snippets. After post-processing the snippets, (e.g., removing badly formatted snippets where question/answer cannot be automatically recovered; removing empty question/answer bodies), and perform language detection to identify English QA pairs. We group the English questions by page and randomly select 200k question pairs for training such that both questions appeared on the same page.

4 Experiments

In this section, we present an experimental study of model selection methods in few-shot text classification. We report test set accuracy achieved by models selected using various methods. For methods that utilize validation examples, we also measure the error incurred by employing validation accuracy as an estimate of test accuracy. Finally, for data created via augmentation methods, we evaluate whether that data is best used when added to the training set, the validation set, or both.

Datasets: experiments are performed with the following datasets: **clinc**, **bank**, **snips**, **curekart**, **powerplay**, and **mattress** (Larson et al., 2019; Casanueva et al., 2020; Coucke et al., 2018; Arora et al., 2020). To resemble the few-shot setting, we subsample each dataset to a specific number of examples per class. When referring to a dataset, we use the suffix -k (e.g., **clinc-**k) to indicate that the dataset has been subsampled to k examples per class[4]. Following previous work, we omit out-of-scope utterances (included in **clinc**, **curekart**, **powerplay**, and **mattress**).

4.1 Model Selection

We study *true* few-shot text classification, i.e., few-shot learning in which no validation data is provided. Given the importance of selecting suitable hyperparameters for state-of-the-art models, we experiment with the following approaches for *constructing* a validation set:

- **HOLDOUT** - 20% of the training data (per class) is held out and used for validation. This resembles a typical workflow for non-few-shot settings.
- **TRAIN** - use the training set as the validation set; overfitting is expected.
- **DAUGSS** - generate the validation set, with 20 examples per class, using a generator trained by DAUGSS[5].
- **EDA** - similar to the previous approach but use task-agnostic data augmentation for generation (Wei and Zou, 2019). This style of augmentation is known to yield little improvements when used for training set augmentation for state-of-the-art models (Longpre et al., 2020).
- **TEST** - use the test set as the validation set; a highly competitive yet unrealistic baseline included for completeness.

Setup: in our experiments, we begin by training a model on the training data using a variety of hyperparameter configurations. After each training epoch (and for every hyperparamter configuration), we evaluate the model's loss on the validation set (constructed via one of the methods above). For all construction methods except for TEST, we select the model (i.e., training epoch and hyperparameter configuration) with the lowest validation loss; for TEST we make the selection based on highest validation accuracy.

Each experiment is repeated 10 times (this includes subsampling new examples). For each validation set construction method, we report mean and standard deviation of test set accuracy. Since training sets differ in each experimental repetition—and we expect high variance—we also compute

[2] https://developers.google.com/search/docs/advanced/structured-data/qapage

[3] https://developers.google.com/search/docs/advanced/structured-data/faqpage

[4] For any class that has fewer than k examples, we select all examples of c.

[5] This value was chosen arbitrarily.

$\mathbf{k = 5}$	**bank**	**clinc**	**curekart**	**powerplay11**	**snips**	**sofmatress**
HOLDOUT	$0.70_{0.01}$	$0.85_{0.01}$	$0.58_{0.06}$	$0.51_{0.04}$	$0.87_{0.02}$	$0.59_{0.05}$
TRAIN	$0.74_{0.01}*$	$0.86_{0.01}*$	$0.54_{0.06}$	$0.53_{0.06}$	$0.86_{0.03}$	$0.60_{0.05}$
DAUGSS	$0.74_{0.01}*$	$\mathbf{0.88_{0.01}}*$	$\mathbf{0.62_{0.06}}$	$0.54_{0.03}*$	$\mathbf{0.89_{0.01}}*$	$0.64_{0.05}*$
EDA	$\mathbf{0.75_{0.01}}*$	$0.87_{0.01}*$	$0.58_{0.06}$	$\mathbf{0.55_{0.03}}*$	$0.88_{0.03}$	$\mathbf{0.65_{0.06}}*$
TEST	$0.75_{0.01}*$	$0.88_{0.01}*$	$0.67_{0.04}*$	$0.50_{0.17}$	$0.91_{0.01}*$	$0.69_{0.04}*$
$\mathbf{k = 10}$						
HOLDOUT	$0.81_{0.01}$	$0.91_{0.00}$	$0.65_{0.06}$	$0.56_{0.03}$	$0.92_{0.01}$	$0.69_{0.03}$
TRAIN	$0.81_{0.03}$	$0.90_{0.01}$	$0.64_{0.04}$	$0.57_{0.03}$	$0.89_{0.02}$	$0.67_{0.04}$
DAUGSS	$0.83_{0.01}*$	$0.91_{0.01}*$	$\mathbf{0.70_{0.04}}*$	$0.59_{0.03}*$	$0.89_{0.03}$	$\mathbf{0.72_{0.02}}*$
EDA	$\mathbf{0.84_{0.01}}*$	$\mathbf{0.92_{0.01}}*$	$0.69_{0.04}$	$\mathbf{0.60_{0.02}}*$	$\mathbf{0.93_{0.01}}$	$\mathbf{0.72_{0.03}}*$
TEST	$0.85_{0.01}*$	$0.92_{0.00}*$	$0.73_{0.03}*$	$0.63_{0.02}*$	$0.94_{0.01}*$	$0.77_{0.01}*$

Table 1: **Test Set Accuracy, FINETUNE, $\mathbf{k = \{5, 10\}}$.** Mean and standard deviation test set accuracy of models selected in the FINETUNE setting. **Bolded** text indicates the highest mean per dataset (other than TEST); asterisk (*) indicates improvement over HOLDOUT is statistically significant (1-sided Wilcoxon signed rank test, $p = 0.05$).

whether each method is significantly better than HOLDOUT using a one-sided Wilcoxon signed-rank test with significance level of $p = 0.05$ (Schuurmans, 2006; Wilcoxon, 1947). For each experimental setting, we either train all model parameters (FINETUNE) or only the parameters in the final layer (FROZEN). We include results for the FROZEN case (also known as the "linear probing" setting) since it is common when latency and/or computing cost are constrained. Moreover, FROZEN training has been shown to generalize better to out-of-distribution data than FINETUNE training when pre-trained representations are "good" (Kumar et al., 2021). This is relevant to the few-shot domain where most data may be considered out-of-distribution because of the scarcity of training data. In the main text, we report results for the HuggingFace roberta-base model optimized with the AdamW optimizer (Wolf et al., 2019; Loshchilov and Hutter, 2018).

Hyperparameters: we tune 4 hyperparameters: learning rate, weight decay, dropout among hidden units, and dropout among classifier units. We employ Optuna—a hyperparameter optimization library (Akiba et al., 2019). For each experimental setting, we give Optuna an operating budget of 100 trials (i.e., unique hyperparameter configurations) with trial pruning turned on. All models are trained for up to 30 epochs[6]. Hyperparameter ranges used during hyperparameter optimization are included in Appendix A.1.

[6]why?

Result: Table 1 contains the mean and standard deviation for each model selection method on all 6 datasets for both $k = 5$ and $k = 10$ (i.e., 5 or 10 examples per class), when training via FINETUNE. The results show that the generative methods (i.e., either DAUGSS or EDA) achieve the highest mean accuracy in all experimental conditions. We note that in some cases, when considering the standard deviation of mean accuracy over the 10 trials, error bars overlap. However, high standard deviations are anticipated (due to the use of unique subsets of data in each data set used in each experimental condition). When considering significantly significant improvements over the HOLDOUT method, we find that, for both $k = 5$ and $k = 10$, DAUGSS sees improvement in the largest fraction of data sets (i.e., 5 out of 6); more than EDA (4 out of 6) and TRAIN (2 out of 6 for $k = 5$ and never for $k = 10$). The results support the notion that both DAUGSS and EDA are consistently high-performing methods of model selection for few-shot text classification. For brevity, we include the presentation and discussion of results in the FROZEN training setting in the Appendix.

4.2 Estimating Test Set Accuracy

While selecting the best performing model is a crucial stage of a typical machine learning workflow, an accurate estimate of the best model's performance is also required for decision making with respect to model deployment. In other words, identifying the best performing model among a set of models is insufficient; developers must have

$\mathbf{k = 5}$	**bank**	**clinc**	**curekart**	**powerplay11**	**snips**	**sofmatress**
HOLDOUT	$\mathbf{0.04_{0.02}}$	$\mathbf{0.04_{0.02}}$	$0.18_{0.07}$	$0.36_{0.04}$	$0.12_{0.04}$	$0.25_{0.09}$
TRAIN	$0.26_{0.01}$	$0.14_{0.01}$	$0.46_{0.06}$	$0.47_{0.06}$	$0.14_{0.03}$	$0.40_{0.05}$
DAUGSS	$0.18_{0.02}$	$0.21_{0.01}$	$\mathbf{0.14_{0.07}}$	$\mathbf{0.19_{0.03}}$	$\mathbf{0.03_{0.02}}$	$\mathbf{0.07_{0.05}}$
EDA	$0.23_{0.01}$	$0.09_{0.02}$	$0.39_{0.06}$	$0.40_{0.03}$	$0.12_{0.03}$	$0.30_{0.06}$
$\mathbf{k = 10}$						
HOLDOUT	$\mathbf{0.03_{0.02}}$	$\mathbf{0.03_{0.01}}$	$0.19_{0.07}$	$0.32_{0.02}$	$0.07_{0.02}$	$0.24_{0.06}$
TRAIN	$0.19_{0.03}$	$0.10_{0.01}$	$0.36_{0.04}$	$0.43_{0.03}$	$0.11_{0.02}$	$0.33_{0.04}$
DAUGSS	$0.34_{0.01}$	$0.28_{0.01}$	$\mathbf{0.03_{0.03}}$	$\mathbf{0.14_{0.03}}$	$\mathbf{0.06_{0.05}}$	$\mathbf{0.06_{0.03}}$
EDA	$0.14_{0.01}$	$\mathbf{0.03_{0.01}}$	$0.29_{0.05}$	$0.38_{0.02}$	$0.07_{0.01}$	$0.24_{0.03}$

Table 2: **FINETUNE Model Fidelity,** $k = 5$**.** The mean and standard deviation of the absolute difference between validation accuracy and test set accuracy of the selected model. **Bolded** text indicates the lowest mean per dataset.

an accurate sense of the model's performance in practice—among many other statistics—in determining whether that model is appropriate for use. We underscore that before deployment, many evaluations must be performed—among which test set accuracy is only one (Ribeiro et al., 2020).

4.2.1 Validation Accuracy of Selected Model

To this end, we measure the extent to which validation accuracy is a faithful estimator of accuracy on the test set for the methods discussed above. In Table 2 we report the mean and standard deviation of the absolute difference between validation and test set accuracy for models selected via each method in the FINETUNE setting for $k = 5$ and $k = 10$. For **bank** and **clinc**, HOLDOUT gives the highest fidelity estimate of test set accuracy. This is unsurprising since the validation set, while small, is (in some sense) drawn from the testing distribution. DAUGSS achieves the lowest mean absolute difference for the remaining four datasets, which supports the notion that when using DAUGSS, it is possible to generate examples of similar difficulty as those in the training set.

Notably, the two datasets for which HOLDOUT yields the highest fidelity estimates—**bank** and **clinc**—are the largest datasets. On the 4 smaller datasets, validation accuracy is an approximately 3x worse predictor of test set accuracy, with higher variability. With a single validation example per class, validation accuracy is a very coarse approximation of test set accuracy.

Validation sets comprised of EDA or TRAIN examples provide poor estimates of test set accuracy. This is unsurprising. Using the train set as the validation set leads to models with high validation performance that belies test set performance. Since EDA generates new examples by perturbing training instances, validation sets constructed via EDA will lead to similarly high performance on the validation set—and overestimates of test set accuracy.

4.2.2 All Hyperparameter Configurations

For completeness, we examine the faithfulness of validation accuracy to test accuracy for all hyperparameter configurations tested, all datasets, and all epochs. We report the root mean square error (RMSE) between validation accuracy and test set accuracy in Table 3. This gives a sense of how accurate test set accuracy can be predicted by validation accuracy for *any* model when the validation set is built according to each of the methods tested. This would be important if, for example, model selection were done using one method, but estimating test set accuracy were done by another.

The Table shows that for all training settings except for FROZEN training with $k = 10$, DAUGSS yields the highest fidelity estimates of test set accuracy. Moreover, for the case of FROZEN training with $k = 10$ it achieves the second lowest RMSE, following HOLDOUT. This supports the notion that a validation accuracy, when computed from a small set of validation examples, may not be a good estimator of test set accuracy. On the other hand, generation of the validation set with DAUGSS generally leads to more accurate estimates of test set accuracy than when the validation set is constructed using one of the other methods test; especially when the amount of data is small ($k = 5$).

For a more detailed view, we visualize the correlation between validation accuracy and test set accuracy in Figure 2. The Figure shows that the fidelity of validation accuracy with HOLDOUT is

	FROZEN-5	FROZEN-10	FINETUNE-5	FINETUNE-10
HOLDOUT	0.05568	**0.04466**	0.13285	0.11662
TRAIN	0.33245	0.27272	0.30056	0.23826
DAUGSS	**0.00143**	0.09386	**0.00397**	**0.09185**
EDA	0.22912	0.16706	0.24456	0.18061

Table 3: **RMSE of Validation Accuracy.** The root mean square error with respect to validation accuracy and test set accuracy for all methods and training regimes. RMSE is computed from all hyperparameter configurations, all epochs, and all datasets. **Bolded** text indicates the lowest RMSE per condition. Note that RMSE for TEST is 0 (by definition).

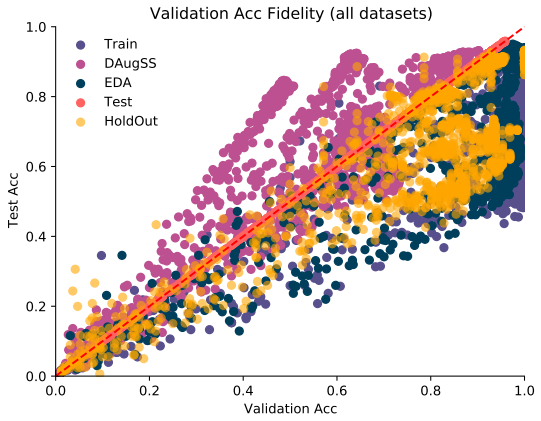

Figure 2: **Fidelity of Validation Accuracy, FINETUNE, k = 5.** Validation set accuracy versus test set accuracy for all hyperparamter configurations, all epochs, and for all datasets.

highly variable, and tends to overestimate test set accuracy (i.e., validation accuracy is greater than test set accuracy). On the other hand, validation sets constructed by DAUGSS lead to consistent underestimates of test set accuracy with low RMSE. Neither EDA nor TRAIN make for validation sets from which test set accuracy can be accurately estimated. Unsurprisingly, both lead to overestimates (o test set accuracy). We include similar figures for the other experimental conditions in Appendix A.3.

5 Related Work

Model selection is a critical component of machine learning workflows, and especially of few-shot learning (Bragg et al., 2021). Despite this, most experimentation with few-shot learning methods either ignores model selection or assumes validation data is available, which is unrealistic (Perez et al., 2021). Our work is a first systematic study of model selection in few-shot text classification, with a focus on validation set construction via data augmentation.

Prior to our work, two other pieces have leveraged generative models to construct validation data. In Datasets from Instructions (DINO), a pre-trained GPT2-XL is prompted to generate labeled sentence pairs to support learning improved sentence embeddings (Schick and Schütze, 2021). The set of generated pairs is split into training and validation sets. In this work, the validation set is used to determine when to (early) stop training, but it is unclear whether it is also used to select among a range of hyperparameter configurations. In our study, we use constructed validation sets to select 4 important hyperparameters, in addition to early stopping. That tasks we focus on are domain-specific, where as DINO is aimed and learning better general-purpose sentence embeddings–where it may be easier to generate relevant data for validation.

The second piece studies prompt order for "in-context learning" (Brown et al., 2020), i.e., when the model is given a handful of examples of a task at inference time but no weights are updated. The authors find that the order of the examples in the prompt used for in-context learning can significantly affect results (fluctuations between state-of-the-art and random chance performance were observed) (Lu et al., 2022). To alleviate this high sensitivity in true few-shot settings, the authors generate an *unlabeled* validation set with a large pre-trained language model and use the set to select prompt orders via a proposed entropy-based method. Unlike their study, we focus on the FINETUNE and FROZEN cases rather than in-context learning. Moreover, we select specific values of continuous hyperparamters rather than the best among 24 prompt-permutations. Finally, we point out that the proposed approach for prompt-order selection cannot be directly used to estimate test set accuracy (as we study in Section 4.2).

A central component of our work is our proposed DAUGSS algorithm. DAUGSS is inspired in part by Example Extrapolation (EX2), which also uses

training examples as prompts; however their work is designed for imbalanced classes rather than few-shot settings (Lee et al., 2021). Like we do, previous work makes use of a sequence-to-sequence model for generation, but unlike our work, their approaches focuses on filling in delexicalized utterances (Hou et al., 2018). Our use of guide words is also similar in spirit to previous work on decoding (Pascual et al., 2021). However, their approach can guarantee that guide words appear in the output by shifting token generation probabilities—which we do not require from our generators.

While we experiment with a handful of approaches, there is a large and growing literature on data augmentation for NLP. We briefly touch on some recently proposed methods, but refer interested readers to surveys on the subject (Feng et al., 2021). Most data augmentation algorithms can be roughly categorized as either retrieval (Du et al., 2021), perturbation (Wei and Zou, 2019), feature (Kumar et al., 2019; Sun et al., 2020; Wei, 2021), or generation-based (Wang et al., 2021; Kumar et al., 2020; He et al., 2021; Yang et al., 2020). Some work focuses on counterfactual augmentation (Kaushik et al., 2020; Joshi and He, 2021); likewise, generating minimally perturbed training examples with different labels (Zhou et al., 2021). In the literature, augmentation is generally employed as a tool for improving test set accuracy. But a recent study explores augmentation for mitigating gender stereotypes (Zmigrod et al., 2019). Unlike our work, virtually all previous studies focused on training set augmentation rather than validation set construction.

6 Conclusion

In this work we study true-few shot classification, i.e., few-shot classification when no dedicated validation set is provided for model selection. We experiment with constructing validation sets via known data augmentation techniques, as well as our proposed technique, DAUGSS, which is designed for true few-shot generative data augmentation. When a RoBERTa model is trained in the FINETUNE setting, models selected using validation sets constructed by DAUGSS most consistently achieve the highest test set accuracy as well as the highest fidelity estimates of test set accuracy among all competing methods tested. In the FROZEN case, DAUGSS also yields high performing models, but so does model selection via the training loss. Overall, DAUGSS provides the highest fidelity estimates of test set accuracy across all data sets, training epochs, and hyperparameter configurations.

7 Limitations

In this work, we study various methods of validation set construction for the true few-shot setting. While we show positive results for model selection via a number of methods, our experiments only deal with few-shot text classification. Moreover, the datasets we use do not cover natural language inference, which may require more complex reasoning about semantics than validation sets constructed with DAUGSS can provide. All of our experiments are conducted on English language data sets. Additionally, our experiments include subsampled data sets with either 5 or 10 examples per class (when enough examples per class exists), but we do not explicitly experiment with (intentionally) unbalanced data sets. We only experiment with the RoBERTa model. We choose RoBERTa because it is high-performing and ubiquitous (and therefore admits comparison to other work), but we acknowledge that better models exist and may provide different results. Despite these limitations, we believe that our results are sound and likely to generalize to models aside from RoBERTa. Finally, we do not experiment with in-context learning methods (i.e., prompting with GPT-3); but we argue that FROZEN and FINETUNE are still prominent training paradigms.

Acknowledgements

This document has been adapted by Steven Bethard, Ryan Cotterell and Rui Yan from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTEX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats

written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Gaurav Arora, Chirag Jain, Manas Chaturvedi, and Krupal Modi. 2020. Hint3: Raising the bar for intent detection in the wild. In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 100–105.

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. Flex: Unifying evaluation for few-shot nlp. *Advances in Neural Information Processing Systems*, 34.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*. Data available at https://github.com/PolyAI-LDN/task-specific-datasets.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5408–5418, Online. Association for Computational Linguistics.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP.

Xuanli He, Islam Nassar, Jamie Kiros, Gholamreza Haffari, and Mohammad Norouzi. 2021. Generate, annotate, and learn: NLP with synthetic text.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-Sequence data augmentation for dialogue language understanding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1234–1245, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Nitish Joshi and He He. 2021. An investigation of the (in)effectiveness of counterfactually augmented data.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2020. Learning the difference that makes a difference with Counterfactually-Augmented data. In *International Conference on Learning Representations*.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. 2021. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and Wlliam Campbell. 2019. A closer look at feature space data augmentation for Few-Shot intent classification. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 1–10, Hong Kong, China. Association for Computational Linguistics.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation.

Shayne Longpre, Yu Wang, and Chris DuBois. 2020. How effective is Task-Agnostic data augmentation for pretrained transformers? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4401–4411, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming Few-Shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A plug-and-play method for controlled text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3973–3997.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *NeurIPS*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6951, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Editor: Dale Schuurmans. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip Yu, and Lifang He. 2020. Mixup-Transformer: Dynamic data augmentation for NLP tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3436–3440, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards Zero-Label language learning.

Jason Wei. 2021. Good-Enough example extrapolation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5923–5929, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Frank Wilcoxon. 1947. Probability tables for individual comparisons by ranking methods. *Biometrics*, 3(3):119–122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics.

Jing Zhou, Yanan Zheng, Jie Tang, Jian Li, and Zhilin Yang. 2021. FlipDA: Effective and robust data augmentation for Few-Shot learning.

Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661, Florence, Italy. Association for Computational Linguistics.

A Experiments

A.1 Hyperparameters

For hyperparameter optimization, we use Optuna (Akiba et al., 2019). Optuna allows a practitioner to identify the hyperparameters over which to conduct the search, as well as the allowable ranges. In our experiments, Optuna tunes the following 4 parameters with the following ranges:

1. learning rate, $[0.00001, 0.1]$;
2. weight decay, $[0.0, 0.1]$;
3. dropout among hidden units, i.e., `hidden_dropout_prob`, $[0.0, 0.5]$; and
4. dropout among classification head units, i.e., `classifier_dropout`, $[0.0, 1.0]$.

Optuna performs 100 trials (each trial may be pruned if the corresponding hyperparameters are deemed unlikely to yield a high performing model. New configurations are sampled using the `TPESampler` (the random seed is set to 37). Training in a full trial lasts for 30 epochs.

A.2 Model Selection with FROZEN Training

In this section we present the results of model selection when training is carried out in the FROZEN setting. Like in the FINETUNE setting, both generative methods (i.e., DAUGSS and EDA) achieve many statistically significant improvements over HOLDOUT: DAUGSS offers improvement in all experimental setting (for both $k = 5$ and $k = 10$) while EDA offers improvements in 5 out of 6 experimental conditions for both $k = 5$ and $k = 10$. However, in this case TRAIN achieves the highest mean accuracy in most experimental conditions (and also offers statistically significant improvement over the baseline in all but 1 experimental condition (for **snips** when $k = 5$).

These results are surprising since using the training set as the validation set is (intuitively) likely to cause overfitting. However, we note that in the FROZEN setting (in which only the last layer of parameters are trainable), overfitting is (perhaps) less likely. Moreover, model selection using the training set is equivalent to selecting the model with the lowest training loss which: i) is a strategy employed when a validation set is unavailable, and ii) is also somewhat performant in the FINETUNE case (Section 4.1). While DAUGSS and EDA offer consistent improvements as in the previous results (Section 4.1), the primary lesson learned from these results is that in few-shot cases, training with all provided data yields better models than holding out data for validation, and that model selection can even be done by selecting the model with the lowest training loss.

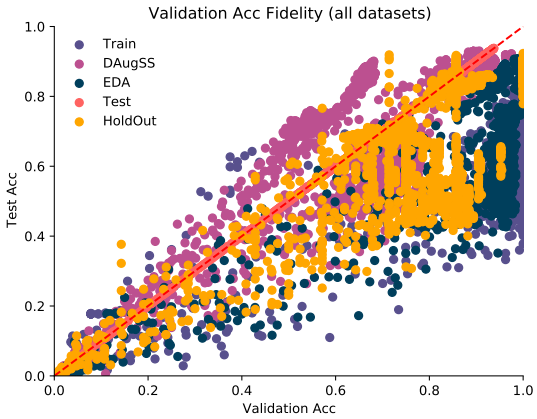

Figure 3: **Fidelity of Validation Accuracy, FINETUNE, k = 10.** Validation set accuracy versus test set accuracy for all hyperparameter configurations, all epochs, and for all datasets.

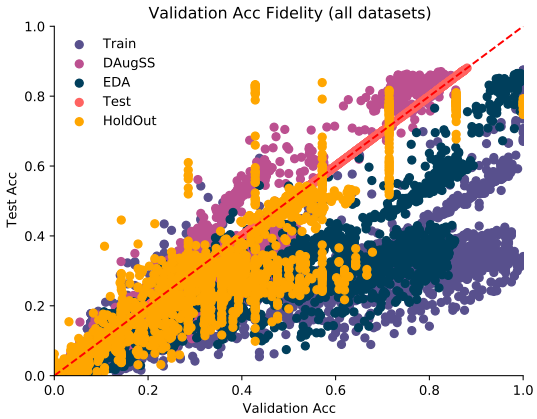

Figure 4: **Fidelity of Validation Accuracy, FROZEN, k = 5.** Validation set accuracy versus test set accuracy for all hyperparamter configurations, all epochs, and for all datasets.

A.3 Fidelity of Validation Set Accuracy

$\mathbf{k=5}$	**bank**	**clinc**	**curekart**	**powerplay11**	**snips**	**sofmatress**
HOLDOUT	$0.33_{0.01}$	$0.52_{0.01}$	$0.28_{0.04}$	$0.29_{0.04}$	$0.76_{0.10}$	$0.30_{0.02}$
TRAIN	$\mathbf{0.39_{0.01}}*$	$\mathbf{0.60_{0.01}}*$	$\mathbf{0.36_{0.03}}*$	$\mathbf{0.33_{0.03}}*$	$0.79_{0.10}$	$0.37_{0.03}*$
DAUGSS	$\mathbf{0.39_{0.01}}*$	$\mathbf{0.60_{0.01}}*$	$0.35_{0.04}*$	$0.32_{0.03}*$	$\mathbf{0.84_{0.02}}*$	$\mathbf{0.40_{0.03}}*$
EDA	$\mathbf{0.39_{0.01}}*$	$\mathbf{0.60_{0.01}}*$	$0.35_{0.03}*$	$0.32_{0.03}*$	$0.75_{0.12}$	$0.38_{0.02}*$
TEST	$0.52_{0.03}*$	$0.69_{0.01}*$	$0.44_{0.01}*$	$0.35_{0.02}*$	$0.86_{0.02}*$	$0.38_{0.02}*$
$\mathbf{k=10}$						
HOLDOUT	$0.54_{0.01}$	$0.74_{0.01}$	$0.43_{0.03}$	$0.34_{0.02}$	$0.85_{0.04}$	$0.38_{0.02}$
TRAIN	$\mathbf{0.66_{0.03}}*$	$\mathbf{0.82_{0.01}}*$	$\mathbf{0.54_{0.04}}*$	$\mathbf{0.37_{0.02}}*$	$\mathbf{0.88_{0.01}}*$	$\mathbf{0.52_{0.02}}*$
DAUGSS	$0.59_{0.02}*$	$0.77_{0.01}*$	$0.50_{0.03}*$	$\mathbf{0.37_{0.03}}*$	$\mathbf{0.88_{0.01}}*$	$0.46_{0.03}*$
EDA	$0.61_{0.03}*$	$0.81_{0.01}*$	$0.53_{0.04}*$	$\mathbf{0.37_{0.02}}*$	$\mathbf{0.88_{0.02}}$	$0.46_{0.02}*$
TEST	$0.67_{0.02}*$	$0.82_{0.01}*$	$0.56_{0.05}*$	$0.41_{0.03}*$	$0.89_{0.01}*$	$0.50_{0.03}*$

Table 4: **Test Set Accuracy, FROZEN,** $k = \{5, 10\}$**.** Mean and standard deviation test set accuracy of models selected in the FROZEN setting. **Bolded** text indicates the highest mean per dataset (other than TEST); asterisk (*) indicates improvement over HOLDOUT is statistically significant (1-sided Wilcoxon signed rank test, $p = 0.05$).

$\mathbf{k=5}$	**bank**	**clinc**	**curekart**	**powerplay11**	**snips**	**sofmatress**
HOLDOUT	$\mathbf{0.05_{0.04}}$	$\mathbf{0.04_{0.04}}$	$0.15_{0.06}$	$0.18_{0.09}$	$0.13_{0.11}$	$0.15_{0.11}$
TRAIN	$0.45_{0.02}$	$0.37_{0.01}$	$0.59_{0.04}$	$0.55_{0.03}$	$0.20_{0.09}$	$0.52_{0.07}$
DAUGSS	$0.08_{0.02}$	$0.15_{0.01}$	$\mathbf{0.08_{0.04}}$	$\mathbf{0.07_{0.03}}$	$\mathbf{0.04_{0.03}}$	$\mathbf{0.09_{0.04}}$
EDA	$0.20_{0.03}$	$0.26_{0.01}$	$0.47_{0.03}$	$0.40_{0.02}$	$0.18_{0.06}$	$0.41_{0.03}$
$\mathbf{k=10}$						
HOLDOUT	$\mathbf{0.03_{0.02}}$	$\mathbf{0.04_{0.02}}$	$\mathbf{0.06_{0.04}}$	$0.09_{0.05}$	$\mathbf{0.07_{0.07}}$	$0.21_{0.08}$
TRAIN	$0.32_{0.01}$	$0.18_{0.01}$	$0.40_{0.03}$	$0.45_{0.03}$	$0.12_{0.01}$	$0.48_{0.02}$
DAUGSS	$0.25_{0.01}$	$0.28_{0.01}$	$0.08_{0.03}$	$\mathbf{0.02_{0.02}}$	$0.12_{0.03}$	$\mathbf{0.04_{0.02}}$
EDA	$0.10_{0.02}$	$0.10_{0.01}$	$0.26_{0.03}$	$0.37_{0.02}$	$0.10_{0.02}$	$0.34_{0.02}$

Table 5: **FROZEN Model Fidelity,** $k = \{5, 10\}$**.** The mean and standard deviation of the absolute difference between validation accuracy and test set accuracy of the selected model. **Bolded** text indicates the lowest mean per dataset.

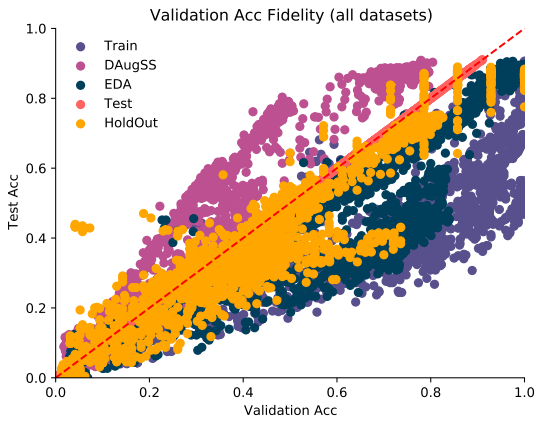

Figure 5: **Fidelity of Validation Accuracy, FROZEN, k = 10.** Validation set accuracy versus test set accuracy for all hyperparamter configurations, all epochs, and for all datasets.