Thesis Notes-

Adaptive Probabilistic Testing of DNNs and LLMs: The DeepSample Framework

Methodology

Probabilistic *operational testing* treats model evaluation as a statistical sampling problem. Instead of exhaustively labeling a massive test set, DeepSample builds a small yet representative sample of inputs drawn according to the model's expected operational profile. The goal is threefold: use a minimal test set (to cut labeling cost) that still yields an unbiased, high-confidence estimate of the model's accuracy, while also exposing as many mis-predictions (failures) as possible. To guide this sample selection, DeepSample leverages *auxiliary variables* – heuristics correlated with the model's error probability. For classification DNNs or LLMs, examples include the model's predicted *Confidence* score (the top-class probability), *Prediction Entropy* (uncertainty of the output distribution), or Surprise Adequacy metrics like *DSA/LSA* which measure how novel an input is relative to training data. By biasing selection toward inputs with low confidence or high surprise (which are more likely to be mispredicted), the tester can **improve estimation accuracy and efficiency** – fewer samples are needed to assess performance – so long as statistical corrections are applied to avoid bias.

DeepSample encompasses a family of sampling strategies (summarized below) that use these auxiliary cues in different ways. All methods ultimately produce an unbiased estimate of the model's accuracy by using appropriate estimators to account for any unequal selection probabilities. The main strategies include:

- **Simple Random Sampling (SRS)** – A baseline where all inputs have an equal chance of being selected. This yields a straightforward unbiased accuracy estimate but does not specifically target difficult cases.
- **Simple Unequal Probability Sampling (SUPS)** – Assigns each input a selection probability proportional to an auxiliary variable (e.g. inverse confidence), *prioritizing examples with higher estimated failure likelihood*. This "probability proportional-to-size" sampling boosts the inclusion of risky cases.
- **RHC-Sampling (RHC-S)** – An unequal-probability method *without replacement*. It divides the pool and weights examples using the Rao-Hartley-Cochran scheme for accurate unbiased estimation. RHC-S avoids picking the same test twice and requires a special estimator to correct bias.
- **Stratified Simple Random Sampling (SSRS)** – Partitions the dataset into strata (e.g. by ranges of an auxiliary variable's variance) and then samples within each partition. The sample allocation follows Neyman's optimal strategy, aiming to reduce overall estimation error by giving more weight to high-variability strata.

- **Gradient-Based Sampling (GBS)** – An *adaptive* stratified approach that decides iteratively which partition to sample next. Inspired by gradient descent variance reduction, GBS dynamically adjusts sampling frequencies based on which stratum would most reduce the uncertainty of the accuracy estimate.
- **Two-stage Unequal Probability Sampling (2-UPS)** – A hierarchical strategy that first uses an auxiliary variable to probabilistically choose a partition, then performs a random selection within that partition. By combining focused partition choice with internal randomness, 2-UPS aims for representative yet cost-effective coverage of the input space.
- **DeepEST (Deep Enhanced Sampler for Testing)** – An adaptive sampling method originally proposed by Guerriero *et al.* that *maximizes failure detection* while still providing unbiased accuracy estimates. DeepEST continually updates the sampling distribution to **balance** two objectives: finding as many mispredicted inputs as possible and accurately estimating overall accuracy.

All the above techniques reduce the required test size compared to naive approaches by focusing on the most informative inputs. Crucially, they remain statistically *unbiased* – any oversampling of “difficult” cases is compensated by weighting in the accuracy estimator. In essence, *DeepSample uses sampling theory to build small yet representative test sets, cutting down labeling costs while still delivering unbiased accuracy measurements*. This probabilistic approach to DNN/LLM testing ensures that even with limited manual labels, we can obtain reliable performance metrics and uncover critical failure cases.

Results

The DeepSample framework has been evaluated on both vision and NLP models, including a case study using **Distil BERT** for sentiment analysis across three datasets. The findings demonstrate that adaptive sampling can drastically improve evaluation efficiency. In terms of **estimation accuracy**, many advanced strategies achieved lower error (RMSE) in accuracy estimates than uniform random sampling on these sentiment tasks. Interestingly, the simple SRS baseline proved hard to beat on a large, balanced dataset (the full IMDb reviews): in that case, SRS and GBS yielded the lowest RMSE, outperforming more complex methods under both confidence- and entropy-based sampling. On a smaller dataset (SST-2), however, a technique like 2-UPS had the best accuracy estimation when guided by the Confidence score – whereas if it used Prediction Entropy on that same set, its RMSE jumped to one of the worst. This highlights that the *choice of auxiliary variable can influence accuracy results*, and methods respond differently to each dataset’s characteristics. Overall, the study observed that methods such as **DeepEST, GBS, RHC-S, SUPS, SRS,** and **2-UPS** consistently delivered more precise accuracy estimates than a purely stratified approach (SSRS). In other words, most sampling-based techniques outperformed naive stratification in terms of estimation error, reinforcing their suitability for high-confidence model evaluation.

Beyond just estimating accuracy, DeepSample techniques proved adept at **misprediction detection** – i.e. finding the test inputs that the model gets wrong. Here, the more *biased* sampling schemes have a clear advantage. Methods heavily driven by auxiliary signals (like low confidence) uncovered substantially more failures under the same test budget than random sampling. For example, a stratified sampler (SSRS) was able to identify around **79**

mispredictions in an IMDb sentiment model, whereas a simple random sample of the same size found only about **20** failures. In general, **SUPS, DeepEST, and SSRS** excelled at exposing model mistakes, significantly outperforming other methods on this metric. SUPS in particular, by always pulling in low-confidence examples, often discovered the most failures early on. This aggressive fault-finding comes at a slight cost to estimation accuracy, but is invaluable when the tester's priority is to reveal problem cases for model debugging. Notably, even RHC-S – which uses a conservative balanced estimator – showed strong failure-finding capability when *prediction entropy* was the guiding variable, underscoring that the right auxiliary choice can turn a method into a failure-hunting tool.

Another aspect investigated was **sensitivity to sample size**. As expected, increasing the test budget improves accuracy estimates for all techniques: with a large sample (e.g. 800 instances) the RMSE dropped markedly compared to very small samples (50 instances) across the board. The performance gap between techniques also narrows at larger sample sizes, since with enough tests even a simple random sample can approximate the true accuracy closely. That said, the trajectory of improvement differs by method. Some strategies yield big gains early: for instance, SUPS achieved very low error with small samples, but then **plateaued** as more samples were added. In contrast, SSRS started with the *highest* error (since its initial sample may over/under-emphasize certain strata), yet it continued to significantly improve as the sample grew, eventually closing much of the gap. We also see that certain methods maintain an advantage regardless of budget – SRS and GBS remained consistently reliable in estimation, even with just tens of samples, especially when using a well-chosen auxiliary. In summary, a larger test budget always helps, but methods like SUPS can deliver more value in the *low-budget regime*, whereas others like SSRS need a larger budget to reach their potential. DeepSample's adaptive approaches thus allow testers to tune their strategy based on how many labels they can afford – ensuring either quick insights with very few tests, or robust accuracy as tests scale up.

Discussing Points-

The comparative results reveal clear **trade-offs between sampling techniques**, suggesting each has a "sweet spot" depending on testing objectives. If the primary goal is to obtain an **unbiased accuracy estimate** with minimal variance, methods that sample broadly and evenly (or adapt to maintain balance) tend to shine. In the experiments, both GBS and even the basic SRS delivered consistently low error across different auxiliary choices, indicating that they are **robust choices for accuracy estimation**. GBS in particular uses feedback to minimize estimation uncertainty, making it a strong option when confidence intervals on accuracy are critical. 2-UPS can also be effective on this front, but it showed sensitivity to the auxiliary variable – performing best with Confidence and less so with Entropy. RHC-S similarly provided accurate estimates and proved especially useful when the tester trusts a predictive uncertainty measure: with Entropy as aux, RHC-S achieved excellent results (both in accuracy and finding difficult cases). In general, **RHC-S and GBS** are good choices when one wants adaptive yet principled sampling that maintains statistical balance, whereas **2-UPS and SUPS** offer more heuristic-driven sampling that can excel or falter depending on the context.

On the other hand, if the testing objective leans more toward **bug-finding and model improvement**, strategies that aggressively target likely failures are more beneficial. **SUPS** is a prime example: by skewing heavily toward low-confidence inputs, it uncovers many errors very quickly. This makes SUPS ideal in scenarios like rapid preliminary testing or safety-critical validation, where catching as many issues as possible with a small number of tests is worth a slight sacrifice in estimate precision. **DeepEST** and **SSRS** also fit this use case – both were *highly effective at exposing failures*, with SSRS offering a more systematic coverage of input sub-populations and DeepEST dynamically hunting for rare errors. These approaches are well-suited when testers suspect the model may have blind spots in certain regions of the input space and want to ensure those are exercised. The flip side is that they might over-sample difficult cases, inflating variance in the accuracy estimate; however, because DeepSample frameworks apply proper statistical weighting, the final accuracy calculation remains unbiased. Thus, the main "cost" of prioritizing failures is the larger error bars on the metrics, not systematic bias.Choosing the right technique often comes down to the **auxiliary variable and data characteristics**. The studies found that some surrogate measures of difficulty are more reliable than others in guiding the sample. For example, model *confidence* is a convenient indicator, but it can be overconfident on out-of-distribution inputs (missing some failures). In those cases, surprise adequacy metrics like LSA/DSA provided more stable guidance, leading to better estimation outcomes. Each DeepSample method can work with different auxiliaries, so practitioners should consider domain-specific proxies for error-proneness – e.g. for vision models, reconstruction error from an autoencoder might flag unusual images, while for language models, response length or ambiguity might correlate with errors. Notably, the DeepSample evaluation on DistilBERT showed that using *either* Confidence or Entropy as aux could yield good results with the adaptive methods, but certain pairings (like RHC-S with Entropy, or 2-UPS with Confidence) were particularly effective. This suggests an opportunity: testers can **match the strategy to the auxiliary** that best captures their model's weaknesses. In practice, one might even employ a hybrid approach – e.g. start with a broad stratified sample to ensure coverage, then switch to an uncertainty-weighted sampling to drill down on likely errors. The DeepSample framework's value lies in offering this flexible toolkit, where depending on whether the immediate priority is *evaluation* or *debugging*, the tester can pick a method (or combination) that yields the most insight for the labeling effort spent.

Avenues to look for….

As the DeepSample approach has shown promise on classification tasks, a natural next step is extending it to a wider range of **LLM applications**. Large Language Models are used in tasks like question answering, text summarization, and free-form generation – domains where evaluating correctness or quality is more complex than a simple label comparison. The DeepSample framework could be adapted to these scenarios by designing appropriate auxiliary variables and sampling criteria. For instance, in a QA system, an auxiliary could be the model's answer confidence or the consistency of its answer with a retrieved knowledge source. In text summarization, one might use metrics such as novelty or divergence from the source text as proxies for summary quality or error. The authors of DeepSample indeed propose to apply probabilistic operational testing to tasks like QA and summarization to validate the framework's generalizability. A challenge in these domains is defining what constitutes a "failure" (since errors can be subjective or gradated), but the same principle of

sampling from an operational input distribution holds. By focusing testing on inputs that the LLM finds most uncertain or that deviate most from training data, we can potentially catch coherence issues or factual errors more efficiently than random testing.
Another promising avenue is to explore **hybrid and adaptive strategies** that dynamically balance the twin goals of accuracy estimation and failure discovery. DeepSample already provides distinct methods that favor one or the other; a future approach could *combine their strengths*. For example, an initial testing phase might use a partitioning strategy (like SSRS or 2-UPS) to cover all input types and get a coarse accuracy read, and a second phase could deploy an unequal sampling (like SUPS or DeepEST) focusing on the hardest cases identified in phase one. Such a hybrid approach could ensure no corner of the input space is overlooked while still homing in on the worst-case behaviors. Additionally, one could imagine an **adaptive scheduler** that monitors intermediate results: if the estimated accuracy's confidence interval is too wide, it allocates more samples to methods that reduce variance (GBS, SRS); if instead the estimate stabilizes early, it diverts the remaining budget to failure-focused sampling (SUPS, SSRS) to uncover more issues. This kind of dynamic allocation of test budget could yield an even better cost-performance trade-off, essentially tuning the testing policy in real-time. In summary, future work will likely integrate DeepSample's techniques into a cohesive *adaptive testing loop*, possibly involving active learning: the model's identified mistakes could be fed back for retraining, and the sampling distribution updated to continually probe new weaknesses. By iterating this process, one can envision a semi-automated system for LLM evaluation and improvement that is both data-efficient and thorough.

Personal Views-

The DeepSample framework represents a significant advancement in the testing of AI models, bridging the gap between rigorous evaluation and practical efficiency. By harnessing probabilistic sampling theory, it enables **scalable and cost-effective assessment** of DNNs and LLMs, without needing to label enormous evaluation sets. At the same time, it provides statistically **unbiased performance metrics**, giving engineers and researchers confidence that the reported accuracy truly reflects operational behavior. Experiments with DistilBERT and other models have validated that sampling-based testing can maintain reliability while drastically cutting down manual labeling effort. Moreover, DeepSample's techniques serve a dual purpose: not only do they estimate accuracy, but they also surface the model's blind spots (e.g. mispredictions) in a systematic way. This is especially crucial for high-stakes applications – the framework helps ensure that models are not just accurate on average, but that their *failures* are understood and can be addressed. In essence, **probabilistic operational testing** as embodied by DeepSample offers a principled solution to evaluate AI models with high confidence while minimizing cost. It delivers practical insights to testers and addresses critical gaps in current DNN/LLM evaluation practices, making it a valuable tool for both industry practitioners and academic researchers. As AI systems continue to grow in scale and complexity, approaches like DeepSample will be key to sustaining robust, reliable model development – allowing us to test smarter, not just harder, in the era of deep and large language models.