ALERT: A Framework for Efficient Extraction of Attack Techniques from Cyber Threat Intelligence Reports Using Active Learning

Fariha Ishrat Rahman[1], Sadaf MD Halim[1], Anoop Singhal[2], and Latifur Khan[1]

[1] The University of Texas at Dallas
{farihaishrat.rahman, sadafmd.halim, lkhan}@utdallas.edu
[2] National Institute of Standards and Technology
anoop.singhal@nist.gov

Abstract. In the dynamic landscape of cybersecurity, curated knowledge plays a pivotal role in empowering security analysts to respond effectively to cyber threats. Cyber Threat Intelligence (CTI) reports offer valuable insights into adversary behavior, but their length, complexity, and inconsistent structure pose challenges for extracting actionable information. To address this, our research focuses on automating the extraction of attack techniques from CTI reports and mapping them to the MITRE ATT&CK framework. For this task, fine-tuning Large Language Models (LLMs) for downstream sequence classification shows promise due to their ability to comprehend complex natural language. However, fine-tuning LLMs requires vast amounts of annotated domain-specific data, which is costly and time-intensive, relying on the expertise of security professionals. To meet these challenges, we propose ALERT, a novel cybersecurity framework which leverages active learning strategies in conjunction with an LLM. This approach dynamically selects the most informative instances for annotation, thereby achieving comparable performance with a significantly smaller dataset. By prioritizing the annotation of samples that contribute the most to the model's learning, our methodology optimizes the allocation of resources. As a result, our framework achieves comparable performance with a dataset that is 77% smaller, making it more efficient for extracting and mapping attack techniques from CTI reports to the ATT&CK framework.

Keywords: Active Learning · LLM · ATT&CK · CTI

1 Introduction

Cybersecurity is a major challenge in today's world, as cyber criminals use advanced and diverse TTPs (Tactics, Techniques, and Procedures) to evade detection and cause damage. One of the most notorious examples of this is the WannaCry ransomware attack [2], which encrypted the data and systems of around 200,000 computers across over 150 countries and demanded ransom for their release. To help combat these growing threats, security analysts rely on

Cyber Threat Intelligence (CTI) reports [28], which provide insights into the behavior of cyber adversaries. These reports contain useful information about CTI artifacts (e.g., IoCs) and attacker behavior (e.g., TTPs). However, applying threat information contained within these resources effectively is still a challenge. The main reason is that CTI reports are often lengthy, complex, and inconsistent in their structure and terminology, which makes it difficult for analysts to extract and use the relevant information.

Various methods have been developed for extracting pertinent information from Cyber Threat Intelligence (CTI) reports. Techniques such as ChainSmith [34] and iACE [17] primarily focus on the identification of Indicators of Compromise (IoCs) using natural language processing (NLP). However, there is a noticeable shift towards higher-level extraction of Tactics, Techniques, and Procedures (TTPs) as they offer greater resilience against changes by adversaries [6], exemplified by methods like TTPDrill [13] and AttacKG [16]. However, despite the progress demonstrated by these approaches, rule-based methods exhibit limitations when dealing with the complexity of CTI reports [3]. Consequently, the field is increasingly turning towards more sophisticated models, such as Large Language Models (LLMs).

Recent research has explored leveraging pre-trained LLMs to extract attack techniques from CTI reports and map them to the ATT&CK framework [3,25]. ATT&CK [22] offers a comprehensive repository of TTPs utilized by real-world attackers. These TTPs are presented in a matrix of tactics and techniques, providing valuable insights for security analysts. Identifying attack behavior is the first step in defense against adversaries. Once ATT&CK techniques are identified, appropriate countermeasures can be taken using tools like D3FEND [20].

For this task, pre-trained LLMs were further fine-tuned by training on domain-specific datasets such as texts from CTI reports. While fine-tuned LLMs demonstrate good performance, their training demands substantial annotated data, which is scarce in the cybersecurity domain. Given the ever-evolving threat landscape, annotating large datasets becomes impractical. Hence, there is a pressing need for more efficient annotation methods in this domain—an area that remains largely unexplored.

Our proposed solution involves integrating active learning strategies with an LLM. By dynamically selecting informative instances for annotation, we achieve comparable performance while significantly reducing the annotation burden. The guiding intuition here is that not all data instances provide equal value to a learner at any given point in time. For instance, a lot of data points may involve repetition or similar data – i.e., they discuss similar content and are therefore labeled similarly. Therefore, if a learner has already encountered similar samples and learned from them, it would be wasteful to utilize annotation resources in order to annotate yet another similar unlabeled data instance. With large volumes of unlabeled threat data arriving continuously, and with annotation requiring a security expert's valuable time, it is crucial that we perform annotation in the most effective way possible. Active Learning helps identify the utility of each potential unlabeled sample with respect to the current learner, using various

heuristics. Thus, this approach optimizes the allocation of resources, ensuring timely and accurate extraction of attack techniques from CTI reports.

1.1 Challenges

In this section, we discuss the primary challenges associated with constructing an automated tool for extracting and mapping attack techniques from Cyber Threat Intelligence (CTI) reports to ATT&CK framework.

- **Extracting attack techniques from noisy, unstructured CTI reports.** Structured Cyber Threat Intelligence (CTI) reports are essential for efficient automation and information aggregation across diverse sources. However, creating these reports is labor-intensive, leading many security firms like Avast, McAfee, and Kaspersky to present their public threat reports in unstructured formats. These reports, while informative, are often lengthy, written in natural language, and relevant information often becomes obscured by extraneous text. Additionally, inherent ambiguity in the text further complicates the extraction of relevant information. Traditional rule-based systems such as regular expressions, heuristics, and dependency parsing [13,16] are inadequate for this task. Hence, despite the widespread availability of these reports, the extraction of pertinent information remains a challenging task, necessitating the adoption of more sophisticated machine-learning-based approaches, such as Large Language Models (LLMs) [3].
- **Annotation is expensive and time-consuming.** A major hurdle in utilizing LLMs for real-world applications is the need for vast amounts of labeled data specific to the task at hand. Unfortunately, in our domain, creating a comprehensive dataset mapping CTI reports to all ATT&CK techniques is a significant challenge. While MITRE's annotated dataset used in TRAM [23] offers a valuable starting point, it only covers a limited portion – 50 out of the 625 existing techniques. Even for these 50 techniques, acquiring the 11,000 instance used in fine-tuning an LLM demanded extensive human effort. Scaling this annotation process to encompass the remaining 575 techniques is simply impractical. Security experts' time is precious, and the ever-evolving threat landscape constantly adds new attack techniques, rendering large-scale manual annotation inefficient and unsustainable. This is why we propose a method that achieves performance comparable to TRAM, but with a significantly reduced requirement for annotated training data.

1.2 Contribution

We demonstrate the effectiveness of Active Learning as an intelligent approach that optimizes the annotation process for mapping CTI (Cyber Threat Intelligence) reports to the ATT&CK framework. Our approach strategically selects the most informative instances from CTI reports for human annotation, significantly reducing annotation efforts while achieving performance comparable to

models trained on large datasets. This translates to **reduced costs** for organizations, **improved efficiency** for security teams, and **greater scalability** as the ATT&CK framework and threat landscape evolve. We provide extensive analysis and provide empirical projections on the annotation-savings that will be made possible for analysts when using the ALERT framework. We also explore additional data augmentation techniques to evaluate its effectiveness. Our contributions include:

1. Development of a novel framework, **ALERT** (**A**ctive **L**earning **E**nhanced **R**obust **T**hreat-Mapper) that leverages Active Learning in conjunction with a Large Language Model for resource-efficient extraction of ATT&CK techniques from CTI reports.
2. Conducting an extensive empirical study to compare the performance of different active learning strategies and evaluating the effectiveness of data augmentation to determine the optimal strategy for our task.
3. Providing projections on how the ALERT framework can reduce analysts' effort by intelligently annotating unlabeled ATT&CK techniques in a real-world dataset.

Furthermore, we are open-sourcing our code on Github (`https://github.com/space-urchin/ALERT/tree/main`) for further research efforts in this direction and for the benefit of the security community.

2 Preliminaries

2.1 Cyber Threat Intelligence Reports

Combating the ever-evolving threat landscape demands a collaborative approach. Public Cyber Threat Intelligence (CTI) reports [28] serve as an effective tool to counter cyber threats. CTI reports equip security professionals with actionable information, which may include Indicators of Compromise (IoCs) such as malicious IP addresses and domain names. Additionally, they may offer in-depth explorations that provide invaluable insights into the motivations, tactics, techniques, and procedures (TTPs) employed by cyber adversaries, and outline mitigation strategies and best practices for defense. Regularly published by security vendors like Trustwave, SonicWall, and CrowdStrike, CTI reports empower organizations to proactively identify and defend against emerging threats. Figure 1 is an excerpt from a CTI report, "NotPetya Technical Analysis," [7] published by CrowdStrike which describes an attack technique from the ATT&CK framework.

2.2 ATT&CK Framework

ATT&CK [22] (Adversarial Tactics, Techniques, and Common Knowledge) is a framework designed to organize and categorize adversary behavior and strategies observed in real-world cyber attacks. It classifies various tactics, techniques, and

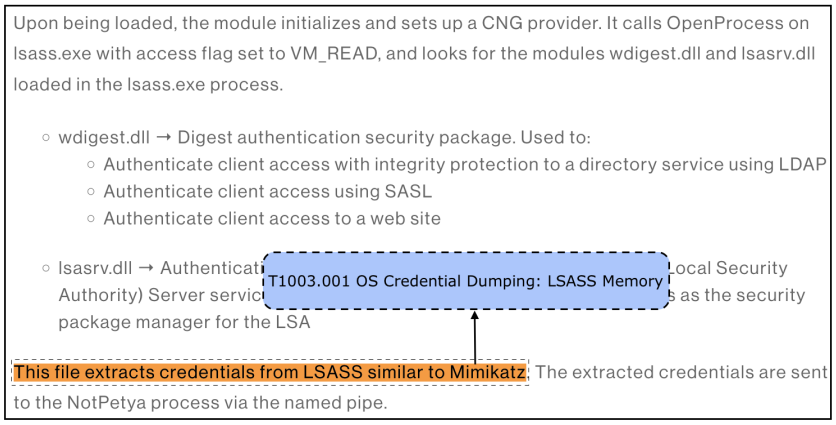

Fig. 1. Excerpt from: NotPetya Technical Analysis – A Triple Threat: File Encryption, MFT Encryption, Credential Theft. The annotation illustrates the mapping of the highlighted line to an attack technique (T1003.001) in the MITRE ATT&CK framework.

sub-techniques employed by threat actors, offering insights into their methodologies. Tactics denote overarching strategies utilized by threat actors to achieve their objective, while techniques are the specific methods or tools they employ to execute these tactics. Techniques can further be broken down into sub-techniques. The ATT&CK matrix organizes these elements, encompassing 14 tactics, 201 techniques, and 424 sub-techniques. In this work, when we refer to techniques, we collectively refer to all 625 techniques and sub-techniques. In Figure 2, we see the hierarchical view of the attack technique referenced in Figure 1. Specifically, the sub-technique T1003.001 is categorized under the technique T1003: Credential Dumping, which falls within the broader tactic TA0006: Credential Access.

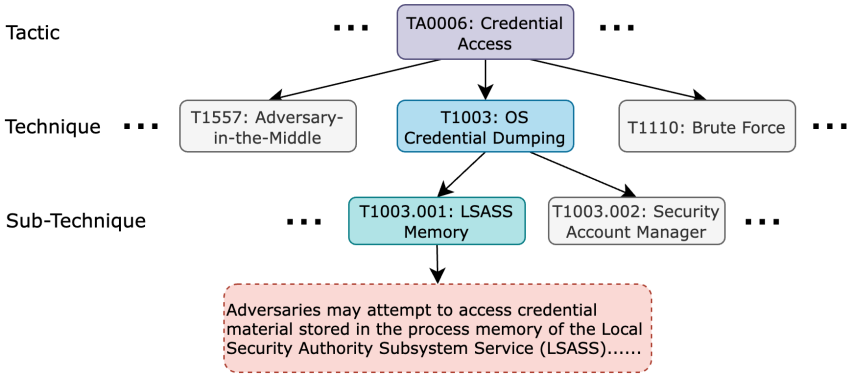

Fig. 2. Hierarchical view of the attack technique T1003.001 from the ATT&CK matrix

2.3 Active Learning

We will now outline the problem setting for pool-based Active Learning (AL) [30] described in Algorithm 1. Let $D^{\mathcal{U}}$ represent the pool of unlabeled data, and $D^{labeled}$ denote the set of initially labeled instances. The objective is to iteratively choose a subset $D^{query} \subset D^{\mathcal{U}}$ for annotation by an oracle (in our context, a security expert). The learner is typically allocated a budget $\mathcal{B}$ which limits the maximum number of labels it can query. Samples are typically selected by some measure of the informativeness of the unlabeled points with respect to the current model, utilizing an acquisition function that employs an active learning strategy. The query strategy can therefore be defined as follows:

$$D^{query} = \arg \max_{D \subset D^{\mathcal{U}}} \mathcal{I}(D, f) \tag{1}$$

where D represents a subset of $D^{\mathcal{U}}$, f is the model trained on the currently labeled data $D^{labeled}$, and $\mathcal{I}(D, f)$ is a measure of informativeness.

Algorithm 1 Pool-based Active Learning

1: **while** $|D^{labeled}| < \mathcal{B}$ **do**
2: Train the model on $D^{labeled}$.
3: Evaluate informativeness of each data point in $D^{\mathcal{U}}$ using an acquisition function.
4: Select D^{query} based on the most informative instances.
5: Request labels for D^{query} from the oracle.
6: Update $D^{labeled}$ with the newly acquired labels.
7: **end while**

We explore strategies spanning 3 state-of-the-art areas in Active Learning: Uncertainty Sampling, Expected Model Change, and Diversity Sampling.

Uncertainty Sampling selects instances for which the model is least certain according to a decision rule [14]. In multiclass scenarios, uncertainty can be defined in various ways. We investigate 4 uncertainty sampling based strategies as described below:

- **Top Confidence**: Let $P(y|x)$ represent the model's predicted probability of label y given the instance x. The instance with the smallest probability for its top predicted label is chosen. This can be expressed as:

$$\operatorname*{argmin}_x \max_y (P(y|x)) \tag{2}$$

- **Margin Sampling**: The margin is defined as the difference between the highest and second-highest predicted probabilities. In this strategy, we select the instance with the smallest margin [27], defined as:

$$\arg\min_x P(y_1|x) - P(y_2|x) \tag{3}$$

where where y_1 and y_2 are the first and second most probable class labels under the model.

- **Maximum entropy**: It considers the entropy [31] of the label distribution, which reflects the uncertainty of the model's prediction for an instance. Higher values of entropy indicate greater uncertainty in the probability distribution. It is defined as:

$$\underset{x}{\operatorname{argmax}} - \sum_{y \in Y} P(y|x) \log(P(y|x)) \tag{4}$$

- **Monte Carlo Dropout**: While dropout is typically applied during training, Gal et al. [10] proposed a method for uncertainty estimation where dropout is applied at test time. For each inference cycle, dropout is applied with a different dropout mask. Each forward pass produces a prediction, so over a number of inference cycles, MC (Monte Carlo) Dropout provides a distribution of predictions rather than a single point estimate. Instance uncertainty is calculated using the average of these predictions using an acquisition function such as max entropy.

Expected Model Change is an approach that aims to select examples expected to induce the most significant changes in the model. For models where gradients can be computed (such as neural networks), one potential method is to select the instance with the highest **expected gradient length (EGL)**, where the expectation is calculated over the probabilities assigned by the model to the labels. This strategy selects instances with the largest expected gradient norm, as they are expected to have a large influence on the model [12,30]. The decision rule is defined as:

$$\underset{x}{\operatorname{argmax}} \sum_{y \in Y} P(y|x) \left\| \nabla l_\theta(\{x, y\}) \right\| \tag{5}$$

where ∇l_θ is the gradient of the objective function l with respect to the model parameters θ, and $\|\nabla l_\theta(\{x, y\})\|$ is the Euclidean norm of the gradient vector for instance {x, y}.

Diversity Sampling The goal of this approach is to improve the model's understanding by actively seeking out diverse instances that can provide new information independently of other labeled samples. One way of doing this is to define active learning as a **Core-Set** selection problem [29]. This problem involves selecting a subset from a fully labeled dataset such that a model trained on this subset performs as closely as possible to the model trained on the entire dataset. This is equivalent to a k-Center problem which is solved by a greedy approximation, called **Greedy Core-Set**, as shown in Algorithm 2.

Algorithm 2 Greedy Core-Set

1: Assign each point in $D^{labeled}$ as a cluster center.
2: **while** $|D^{labeled}| < \mathcal{B}$ **do**
3: Calculate pairwise distance between each data point in $D^{\mathcal{U}}$ and its closest cluster center.
4: Select the point c^{new} in $D^{\mathcal{U}}$ that is farthest from its cluster center.
5: Assign c^{new} as a new cluster center.
6: Add c^{new} to D^{query}.
7: **end while**

3 Proposed Approach

We propose ALERT, which utilizes an active learning pipeline for fine-tuning our learner for mapping text to ATT&CK. This is illustrated in Figure 3. In this section, we discuss the components in detail.

Learner Mapping an instance from a CTI report to an attack technique in ATT&CK is a sequence classification task suited for Large Language Models, particularly encoder-only architectures such as BERT [8]. Variants of BERT, further fine-tuned on additional datasets [18,5] have also demonstrated strong performance in this task. For our learner, we opt for SciBERT [5], a pre-trained BERT-based model specifically trained on scientific data. SciBERT has shown robust performance in computer science tasks, making it a suitable choice for our purpose. Notably, it is also the model of choice for MITRE's TRAM [25] for their mapping task, facilitating direct comparisons.

Pipeline We employ pool-based active learning with a batch mode for our experiments. To warm-start our fine-tuning process, we randomly select 1% of the training pool to create our initial labeled set, $D^{labeled}$. In each annotation cycle, we fine-tune our SciBERT model on the labeled dataset, $D^{labeled}$, for 10 epochs. Subsequently, we evaluate the remaining unlabeled set, $D^{\mathcal{U}}$, using the fine-tuned model. To select an informative set of unlabeled samples D^{query}, we apply one of the acquisition functions described in Section 2.3.

Since we use deep learning based transformer models like SciBERT, training requires significant computational resources, and thus retraining the model after every new data point addition is highly impractical. To mitigate this, a batch-mode approach is often adopted [9,11], where the model queries for a set of points, instead of a single one, at each iteration. We thus adopt the same batch-mode setting.

In each cycle, we select 10 unlabeled instances for annotation (i.e., $|D^{query}| = 10$), a popular and effective choice in batch-mode AL [26,19]. These unlabeled samples are then labeled and added to $D^{labeled}$, and the process repeats until the budget, $\mathcal{B}$ is exhausted. Lastly, we evaluate the final model on a test set.

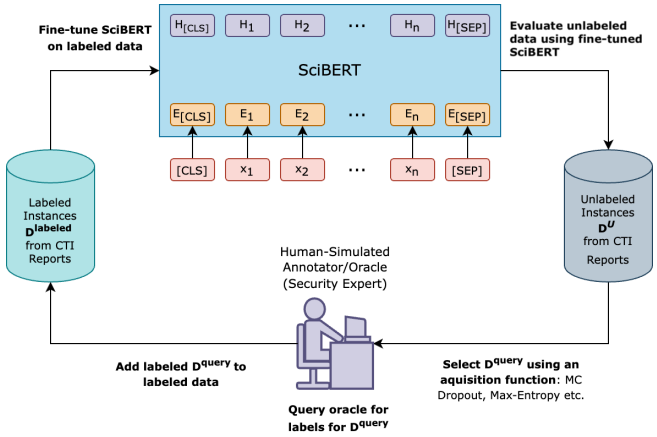

Fig. 3. Pool-based Active Learning Pipeline in ALERT

3.1 Active Learning Strategies

We compared 6 different AL strategies - Top Confidence, Maximum Entropy, Margin Sampling, Monte Carlo Dropout, Approximated Gradient Length and Core-Set.

In **Top Confidence**, **Maximum Entropy**, and **Margin Sampling**, we utilize the logit outputs from the SciBERT model. The logits scores in SciBERT are generated by applying a classifier to the [CLS] (classification) output. The [CLS] vector represents the aggregated sequence representation for classification tasks, and the classifier computes the logits scores based on this representation. By applying softmax to the logit scores, we obtain the probability distribution $P(y|x)$. We then apply the corresponding formula defined in Section 2.3 to select the instances.

In **Monte Carlo Dropout**, we employ a method similar to the one described in [9]. We introduce dropout via a perceptron, where the input to the perceptron is the CLS vectors from the fine-tuned SciBERT model. The dropout rate is set to 0.9. Instance uncertainty is then calculated by averaging the softmax probabilities over 10 inference cycles, using the max-entropy acquisition function. This is illustrated in Figure 4.

In **Approximated Gradient Length**, we adopt an idea similar to that of EGL defined in Section 2.3. However, computing the expected gradient over all 50 class labels is computationally expensive. Instead, in our approach, for faster computation, we compute the gradient as if the model's prediction on the unlabeled instance were the true label, similar to the idea presented in [4].

In **Core-Set**, we apply the greedy approach outlined in Algorithm 2. We choose the Euclidean distance as the pairwise distance metric and it is calculated in the CLS vector space.

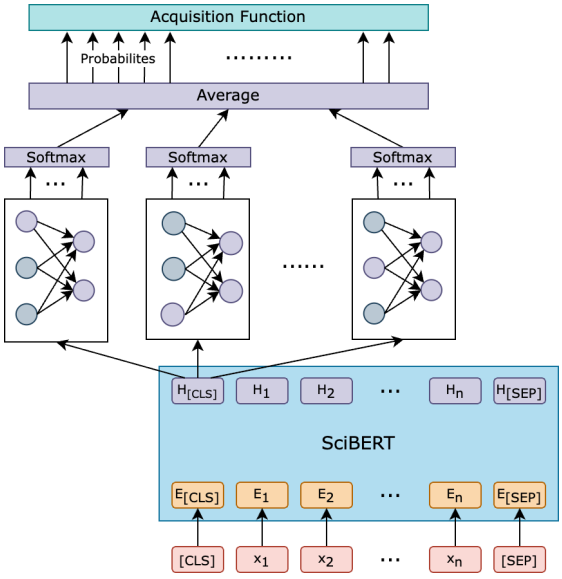

Fig. 4. Pipeline for calculating instance uncertainty using Monte Carlo Dropout

4 Experiments and Results

4.1 Dataset

To evaluate our approach, we selected MITRE's corpus [23], which to our knowledge is the only large scale dataset mapping CTI reports to ATT&CK techniques. The corpus consists of 11,130 sentences and phrases extracted from CTI reports describing ATT&CK techniques. Although ATT&CK comprises of 625 ATT&CK techniques, MITRE focused their efforts on annotating a subset of the 50 most common techniques [21] for the dataset. We split the dataset into training and test sets using an 80/20 ratio. Specifically, we allocate 8,904 instances to the training set, while the remaining 2,226 instances form our test set.

We evaluate our model on 4 different budgets, based on annotation cycles. As an example, if we have 50 annotation cycles with $|D^{query}| = 10$, this gives 500 samples that are actively acquired. With an initial warm start of 1% data (89 samples), this gives a total of 589 samples when using 50 annotation cycles. The budgets are as follows:

1. $\mathcal{B}_1$: 50 annotation cycles, corresponding to 589 samples, approximately 7% of the training pool
2. $\mathcal{B}_2$: 100 annotation cycles, corresponding to 1089 samples, approximately 12% of the training pool
3. $\mathcal{B}_3$: 150 annotation cycles, corresponding to 1589 samples, approximately 18% of the training pool
4. $\mathcal{B}_4$: 200 annotation cycles, corresponding to 2089 samples, approximately 23% of the training pool.

4.2 Comparative Study of AL Strategies.

Our findings in Table 1 confirm the significant advantage of AL strategies compared to random selection. Random Selection serves as a control experiment for Active Learning (AL). During each annotation cycle, instead of selecting instances using an AL strategy, we randomly select 10 instances. This allows us to assess the effectiveness of AL in improving the overall performance. All AL methods outperform the random baseline, highlighting the importance of selecting informative data points for effective model training.

When we examine Figure 5 which is a plot of the F1 scores (y-axis) of different AL strategies across different annotation budgets (x-axis), three AL strategies stand out: Top Confidence, Monte Carlo Dropout, and Core-Set. Top Confidence offers the most efficient choice for smaller budgets ($\mathcal{B}_1$ and $\mathcal{B}_2$) – it achieves the best performance while boasting the lowest computational cost as seen in Table 2. This efficiency stems from its reliance on a simple minimum operation on the model's confidence scores. For the largest budget ($\mathcal{B}_4$), Core-Set delivers the highest accuracy, closely mirroring the performance of fine-tuning on the entire training pool. However, this enhanced accuracy comes at the cost of computational speed, as it involves calculating pairwise Euclidean distances between all labeled and unlabeled data points. Finally, while Monte Carlo Dropout demonstrates effectiveness for budget $\mathcal{B}_3$, it's the most computationally expensive option. This method requires multiple model inferences per annotation cycle, leading to significantly longer runtimes.

In conclusion, when computational resources are limited, Top Confidence stands out for its efficiency. For the strategy that most closely approaches the best performance at our annotation budget limit, Core-Set offers the best results.

4.3 Performance against Baselines

In our work, we compare our proposed approach against three baseline methods that leverage Large Language Models (LLMs) for ATT&CK mapping: TRAM, TTPClassifier and SMET.

TRAM (Threat ATT&CK Mapper) [25], developed by MITRE, utilizes SciBERT for sequence classification. We fine-tune SciBERT on the entire training pool and evaluate its performance on the test set.

TTPCLassifier [3] uses pre-trained Sentence-BERT (SBERT) [24] to generate embeddings and uses cosine similarity between embeddings to map text to ATT&CK. SMET [1], on the other hand, employs SBERT fine-tuned on 38,396 pairs of attack vectors extracted from ATT&CK technique descriptions and procedure examples to generate embeddings used to train a multinomial logistic regression model. It's important to note that both these approaches were originally designed to map only to attack techniques, not sub-techniques. Since our test set includes both, we consider the prediction of these methods correct during evaluation even if it maps to the higher-level attack technique instead of the sub-technique (e.g. for Figure 1, mapping to T1003 instead of T1003.001 is considered a correct prediction).

Table 1. Performance Metrics for different Active Learning Strategies

Budget	Model	Precision	Recall	F1
$\mathcal{B}_1$ **7%** of data 589 samples	**ALERT w/ Top Confidence**	**0.79**	**0.75**	**0.75**
	ALERT w/ Maximum Entropy	0.73	0.70	0.70
	ALERT w/ Margin Sampling	0.74	0.73	0.71
	ALERT w/ Monte Carlo Dropout	0.76	0.74	0.74
	ALERT w/ Approximated Gradient Length	0.74	0.72	0.71
	ALERT w/ Core-Set	0.75	0.73	0.72
	ALERT w/ Random Selection	0.16	0.15	0.14
$\mathcal{B}_2$ **12%** of data 1089 samples	**ALERT w/ Top Confidence**	**0.81**	**0.80**	**0.80**
	ALERT w/ Maximum Entropy	0.78	0.77	0.77
	ALERT w/ Margin Sampling	0.79	0.77	0.76
	ALERT w/ Monte Carlo Dropout	0.80	0.79	0.79
	ALERT w/ Approximated Gradient Length	0.79	0.77	0.77
	ALERT w/ Core-Set	0.80	0.79	0.79
	ALERT w/ Random Selection	0.10	0.09	0.09
$\mathcal{B}_3$ **18%** of data 1589 samples	ALERT w/ Top Confidence	0.84	0.82	0.83
	ALERT w/ Maximum Entropy	0.84	0.83	0.83
	ALERT w/ Margin Sampling	0.79	0.78	0.78
	ALERT w/ Monte Carlo Dropout	**0.85**	**0.84**	**0.84**
	ALERT w/ Approximated Gradient Length	0.82	0.81	0.81
	ALERT w/ Core-Set	0.84	0.83	0.83
	ALERT w/ Random Selection	0.06	0.06	0.06
$\mathcal{B}_4$ **23%** of data 2089 samples	ALERT w/ Top Confidence	0.85	0.84	0.84
	ALERT w/ Maximum Entropy	0.85	0.84	0.84
	ALERT w/ Margin Sampling	0.79	0.79	0.79
	ALERT w/ Monte Carlo Dropout	0.85	0.84	0.84
	ALERT w/ Approximated Gradient Length	0.82	0.81	0.81
	ALERT w/ Core-Set	**0.86**	**0.85**	**0.85**
	ALERT w/ Random Selection	0.07	0.06	0.06

The results are seen in Table 3. We observe that simply using a pre-trained SBERT as in the case of TTPClassifier achieved poor performance - an F1 score of only 0.46. However, fine-tuning SBERT on a domain-specific dataset such as in the case of SMET achieved a better performance. Nonetheless, our approach still significantly outperformed SMET eventhough it was trained on a larger dataset, achieving an F1 score of 0.85 compared to SMET's score of 0.69. Moreover, our approach was fine-tuned on a subset of dataset which was 77% smaller than that of TRAM yet achieved comparable performance to TRAM (F1 score of 0.88).

4.4 Efficacy of Augmentation

In scenarios where training data is scarce, augmentation helps by artificially expanding the dataset. Easy Data Augmentation (EDA) [32] provides a simple yet effective way to augment training data, improving performance on low-resource tasks. EDA involves four simple processes: synonym replacement, random inser-

Table 2. Runtime of three top performing AL strategies for 50 annotation cycles - Top Confidence, Monte Carlo Dropout, Core-Set

	ALERT w/ Top Confidence	ALERT w/ Monte Carlo Dropout	ALERT w/ Core-Set
Time(s)	11098	13210	11815

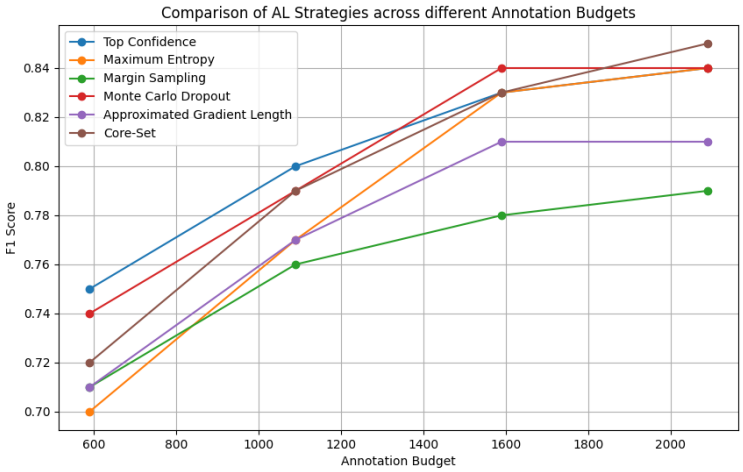

Fig. 5. Comparison of F1 scores obtained by different AL Strategies at different annotation budgets

tion, random deletion, and random swap. In our experiments, we further investigate whether augmentation techniques like EDA enhances model performance in conjunction with active learning strategies.

The data augmentation experiment, as reported in Table 4, further emphasizes the importance of selecting informative data for annotation. We conducted this experiment with the top-performing AL strategies for each annotation budget. After selecting samples using the respective AL strategy, for each sample in the labeled set, we generated four new synthetic samples using the four processes. The models were then fine-tuned on significantly larger augmented datasets. However, we observed no improvement compared to the results in Table 1. Since the new synthetic data were derived from the original samples selected by the AL strategy, they provided no new information to the model. This finding highlights that augmentation on top of AL strategies to is not useful and instead efforts should be directed toward curating a diverse dataset to improve generalization.

We can therefore conclude that for our task, simply increasing the size of the fine-tuning corpus is not as effective as selecting diverse samples that provide more informative data for the learner.

Table 3. Performance Metrics against Baselines

Model	Fine-tuning Corpus	Precision	Recall	F1
ALERT w/ Core-Set	**2089 instances** of labeled text from CTI Reports	0.86	0.85	0.85
TRAM (SciBERT)	**8904 instances** of labeled text from CTI Reports	0.88	0.88	0.88
SMET (Fine-tuned SBERT)	**38,396 instances** of attack vectors extracted from ATT&CK technique description and procedure examples	0.78	0.63	0.69
TTPClassifier (Pre-trained SBERT)	Zero-shot	0.72	0.42	0.46

Table 4. Performance of AL strategies with Augmentation

AL Strategy	Fine-tuning Corpus (total instances: real / synthetic)	Precision	Recall	F1
Top Confidence + EDA	**2945 instances:** 589 / 2356	0.75	0.73	0.73
Top Confidence + EDA	**5445 instances:** 1089 / 4356	0.82	0.80	0.80
Monte Carlo Dropout + EDA	**7945 inst ances:** 1589 / 6356	0.82	0.81	0.81
Core-set + EDA	**10445 instances:** 2089 / 8356	0.83	0.82	0.82

4.5 Case Study on ALERT's impact

We perform a case study to highlight the impact of ALERT's efficiency. These experiments are conducted using the ALERT w/ Core-set model. MITRE's dataset containing CTI reports currently contains annotations of 50 different ATT&CK techniques. However, in total, we know that there are 625 techniques as outlined in Section 2.2. Thus, the remaining 575 techniques currently remain unlabeled, and we would like to provide approximations for the amount of annotation effort we can save by using ALERT.

To do so, we create subsets of MITRE's labeled dataset (with 50 techniques), by removing all data instances belonging to randomly chosen techniques. In this way, we create subsets with reduced numbers of techniques. Specifically, we create 4 subsets containing 2, 5, 10, and 25 techniques respectively. Along with the original dataset with 50 techniques, this gives 5 *scenarios* ranging from datasets with 2 techniques (binary classification), to 50. Now, for each of these scenarios, we use TRAM (a baseline model without AL optimizations), to calculate the F-1 scores it achieves. Next, we use ALERT to identify the minimum number of annotations necessary to achieve an F-1 score which is within $\triangle f$ of TRAM's F-1 scores, where $\triangle f$ is a very small value. From Table 3, we noted that ALERT's F1 score is only 0.03 short of TRAM while using 6815 samples fewer (over 77% fewer) samples. We thus set $\triangle f = 0.03$ as our target, and check how many training samples ALERT requires to achieve an F-1 score within $\triangle f$ of TRAM.

After identifying the required number of samples in the 5 different scenarios, we plot these points, shown in Figure 6(a). Finally, we use a simple curve-fitting

approach based on least-squares-optimization [33], to extrapolate the number of training samples that experts would need to annotate to classify larger numbers of ATT&CK techniques, all the way up to 625 (the total number of techniques found in the ATT&CK framework). Figure 6(b) visualizes these projected values. Thus, for 625 techniques, we project that the ALERT framework would require 27,418 samples instead of TRAM's 115,682. In other words, ALERT would require a dataset over 4 times smaller to achieve comparable performance, making this much more feasible for annotators and security analysts. Since only 50 techniques are currently labeled in the MITRE dataset, the practical utility is clear.

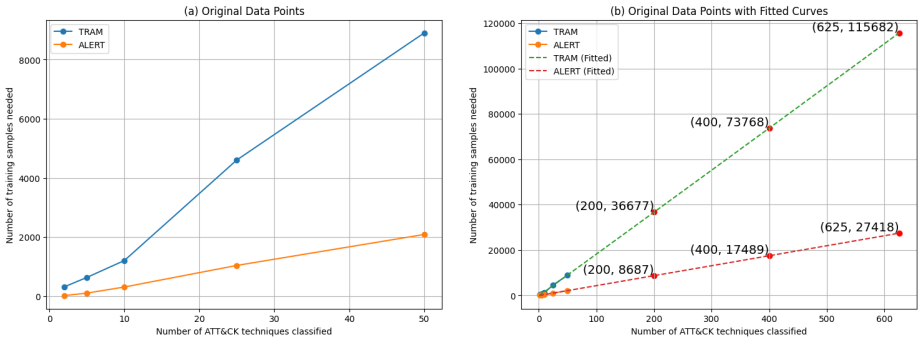

Fig. 6. (a) Relationship between the number of different ATT&CK techniques classified against the number of training samples (b) An extrapolation from our real data points to project the number of of samples needed for all ATT&CK techniques

5 Related Work

To tackle the complexity of CTI reports, recent approaches have used Large Language Models (LLMs) [25,3,1] to extract and map text to ATT&CK techniques.

TTPClassifier [3] computes embeddings for extracted text from CTI reports using a pre-trained Sentence BERT (SBERT) [24]. It also computes embeddings for the title and description of each ATT&CK technique. It then uses a similarity metric (weighted cosine similarity) to match the text with the title and description to map to the most relevant technique. However, this zero-shot approach is not very effective, and their method exhibits high false positive rates.

SMET [1] fine-tunes the SBERT model on 38,396 pairs of attack vectors extracted from each ATT&CK technique description and procedure example to generate a semantically meaningful embedding of attack vectors. Finally, SMET uses a logistic regression model trained on these embeddings to estimate the probability of an attack vector belonging to each ATT&CK technique and rank techniques based on the estimated probability.

MITRE developed Threat Report ATT&CK Mapper (TRAM) [23] to automate the mapping of CTI reports to ATT&CK. They built a training corpus of 11,300 sentences and phrases extracted from CTI reports that map to the 50 most common ATT&CK techniques. This corpus was then used to fine-tune SciBERT [5] for sequence classification, to map the sentences/phrases to ATT&CK.

While previous work focused on classification metrics, to our knowledge, ours is the first to address the *efficiency* of the process, by utilizing Active Learning to attain comparable and consistent performance while using a much smaller yet highly informative data subset. Efficiency is key in this domain. In fact, the main challenge of MITRE's work with TRAM has been the annotation process, and hence, they were only able to annotate 50 out of the total 625 techniques.

Active learning has been a widely adopted solution for data-driven machine learning [30], especially when annotation is expensive and time-consuming. This idea has been proven effective in various domains, including image classification [29], speech recognition [12], and natural language processing [9]. There has been limited work using active learning in the cybersecurity domain, some focusing on LSTMs [15]. However, leveraging active learning with LLMs in cybersecurity, particularly for the extraction of ATT&CK techniques, has yet to be explored.

Our work bridges this gap by conducting an extensive study on using active learning strategies with an LLM in this domain. We propose a solution that reduces the annotation burden for efficient extraction and mapping of ATT&CK techniques from CTI reports.

6 Conclusion and Future Work

Our evaluations demonstrate that ALERT significantly improves the efficiency of ATT&CK technique extraction from CTI reports by reducing annotation costs. Thus, ALERT will be a valuable tool for analysts going forward, as they set out to efficiently annotate unlabeled threat reports in the future. For future work, we aim to go beyond extracting attack techniques. We aim to identify *relationships* between different techniques in CTI reports. For instance, if reports describe a chronological order of techniques in an APT kill chain, detecting a technique at a specific stage may enable proactive defenses against later attacks. Finally, we aim to leverage out-of-distribution detection methods to find novel attack techniques not yet present in the ATT&CK framework.

Acknowledgements. The research reported herein was supported in part by NIST Award # 60NANB23D007, NSF awards DMS-1737978, DGE-2039542, OAC-1828467, OAC-1931541, and DGE-1906630, ONR awards N00014-17-1-2995 and N00014-20-1-2738, and the National Center for Transportation Cybersecurity and Resiliency (TraCR).

Disclaimer. Certain equipment, instruments, software, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

References

1. Abdeen, B., Al-Shaer, E., Singhal, A., Khan, L., Hamlen, K.: Smet: Semantic mapping of cve to att&ck and its application to cybersecurity. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 243–260. Springer (2023)
2. Akbanov, M., Vassilakis, V.G., Logothetis, M.D.: Wannacry ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms. Journal of Telecommunications and Information Technology (1), 113–124 (2019)
3. Alam, M.T., Bhusal, D., Park, Y., Rastogi, N.: Looking beyond iocs: Automatically extracting attack patterns from external cti. In: Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses. pp. 92–108 (2023)
4. Ash, J.T., Zhang, C., Krishnamurthy, A., Langford, J., Agarwal, A.: Deep batch active learning by diverse, uncertain gradient lower bounds. arXiv preprint arXiv:1906.03671 (2019)
5. Beltagy, I., Lo, K., Cohan, A.: Scibert: A pretrained language model for scientific text. arXiv preprint arXiv:1903.10676 (2019)
6. Bianco, D.: The pyramid of pain (2013), https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html
7. CrowdStrike: Notpetya technical analysis – a triple threat: File encryption, mft encryption, credential theft. https://www.crowdstrike.com/blog/petrwrap-ransomware-technical-analysis-triple-threat-file-encryption-mft-encryption-credential-theft/
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Dor, L.E., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y., Slonim, N.: Active learning for bert: an empirical study. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP). pp. 7949–7962 (2020)
10. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: international conference on machine learning. pp. 1050–1059. PMLR (2016)
11. Gentile, C., Wang, Z., Zhang, T.: Achieving minimax rates in pool-based batch active learning. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 7339–7367. PMLR (17–23 Jul 2022), https://proceedings.mlr.press/v162/gentile22a.html
12. Huang, J., Child, R., Rao, V., Liu, H., Satheesh, S., Coates, A.: Active learning for speech recognition: the power of gradients. arXiv preprint arXiv:1612.03226 (2016)
13. Husari, G., Al-Shaer, E., Ahmed, M., Chu, B., Niu, X.: Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In: Proceedings of the 33rd annual computer security applications conference. pp. 103–115 (2017)
14. Lewis, D.D.: A sequential algorithm for training text classifiers: Corrigendum and additional data. In: Acm Sigir Forum. vol. 29, pp. 13–19. ACM New York, NY, USA (1995)
15. Li, T., Hu, Y., Ju, A., Hu, Z.: Adversarial active learning for named entity recognition in cybersecurity. Computers, Materials & Continua **66**(1) (2021)

16. Li, Z., Zeng, J., Chen, Y., Liang, Z.: Attackg: Constructing technique knowledge graph from cyber threat intelligence reports. In: European Symposium on Research in Computer Security. pp. 589–609. Springer (2022)
17. Liao, X., Yuan, K., Wang, X., Li, Z., Xing, L., Beyah, R.: Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 755–766 (2016)
18. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
19. Lourentzou, I., Gruhl, D., Welch, S.: Exploring the efficiency of batch active learning for human-in-the-loop relation extraction. In: Companion proceedings of the the web conference 2018. pp. 1131–1138 (2018)
20. MITRE: D3fend (2023), https://d3fend.mitre.org/
21. MITRE: Large language models: Architecture. https://github.com/center-for-threat-informed-defense/tram/wiki/Large-Language-Models#architecture (2023)
22. MITRE: Mitre att&ck framework (2023), https://attack.mitre.org
23. MITRE: Threat report attck mapper (tram) (2023), https://github.com/center-for-threat-informed-defense/tram/
24. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019)
25. Ross, J., Lasky, J.: Our tram large language model automates ttp identification in cti reports. MITRE-Engenuity (Aug 2023), https://medium.com/mitre-engenuity/our-tram-large-language-model-automates-ttp-identification-in-cti-reports-5bc0a30d4567
26. Sahan, M., Smidl, V., Marik, R.: Batch active learning for text classification and sentiment analysis. In: Proceedings of the 2022 3rd International Conference on Control, Robotics and Intelligent System. pp. 111–116 (2022)
27. Scheffer, T., Decomain, C., Wrobel, S.: Active hidden markov models for information extraction. In: International symposium on intelligent data analysis. pp. 309–318. Springer (2001)
28. Schlette, D., Caselli, M., Pernul, G.: A comparative study on cyber threat intelligence: The security incident response perspective. IEEE Communications Surveys & Tutorials **23**(4), 2525–2556 (2021)
29. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489 (2017)
30. Settles, B.: Active learning literature survey (computer sciences technical report 1648) university of wisconsin-madison. Madison, WI, USA: Jan (2009)
31. Shannon, C.E.: A mathematical theory of communication. The Bell system technical journal **27**(3), 379–423 (1948)
32. Wei, J., Zou, K.: Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196 (2019)
33. Weisstein, E.W.: Least squares fitting. https://mathworld. wolfram. com/ (2002)
34. Zhu, Z., Dumitras, T.: Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In: 2018 IEEE European symposium on security and privacy (EuroS&P). pp. 458–472. IEEE (2018)