2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020622

VWC-BERT: Scaling Vulnerability–Weakness–Exploit Mapping on Modern AI Accelerators

Siddhartha Shankar Das*, Mahantesh Halappanavar†, Antonino Tumeo†,
Edoardo Serra‡, Alex Pothen*, Ehab Al-Shaer§,
*Purdue University, †Pacific Northwest National Laboratory, ‡Boise State University and Pacific Northwest National Laboratory,
§Carnegie Mellon University
*{das90, apothen}@purdue.edu, †{hala,Antonino.Tumeo}@pnnl.gov, ‡edoardoserra@boisestate.edu, §ehab@cmu.edu,

***Abstract*—Defending cybersystems needs accurate mapping of software and hardware vulnerabilities to generalized descriptions of weaknesses, and weaknesses to exploits. These mappings enable cyber defenders to build plans for effective defense and assessment of potential risks to a cybersystem. With close to 200k vulnerabilities, manual mapping is not a feasible option. However, automated mapping is challenging due to limited training data, computational intractability, and limitations in computational natural language processing. Tools based on breakthroughs in Transformer-based language models have been demonstrated to classify vulnerabilities with high accuracy. We make three key contributions in this paper: (1) We present a new framework, VWC-BERT, that augments the Transformer-based hierarchical multi-class classification framework of Das et al. (V2W-BERT) with the ability to map weaknesses to exploits. (2) We implement VWC-BERT on modern AI accelerator platforms using two data parallel techniques for the pre-training phase and demonstrate nearly linear speedups across NVIDIA accelerator platforms. We observe nearly linear speedups for up to 16 V100 and 8 A100 GPUs, and about 3.4× speedup for A100 relative to V100 GPUs. Enabled by scaling, we also demonstrate higher accuracy using a larger language model, RoBERTa-Large. We show up to 87% accuracy for strict and up to 98% accuracy for relaxed classification. (3) We develop a novel parallel link manager for the link prediction phase and demonstrate up to 21× speedup with 16 V100 GPUs relative to one V100 GPU, and thus reducing the runtime from 2.5 hours to 10 minutes. We believe that generalizability and scalability of VWC-BERT will benefit both the theoretical development and practical deployment of novel cyberdefense solutions and vulnerability classification.**

***Index Terms*—Deep Learning, Cybersecurity, Transformers, Language Models, AI Accelerators**

I. Introduction

We consider the problem of mapping observed vulnerabilities in specific software to a hierarchically constructed dictionary of weaknesses, and mapping of these weaknesses to a dictionary of exploits, in cybersecurity using machine learning [2], [6], [9]. We combine Transformer-based language models [13] and a link prediction framework that uses the Siamese model [1] to map vulnerabilities to weaknesses and weaknesses to exploits. Since these methods are compute intensive, we employ accelerator platforms to obtain scalable performance for large data sets and language models (§II).

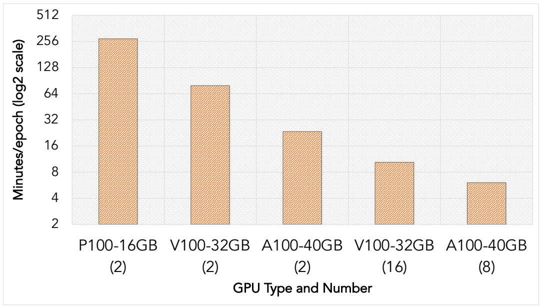


Fig. 1: Pre-training time for BERT-Large across different generations and numbers of artificial intelligence (AI) accelerator platforms (§IV). Y-axis is in log scale.

A *weakness* is an architecture, design, or implementation bug, error, or fault that occurs in cyberproducts (such as software, operating system, or hardware) that allows an unintentional and exploitable behavior of the product. A *vulnerability* is a set of one or more weaknesses in a specific cyberproduct that can be potentially exploited by an attacker for malicious operations. Unpatched vulnerabilities are a leading cause of cybersecurity incidents that lead to significant economic damages to organizations. In particular, newly discovered and unpatched vulnerabilities known as zero-day vulnerabilities are only known to hackers, who can actively exploit them before the vulnerabilities can be patched [10]. A large number of product-specific vulnerabilities are known, and new vulnerabilities are discovered each day by both attackers and defenders. The Common Vulnerabilities and Exposures (CVE) reports list the vulnerabilities in different software products as they are discovered and documented by cybersecurity experts. In contrast, the Common Weakness Enumerations (CWE) provide a blueprint for understanding product flaws and their impact through a hierarchically designed dictionary of product weaknesses. The content of such a dictionary is shared knowledge among security analysts. Since weaknesses are independent of specific software products, the number of weaknesses remains relatively constant and evolves slowly. Similarly, the Common Attack Pattern Enumeration and Classification (CAPEC) provides a database of common attack

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 15,2025 at 09:36:32 UTC from IEEE Xplore. Restrictions apply.

patterns on how weaknesses can be exploited by attackers. CAPEC database is also a complex hierarchy of attack classes with textual, categorical, and referential data.

Automated mapping (classification) of existing and newly discovered vulnerabilities to hierarchically structured weakness enumerations and exploits is an important tool to swiftly understand and mitigate the vulnerabilities. However, there are several challenges for automation, such as semantic gaps in the languages of CVEs, CWEs and CAPECs; non-disjoint hierarchy of CWE classes (multiple paths to the same weaknesses from a lower level of the hierarchy); and lack of sufficient training data, where rare CWE and CAPEC classes have few or no CVEs mapped to them. Recently, Das et al. [3] proposed a novel Transformer-based framework to classify vulnerabilities with high accuracy. In this paper, we extend the work of Das et al. along three important dimensions: (i) Ability to map weaknesses to exploits, thus enhancing the utility of VWC-BERT (detailed in §III) to map vulnerabilities to exploits; (ii) efficiently parallelize VWC-BERT (detailed in §III), targeting AI accelerators (§II); and (iii) enable the use of larger language models for enhanced accuracy (§IV). In particular, VWC-BERT brings together Transformer-based language models [13] and a link prediction framework that uses the Siamese model [1] to embed semantically different text forms in CVEs, CWEs and CAPECs into the same semantic space. The framework effectively addresses the hard problem of few-shot and zero-shot learning for CWE classes with sparse training data.

On the other hand, intense interest in artificial intelligence (AI) has led to a "Cambrian" explosion of specialized accelerators [11]. Transformer-based language models are critically dependent on accelerator technologies to scale with the size of the model (number of parameters) and the corpus (volume of text). We consider Tesla P100, V100, and A100 in multi-GPU configurations, the first in a distributed cluster, the others in "datascale" configurations (DGX-2 with 16 Tesla P100, and DGX A100 with 8 Tesla A100). Both the V100 and A100 GPUs implements functional units specialized for tensor operations. The A100 also introduces specialized number types for machine learning (BFLOAT16 and TF32). Thus, we see superior performance across GPU generations (§IV).

Contributions: The key contributions of this work are:

1) We extend the V2W-BERT framework of Das et al. to include the ability to map weaknesses to exploits, and thus, enable prediction of attack pattern for CWEs and CVEs that currently does not exist in the MITRE framework.
2) We present a scalable framework (VWC-BERT), where we scale both pre-training and link prediction phases in Data-Parallel (DP) and Distributed Data-Parallel (DDP) modes using PyTorch (§III-D), and benchmark performance on AI accelerators.
3) We develop two approaches for scaling link prediction depending on the dataset size, the number of nodes and GPUs in the system: Link Pair Processor (LPP), which scales well with more GPU resources; and Principal Link Manager (PLM), which is work-efficient on small-scale systems.
4) We demonstrate nearly linear speedups for both pre-training and link prediction steps on all the three generations of Nvidia GPUs, and about $3.4\times$ speedup for A100 relative to V100 GPUs. Further with Principal Link Manager (PLM), we implement an effective batching and data control strategy to reduce average training time per epoch from 2.5 hours to 13 minutes.
5) We use both base and large models of BERT [4], RoBERTa [7], and DistilBERT [12]. We observe that RoBERTa-Large provides the best accuracy with 87%-98% top k prediction accuracy, and achieves a 2% gain in the accuracy relative to BERT-Base model.

To the best of our knowledge, this is the first work to scale a Transformer-based framework for mapping CVEs to hierarchically-structured CWE and CAPEC descriptions on novel AI accelerators. As empirically demonstrate that scalability leads to advances in both Transformer and link prediction based approaches for learning, which in turn enables the application of VWC-BERT to solve challenging problems in cybersecurity theory and practice.

II. Overview of AI Accelerators

We consider the P100, V100, and A100 GPUs. The P100 is based on the Pascal architecture, directly derived from Maxwell and, thus, still leveraging a design with wide vector units to implement the single instruction multiple thread (SIMT) model. However, it has a better balanced ratio of 32-bit (FP32) and 64-bit (FP64) floating-point arithmetic-logic units (ALUs), 2:1, 64 and 32 per streaming multiprocessor (SM), respectively, totaling 3584 and 1792 in an entire GPU. It also interfaces to 3D-stacked High-Bandwidth Memory 2 (HBM) rather than GDDR RAM. In our experiments, we use Tesla P100 accelerators with 16 GB of HBM2 (providing a bandwidth of 549 GB/s) installed in a distributed cluster. Each cluster node has 2 Tesla P100 boards, connected through PCI Express 3.0 to 2 Intel Xeon(R) E5-2620 v4 CPUs at 2.10GHz (8 cores each) with 64 GB of DDR4 RAM and an InfiniBand Fourteen Data Rate (FDR) network interface. The V100 is based on the Volta architecture, which implements a novel design of the SMs based on temporal SIMT with a larger on-chip memory, increases the number of FP32 ALUs to 5120 (still half FP64), and integrates an additional 640 functional units named Tensor Cores that perform specialized tensor operations (matrix multiply and accumulate on 4x4 matrices, supporting FP inputs at 16-bit and outputs at 32-bit). We use a DGX-2 system that integrates 16 Tesla V100 (operating at a frequency of 1245 Mhz that can be boosted up to 1380 Mhz) with 32 GB of HBM2 (at 876 MHz, providing more than 900 GB/s of bandwidth), directly interconnected through NVLINK2 (300 GB/s of bandwidth from and to each GPU). The DGX-2 uses 2 Intel Xeon Platinum 8168 at 2.7 GHz with 24-cores each as host processors, paired with 1.5 TB of DDR4 system memory. The A100 is the newest design, based on the Ampere architecture. Ampere integrates 6912 FP32 ALUs (3456 FP64), and 432 third generation Tensor Cores. While

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 15,2025 at 09:36:32 UTC from IEEE Xplore. Restrictions apply.

lower in number, these new Tensor Cores are much more efficient, supporting a variety of specialized types for machine learning (integer and FP) and taking advantage of structured sparsity in the input tensors to accelerate computations. We use a DGX A100 system, which integrates 8 Tesla A100 fully interconnected through NVLINK3 (600 GB/s between any two GPUs). Each GPU hosts 40 GB of HBM2, with a bandwidth up to 1,555 GB/s. There are 2 AMD Rome 7742 host processors (128 cores total) at 2.25 GHz, and 1 TB of DDR4 host memory.

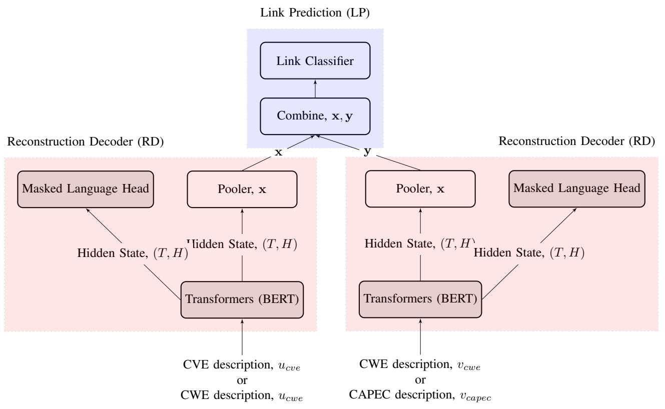


Fig. 2: Architectural overview of the VWC-BERT framework. The Transformer component including the Reconstruction Decoder (RD) is highlighted in red, and the Link Prediction (LP) component is highlighted in blue. Pre-training is done only on the Masked Language Model (highlighted in gray), but the entire framework is utilized for link prediction. The architecture can take CVE-CWE or CWE-CAPEC pairs.

III. Overview of VWC-BERT Framework

We now discuss the general architecture, objective, learning process and parallelization of VWC-BERT. The architecture uses Siamese networks [1] of transformer models to formulate links between CVEs to CWEs, and CWEs to CAPECs. Figure 2 illustrates the general structure of the entire framework. VWC-BERT has two primary components: (i) Link Prediction (LP) to establish the links between a pair of CVE and CWE descriptions, and (ii) Reconstruction Decoder (RD), a masked language head on top of Transformer model to preserve the context.

A. Pre-training of Language Models

The CVEs are expressed in a cyber-security domain-specific language, and the goal of pre-training step is to learn it such that the representation of two similar texts gets mapped into the same semantic space. The Masked Language Model (MLM) is used to learn such contexts with custom Neural Network (NN) layers on top of the Transformer models (highlighted in gray in Figure 2). This custom NN is known as Masked Language Head, and the final layer has output neurons of the size of the vocabulary used by the respective tokenizer of the models. Some of the input tokens are randomly masked during the feedforward step, and the model tries to predict them. Throughout the training process, the model learns the sentence structure in cybersecurity domain (represented by CVE corpus) and avoids focusing on specific keywords. MLM can also utilize 47% of the CVE descriptions that are not labeled, and the self-supervised [15] pre-training mode can benefit from that data. We use multiple transformer models: BERT [4], RoBERTa [7], and DistilBERT [12]). We use the corresponding specific Mask Language Head for each model.

B. Fine-tuning for Link Prediction

In the link prediction step, VWC-BERT takes CVE-CWE or CWE-CAPEC description pairs, and generates individual vector representation for them through the pooler. The representations are combined using appropriate combination operators and passed to the link classifier. By default, the concatenation of $|\mathbf{x} - \mathbf{y}|$ and $\mathbf{x} \times \mathbf{y}$ is used for this work and has been found to be best performing [3]. The link classifier contains custom NN layers to take the combined representation and returns prediction with two-dimensional outputs. This output vector represents the unlink or link prediction confidence. While processing the input descriptions for pooled representation, the same pre-trained Masked Language Head is used to ensure that the learned context remains preserved. For a CVE-CWE description pair, the overall loss to optimize is expressed as:

$$l = \alpha_1 LP_l(cve, cwe) + \alpha_2 RD_l(cve) + \alpha_3 RD_l(cwe), \quad (1)$$

where $LP_l(cve, cwe)$ is the cross-entropy loss of link classifier between a CVE and a CWE. Masked Language Head loss is represented as $RD_l(cve)$ and $RD_l(cwe)$, for the CVE and the CWE, respectively. The parameters, $\alpha_1, \alpha_2, \alpha_3$, are set to 1 by default and can be adapted to emphasize different objectives.

C. Hierarchical Prediction & Training Process of CVE-CWE-CAPEC mapping

The CWEs maintain a hierarchical structure, and during prediction for a single CVE, we compute the link confidence value of CWEs from the first layers of the hierarchy. Best k confident predictions are selected, and the process continues with their children iteratively until we find a leaf node. Note that masking of tokens and Masked Language Head is omitted during test time to utilize the entire sets of tokens in the description. During the pre-training step, all layers of the Transformer model are considered trainable to learn the context. At each update of the model, we take a random batch of CVEs. In the second phase, we prepare the training links considering CVEs and CWEs a priori. For a batch of links, the prediction loss is computed using Equation 1. Typically, the number of positive links is far smaller compared to the number of negative links. We select k-random links for a CVE, and positive links are either weighted or repeated to balance the larger number of negative links.

Similar to the CVE-CWE mapping model, we map CWEs to CAPECs. CAPECs are hierarchically designed and have many-to-many relations with CWEs. However, unlike CVEs, there is limited CWEs and CAPECs information (about 546), and the

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 15,2025 at 09:36:32 UTC from IEEE Xplore. Restrictions apply.

mappings between CWEs to CAPECs are incomplete at the moment. Therefore, we use positive CWE-CAPEC relations from MITRE in our training process. Further, we also include negative links by considering CWE and CAPEC pairs that are not in the corresponding hierarchical branch (connected component) for training purposes.

D. Parallelization of VWC-BERT *on GPU*

A learning model with a large number of hidden layers and parameters will not fit in the memory of a single accelerator (GPU). Such models are typically partitioned into different manageable parts and deployed into multiple GPUs. VWC-BERT considered here has a maximum 408M parameters, which fit a single GPU memory. Therefore, we also consider data parallel methods. Considering a single-node multi-GPU scenario, the Data Parallel (DP) mode has a principal GPU that distributes (scatters) the batch and replicates the model onto other GPUs (including itself). The GPUs perform forward propagation and compute loss and gradients in parallel. Finally, the gradients are gathered back to the principal GPU, where the gradients are aggregated to update the model. In contrast, in Distributed Data Parallel (DDP), each GPU across each node gets a separate process. Each GPU process can see only a portion of the batches and computes gradients in parallel. The gradients are synchronized and averaged across processes, and each process updates its optimizer. DP is limited to function only on a single node and is slower than DDP even for a single node multi-GPU configuration due to the overhead from Global Interpreter Lock (GIL) contentions across multiple threads, the additional overhead introduced by scatter-gather operations, and the per-iteration model replication.

VWC-BERT has two training phases: Pre-training of the masked language model, and link prediction training. Scaling the pre-training phase is relatively straightforward as we create a batch of CVE descriptions and process them in parallel. For Link prediction training, we propose two types of parallel computation: Link pair processor and principal link manager.

Link Pair Processor (LPP): In the first approach, we prepare training link pairs based on CWE class labels. For example, on average, a single CVE has three positive links (corresponding to the three levels in the CWE hierarchy) and 121 negative links (considering all 124 CWE classes). We can repeat positive links (or weight) to balance with negative links, which gives us 242 links on average for a single CVE. We create possible link pairs for a batch of CVEs, and we sample a mini-batch of links and process them in parallel.

This method can be implemented to run in DDP mode and is suitable for multi-node multi-GPU architectures at various scales. However, it is computationally expensive for a single CVE since the same CVE is passed through the BERT model 242 times. Additionally, with 60K training data and 124 classes, we get about 14.5M links. Therefore, even considering k random negative links, the model has to process many links to achieve good performance.

Principal Link Manager (PLM): To address the combinatorial explosion, the second approach uses a principal GPU that takes a batch of CVEs and CWEs and partitions them to the GPUs for parallel processing. Once the principal GPU gets the representation, it creates link pair vectors $(\mathbf{x}_{cve}, \mathbf{y}_{cwe})$ with necessary repetition to balance positive and negative links and scatter them to link classifier for parallel processing. The link pairs are computed on CPUs after getting a batch of CVEs and CWEs. We keep track of only the indices and pass that information along to the principal GPU. PLM is significantly faster, as unlike LPP, each CVE or CWE only needs to make one forward pass through the transformer model to get the representation. The PLM scheme is faster on small-scale single-node multi-GPU settings and considerably faster in processing the same amount of links. However, with an increasing number of GPUs, the principal GPU's coordination overhead increases and consequently reduces parallel efficiency (yet outperforms LPP in training time).

IV. Experimental Results

Experimental Setup We benchmark performance on different AI accelerators for both Data-Parallel (DP) and Distributed Data-Parallel (DDP) modes implemented using Pytorch (v1.4.0) and Pytorch Lightning [5](v1.2.4) libraries. We use the Transformers [14] (v4.4.2) library to access different pre-trained transformer models. In the training phase, we vary batch sizes and the number of GPUs to highlight strong scaling performance for different hyper-parameter settings. We used Tesla P100-PCIE-16GB (1 to 8 GPUs), Tesla V100-SXM3-32GB (1 to 16 GPUs), and A100-SXM4-40GB (1 to 8 GPUs) for evaluation. By default, we use $\min\{$nCPU, batch_size, $4\times$ nGPU$\}$ worker threads to prepare the batches.

Dataset and VWC-BERT Settings: We use the Common Vulnerabilities and Exposure (CVE) dataset from 1999-2020 from the National Vulnerability Database. There are 137K usable CVEs among which about 83K are labeled. The labeled CVEs use about 124 CWEs. The hierarchical relations of these CWEs are taken from MITRE. The CWE to CAPEC relations are also taken from Mitre. Since masked language models do not require labeled information, we use all the 137 CVE descriptions for pre-training. For evaluation, we consider stratified k-fold cross-validation, where 70% of the data from each category are taken for training, 20% for testing, and 10% for validation of hyper-parameter settings. This leads to data sizes of 58K for training, 8K for validation, and 17K for testing. Similar to the settings of Das et al. [3], we allow all layers to be updated in the pre-training phase. However, during link prediction, the first nine layers in BERT-Base, and RoBERTa-Base are fixed. For BERT-Large and RoBERTa-Large, the first 21 layers are kept fixed during link prediction. Moreover, since DistilBERT [12] has relatively fewer layers, we allow all layers to be updated during link prediction. The Adamw [8] optimizer is used by default with a learning rate of $2e^{-5}$ and zero warm-up steps. Codes are shared in a public GitHub repository.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 15,2025 at 09:36:32 UTC from IEEE Xplore. Restrictions apply.

A. *Qualitative Assessment*

For qualitative assessment, we consider the hierarchical prediction accuracy similar to the original work of Das et al. [3]. While the setting $\{k_1 = 1, k_2 = 1, k_3 = 1\}$ gives precise prediction for selecting the best CWE from each level in the hierarchy, the setting $\{k_1 = 5, k_2 = 2, k_3 = 2\}$ gives a relaxed prediction by selecting the best five, two and two CWEs for the first, second and third level respectively. If the original CWE belongs to one of the paths, the prediction is considered accurate. We also report the F-1 scores of correctly predicted links between a CVE and a CWE. Accuracy measures of VWC-BERT using different pre-trained Language Models are summarized in Table I. We executed 20 epochs with a batch size of 16 for the pre-training phase of all the models, and used a batch size of 32 with 15 epochs for the link prediction phase. The prediction accuracy is comparable with Das et al. Some differences exist because of the differences between training and test datasets.

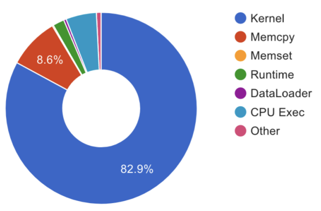
TABLE I: Qualitative performance of VWC-BERT with different Transformer models. The test accuracy represents the fraction of correctly predicted CWEs. The test link F-1 score corresponds to the binary prediction of links between a CVE and a CWE.

Model	**Test Accuracy**			**Test Link**
	1, 1, 1	**3, 2, 1**	**5, 2, 2**	**F-1 Score**
BERT-Base	0.8423	0.9463	0.9698	0.9746
BERT-Large	0.8460	0.9450	0.9689	0.9757
DistilBERT	0.8652	0.9546	0.9755	0.9788
RoBERTa-Base	0.8502	0.9526	0.9750	0.9784
RoBERTa-Large	**0.8654**	**0.9595**	**0.9791**	**0.9808**

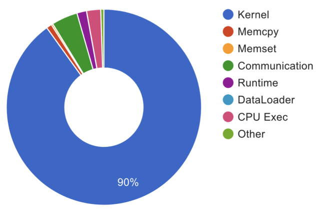
In comparison to the previous work, we demonstrate that switching from BERT-Base to BERT-Large to RoBERTa improves the precise prediction accuracy by 2%. We also observe that DistilBERT performs better than BERT-Base. We hypothesize that the improvements can be attributed to the link prediction training updates allowed for all layers of DistilBERT, thus making the model flexible. Given the novelty of the problem, mapping CWEs to CAPECs is challenging due to the lack of ground truth association information. We observe over 90% accuracy for link prediction (link between a given pair of CWE and CAPEC), and about 75% accuracy for predicting the CAPEC for a given CWE.

B. *Pre-training Performance*

Data Parallel vs. Distributed Data Parallel Modes: Fig. 4 shows the minute/epoch pre-training runtime of BERT-Base model on different AI accelerators using DP and DDP parallelization schemes. We observe better strong scaling for DDP on a single-node multi-GPU system. In contrast, for Data Parallel, the increase in the number of GPUs leads to an increase in synchronization overheads for the principal GPU. As a result, GIL lock overhead grows, thus contributing to higher runtime relative to DDP. Therefore, when using DP, we need to consider the overhead costs to offset the increased concurrency gains with more GPUs. We summarize execution times for DP and DDP in Fig. 3.



(a) DP



(b) DDP

Fig. 3: Execution summary of DP and DDP.

Strong Scaling Across Platforms: The average training time per epoch while changing the AI accelerator is captured in Figures 4 and 5. Considering DDP (Fig. 5a, 5b), A100 is about 3× faster than V100 for both BERT-Base and BERT-Large. In terms of DP (Fig. 5c), performance with two A100 GPUs is about 2.5× faster than V100, and V100 is about 2× faster than P100. However, this scaling behavior does not remain consistent for DP with larger number of GPUs due to overhead costs as discussed earlier. Increasing batch size improves performance but is constrained by the amount of memory available on a given system, and therefore, newer generations benefit from larger memories.

C. *Performance Highlights*

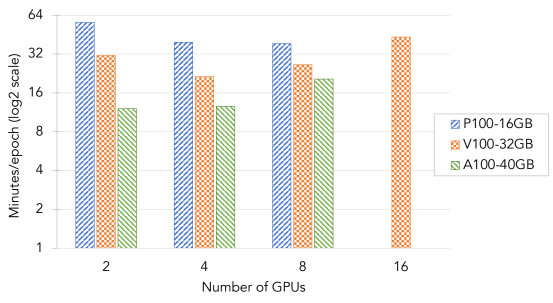
We summarize the key strong scaling results for pre-training and link prediction in Table II. We observe good strong scaling behavior using the Distributed Data Parallel (DDP) for pre-training, as well as with Link Pair Processor (LPP) for link prediction. VWC-BERT scales almost linearly with the increase in GPUs for both V100 and A100 for all Transformer models. In addition, larger Transformer models provide increased accuracy for classification. Due to the synchronization overhead on the Principal GPU, batch sizes need to increase proportionally with the number of GPUs to obtain better scaling of training time. The Principal Link Manager (PLM) scales better for RoBERTa [7] and DistilBERT [12], compared to BERT since these models are optimized to perform well for larger batch sizes as argued by the respective authors. The CPU/GPU profiler results for the rank-0 node in the distributed data-parallel computation are shared[1]. However, for brevity and page limitation, the results are not described here.

TABLE II: Key scaling performance results on V100 and A100 GPUs. For strong scaling in the Link Pair Processor (LPP) method, the entry "M× (N)" states that M fold speedup was obtained when the number of GPUs was increased by a factor of N with the batch size being fixed. For the Principal Link Manager (PLM) method, the entry "M× (N)" states that M fold speedup was obtained when the number of GPUs as well as the batch size were increased by a factor of N.

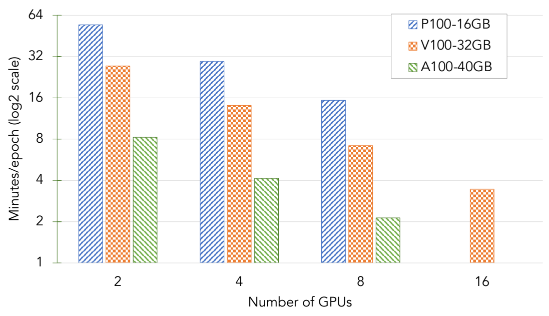
Model	**Strong Scaling, fixed batch size**			**Changing batch and GPUs**	
	Pre-train (DDP)		**Link Prediction (LPP)**	**Principal Link Manager (PLM)**	
	V100	**A100**	**A100**	**V100**	**A100**
BERT-Base	$8\times(8)$	$4\times(4)$	$4\times(4)$	$3\times(4)$	$2\times(4)$
BERT-Large	$8\times(8)$	$4\times(4)$	$4\times(4)$	$4\times(4)$	$2\times(4)$
RoBERTa-Base	$8\times(8)$	$4\times(4)$	$4\times(4)$	$11\times(8)$	$4\times(4)$
RoBERTa-Large	$8\times(8)$	$4\times(4)$	$4\times(4)$	$21\times(16)$	$4\times(4)$
DistilBERT	$8\times(8)$	$4\times(4)$	$4\times(4)$	$12\times(8)$	$5\times(4)$

[1]CPU/GPU profiler with tensorboard logs

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 15,2025 at 09:36:32 UTC from IEEE Xplore. Restrictions apply.

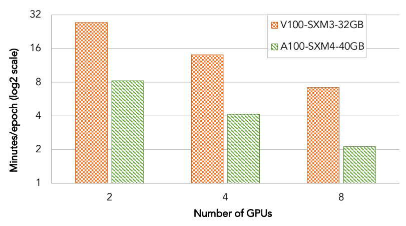


(a) Data-Parallel (DP), $b = 32$

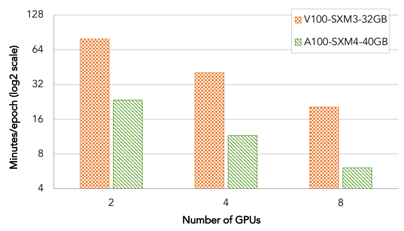


(b) Distributed Data-Parallel (DDP), $b = 32$

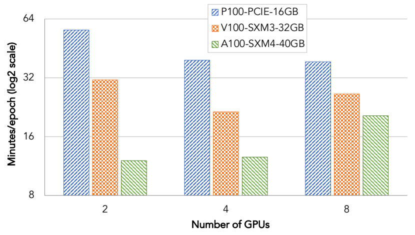
Fig. 4: Scaling behavior of the training time (a) Data-Parallel and (b) Distributed Data Parallel modes across different accelerators. The Y-axis plots the average training time of BERT-Base (minutes/epoch) with a batch size of 32.



(a) BERT-Base, DDP, $b = 32$, 2-8 GPUs



(b) BERT-Large, DDP, $b = 32$, 2-8 GPUs



(c) BERT-Base, DP, $b = 32$

Fig. 5: Scaling across different generations of Nvidia GPUs. (a,b) Average training time per epoch on V100 and A100 with different numbers of GPUs for BERT-Base and BERT-Large. (c) Run-time across generations of GPU in DP mode.

V. Summary and Future Work

Hundreds of new vulnerabilities are discovered every week. Therefore, automated methods to map vulnerabilities to weaknesses and exploits is an important tool in cyber-defense. The scalable VWC-BERT framework presented in this paper is an important step towards automation by using the rich semantic knowledge that has been carefully curated for decades. Modern AI architectures enable us to exploit recent advances in AI for practical applications. Our future work will include further experiments on mapping CWEs to CAPECs, inclusion of sequence-to-sequence language models, and enhancing the text corpus with cybersecurity specific documents.

Acknowledgements

This work is supported in part by Project 38, a collaboration between Department of Defense & Department of Energy, and award #IGEM22-001 from the Idaho Global Entrepreneurial Mission/Higher Education Research Council. We are grateful for access to institutional computational resources at PNNL.

References

[1] Davide Chicco. Siamese neural networks: An overview. *Artificial Neural Networks*, pages 73–94, 2020.

[2] Daniela Soares Cruzes, Michael Felderer, Tosin Daniel Oyetoyan, Matthias Gander, and Irdin Pekaric. How is security testing done in agile teams? A cross-case analysis of four software teams. In *XP*, volume 283 of *Lecture Notes in Business Information Processing*, pages 201–216, 2017.

[3] Siddhartha Shankar Das, Edoardo Serra, Mahantesh Halappanavar, Alex Pothen, and Ehab Al-Shaer. V2w-bert: A framework for effective hierarchical multiclass classification of software vulnerabilities. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–12. IEEE, 2021.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

[5] WA Falcon and .al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

[6] M. Jimenez, M. Papadakis, and Y. L. Traon. An empirical analysis of vulnerabilities in openssl and the linux kernel. In *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, pages 105–112, Dec 2016.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, et al. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 2019.

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

[9] Robert A. Martin and Sean Barnum. Common weakness enumeration (CWE) status update. *Ada Lett.*, pages 88–91, 2008.

[10] K. Radhakrishnan, R. R. Menon, and H. V. Nath. A survey of zero-day malware attacks and its detection methodology. In *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pages 533–539, 2019.

[11] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. Survey of machine learning accelerators. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–12. IEEE, 2020.

[12] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv:1910.01108*, 2019.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[14] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, October 2020.

[15] Mingzhi Zheng, Dinghan Shen, Yelong Shen, Weizhu Chen, and Lin Xiao. Improving self-supervised pre-training via a fully-explored masked language model. *arXiv:2010.06040*, 2020.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 15,2025 at 09:36:32 UTC from IEEE Xplore. Restrictions apply.