Investigating Side-Channel Attacks on the Vision Mixture-of-Experts Model

CS585 Project Writeup

Mark Chen, Michael Li, Islina Shan, Yi Zhou

April 19, 2024

Abstract

In this project, we investigate the vulnerability of Vision Mixture of Experts (V-MoE) models to side-channel attacks. Each expert in a machine learning (ML) model that leverages MoE components may run on a different GPU Trusted Execution Environment (TEE) during inference, which may allow an attacker to observe communication metadata between the router and experts. We demonstrate that the class label of input data can be inferred by observing the communication between different computing nodes assuming an unencrypted communication channel, while the class label of the dominant class in a batched query can be inferred assuming an encrypted communication channel. We propose a series of mitigation techniques for enhancing the security of communication channels, including co-hosting of V-MoE experts and traffic engineering.

1 Introduction

Mixture of Experts (MoE) models have become increasingly important in large-scale machine learning models due to their ability to enhance training efficiency [1, 2]. A key feature of MoE is the presence of specialized gating networks (routers), which selectively activate a subset of experts within an MoE block, thereby significantly expanding model capacity while reducing computation cost [3]. Experts are usually spread across multiple computing nodes, allowing the model to leverage parallel computing. The communication between computing nodes, however, could leak information to an malicious attacker, which may compromise the confidentiality of training data [4].

In this paper, we explore a potential attack on Sparse MoE models designed for vision applications (V-MoE)[5]. We demonstrate that an attacker could infer information about the original input data when they eavesdrop on the communication channel between routers and experts. We also propose several mitigation techniques to enhance the security of these communications.

2 Background and Related Work

2.1 Trusted Execution Environments

Trusted Execution Environment (TEE) [6] has experienced growing popularity, especially in the field of cloud computing. A TEE can establish a region with isolated resources that guarantees the user program's confidentiality and integrity and operates in an isolated environment. Additionally, TEEs offer remote attestation mechanisms that guarantees the trustworthiness of participating enclaves.

GPU TEEs, such as Graviton [7], play an increasingly important role with the growing need for secure GPU-accelerated training and deployment of ML models. GPU TEEs, similar to CPU TEEs, provide a secure context for the safe training of models.

While TEEs protect the confidentiality and integrity of data and code within, they typically do not have built-in protections against side-channel attacks. This allows attackers to intercept and interpret the metadata of encrypted communication, which may lead to the extraction of sensitive information. For example, the authors of Telekine, a system that provides secure communication channel between client machines and cloud GPUs, demonstrate that an adversary could classify images from ImageNet with up to 78% accuracy solely based on the timing of GPU kernel execution [4].

2.2 Distributed Deployment of Machine Learning Models

Distributed deployment of ML models is essential for large ML models primarily due to the immense computational resources they require, which often exceed the capacity of a single machine [8]. By spreading the workload across multiple processors or nodes, distributed deployment enables the handling of more substantial datasets and the execution of more complex algorithms at a scale that would not be possible on a single device. Modern machine learning libraries like PyTorch [9] provide support for training Neural Networks in parallel.

Unfortunately, distributed deployment of an ML model exposes the model to more potential attacks. Attackers are able to infer information about the input data by observing the communications between different computing nodes [10, 11]. In our work, we investigate such an attack on a V-MoE model, where the data leakage of a distributed deployment is simulated.

2.3 Mixture of Experts

Mixture of Experts, or MoEs for short, is a class of transformers. In particular, an MoE consists of two main components.

1. MoE blocks: MoE blocks are layers that include multiple "expert networks" (typically smaller neural networks that are each capable of handling specific types of input data or tasks). This is aimed at improving efficiency and scalability, particularly in handling diverse or complex datasets. In practice, each expert can also use more complex networks or even a MoE itself leading to hierarchical MoEs [5].

2. Gating Network: the gating network (router) determines which expert in the MoE block is best suited to process a given part of the input data. The router dynamically dispatches

inputs to the most appropriate experts based on the inputs' features. This decision-making process can be learned during training or based on a set of predefined criteria and is crucial for maximizing the effectiveness and efficiency of the MoE block [5].

Vision MoE [5], or V-MoE, is a sparse version of the Vision Transformer (ViT) that matches the performance of state-of-art networks while requiring half as much compute during inference. Figure 1 shows the standard architecture for a V-MoE model. It is composed of L ViT blocks. In MoE blocks, multi-head attention is followed by a sparsely activated mixture of MLPs, where each MLP is an expert. The input image for V-MoE is divided into P identically-sized patches and then encoded into tokens before being processed by the first ViT block.

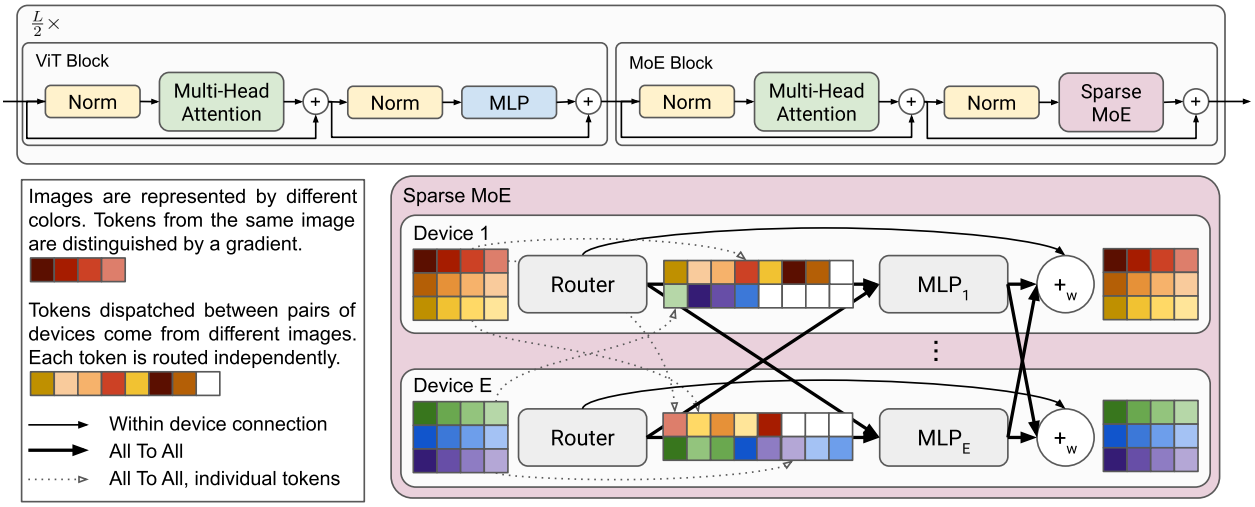


Figure 1: V-MoE architecture [5]. The image shows a standard architecture where MoE and ViT blocks are found in alternate layers. Note that the checkpoint used in this paper does not adhere to this configuration. Each image is split into "patches" and each patch is encoded into "tokens". We may refer to "patches" and "tokens" interchangeably.

The MoE takes as input the previous block's output encoding for each patch. Then the MoE blocks use the routing function $g(\mathbf{x})$ in Equation 1 to determine the assignment for the encoding of each patch, where operation $\text{TOP}_K(\mathbf{x})$ gets the index of largest K elements in $\mathbf{x}$ and $\epsilon \sim \mathcal{N}(0, \frac{1}{E^2})$.

$$g(\mathbf{x}) = \text{TOP}_K(\text{softmax}(\mathbf{W}\mathbf{x} + \epsilon)) \tag{1}$$

3 Attack on V-MoE

3.1 Threat Model

We assume the model is deployed on multiple GPU TEEs, which guarantee the integrity and confidentiality of data during deployment on each computing nodes. However, GPU TEEs do not secure the communication between these nodes. Unencrypted communication channels allow attackers to directly parse messages and extract the expert assignment of each patch. Moreover, even with encrypted communication, attackers can still infer critical data by analyzing the timing, size, and flow of the messages.

Our study show that the attacker is able to infer information about the original input data by observing either encrypted or unencrypted channel during model evaluation. For the encrypted channel, we focus on the communication size side-channel.

3.2 Exploit Intermediate Traffic

We focus on exploiting the traffic between routers and experts in this section. To train the adversary, the adversary can use the victim model as an oracle for intermediate traffic and output class. For a batch b of size B, assume the classes of every image to be $\mathbf{C}(b) \in \mathbb{N}^B$, and the observation of intermediate traffic to be $\mathbf{O}(b)$.

By observing several batches in the inference stage, the adversary eavesdrops on multiple $(\mathbf{C}(b_i), \mathbf{O}(b_i))$ to learn the relationship between $\mathbf{C}(b_i)$ and $\mathbf{O}(b_i)$. During the attacking phase, given a new batch b', the adversary's goal is to infer the class of any image in b' using $\mathbf{O}(b')$ and existing observations.

3.2.1 Unencrypted Channel

The attacker's goal is to infer the class label of each image in a batch that a client asks the V-MoE model to process.

For each image, there are P tokens that needs to be dispatched to one or more of the E experts in every MoE block. If the router and experts communicates through an unencrypted communication channel, the adversary that eavesdrops on the channel then has the capability to know the dispatch plan of every token in every batch. i.e. every token to expert mapping.

For the p-th token in the i-th image in a batch b, the adversary can infer that the p-th token is dispatched to expert set $e^l_{p,i} = \{e^l_{p,i,1}, \ldots, e^l_{p,i,K}\}$ in the l-th MoE block, where one token is processed by K experts. More specifically, at most $K = 2$ MoE blocks are used to evaluate one token in our victim model, so we denote the dispatch plans of the p-th token of the i-th image at the two MoE layers to be $\{e^1_{p,i,1}, e^1_{p,i,2}, e^2_{p,i,1}, e^2_{p,i,2}\}$. The adversary's observation in the unencrypted channel setting will be $\mathbf{O}(b) = \{e^1_{p,i,1} \parallel e^1_{p,i,2} \parallel e^2_{p,i,1} \parallel e^2_{p,i,2} \mid \forall p \in b, \forall i \in b\}$.

3.2.2 Encrypted Channel

The attacker's goal is to infer the class label of the dominant class in a batch that a client asks the V-MoE model to process.

In the unecrypted channel, the adversary is capable of observing fine-grained information of token to expert mapping. This fine-grained information leakage can be easily mitigated through the use of secured communication channels. However, the number of tokens each expert receives can still be potentially observed through the length of messages between the router and experts or even through timing side channel of experts. Since the routing function selects K experts for each token based on the expert's performance conditioned on that token, the number of tokens each expert receive may be correlated to the classes labels that are present in the input batch.

Similarly, let the number of tokens the k-th expert receive to be $n_k = \sum_{p,i}[(k = e^1_{p,i,1}) + (k = e^1_{p,i,2}) + (k = e^2_{p,i,1}) + (k = e^2_{p,i,2})]$. The observation of the attacker in the encrypted channel threat model becomes $\mathbf{O}(b) = \langle n_1, n_2, \ldots, n_{2E} \rangle$ where E is the number of experts per MoE block.

Since the adversary is incapable of inferring fine-grained information when the communication channel is encrypted, the goal becomes inferring the dominant class (the most frequent class) in the input batch b' according to past observations and $\mathbf{O}(b')$.

4 Evaluation

To evaluate the performance of our attacks, we use the pretrained checkpoint provided in [12] as our victim model. The victim model deviates from the standard architecture shown of V-MoE models in Figure 1. Instead of having alternating MoE and ViT blocks, the specific variant that we used has two MoE blocks in the last two odd-numbered blocks where each MoE block has $E = 8$ experts. The remaining 6 blocks are vanilla ViT blocks. The specific checkpoint that we used has the prefix `vmoe_s32_last2_ilsvrc2012_randaug_light1_ft_ilsvrc2012`, and according to [12] was trained and finetuned on the ILSVRC2012 dataset [13]. Please see section 3.1 for details on V-MoE models in general.

4.1 Unencrypted Channel

In the unencrypted channel threat model, the attacker seeks to infer the class label of each image in an input batch which is submitted by a client of the V-MoE model. Recall that the threat model allows the attacker to read unencrypted traffic between the router and experts in both MoE blocks, meaning that the attacker knows the expert assignment of each patch (see Figure 1 for more on "patches").

First, we collect patch ID to expert ID mappings for each ImageNet image that has (384×384) pixels. Since there are two MoE blocks in total and each patch can be assigned to up to two experts in each MoE block, there are four expert assignments for one patch, where one patch has (32×32) pixels. This means that each ImageNet image has $\frac{384 \times 384}{32 \times 32} \cdot 4 = 576$ expert assignments. The expert assignments are reshaped into a $(24 \times 24 \times 1)$ matrix, which is the shape of the training and evaluation data used for each attacker model.

During the data collection process, all 1000 ImageNet classes are used. We trained ten attacker models, where each model is exposed to ImageNet training data from n randomly selected classes, where $n \in \{2, 4, 8, 16, 32, 64, 128, 256, 512, 1000\}$. This means that a model $f_n(\mathbf{x})$ trained on four classes is only evaluated using data from the same four classes. note that the data used to train models exposed to fewer classes is not a subset of the data used to train models exposed to more classes. Also note that the number of data points (where each data point is a (24×24) matrix) varies from model to model.

Then, we train different attacker models by exposing them to different number of input classes. The statistics (i.e., variance and arithmetic mean) of each model's training dataset (σ_n^2, μ_n) are separately calculated, and training and evaluation data is normalised using before being processed by the model. We use a modified version of the ResNet-20-like architecture (see section 4.2 of [14]). Specifically, we modified the number of output channels of all convolution layers so that there are a higher number of output channels for models exposed to more classes; we used an average pooling kernel size of 2; and there are two fully-connected layers after average pooling, instead of one fully-connected layer. This means that each instnce of the model has 21 layers in total. For all models, we trained for 200 epochs with learning rate 0.01 which is decayed at epoch 100 and 150,

both by factors of 10. We used l_2 regularization with strength 10^{-4}, a batch size of 512, and an SGD optimizer with 0.875 momentum and no Nesterov momentum. Cross entropy loss is used as the criterion. The training procedure and model are implemented using PyTorch [15].

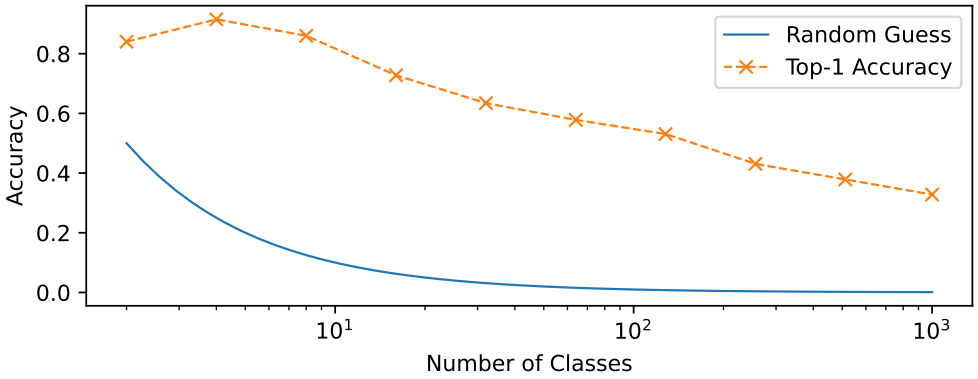


Figure 2: Performance of the attack (unencrypted threat model) measured using fraction of correct class label prediction for each model f_n.

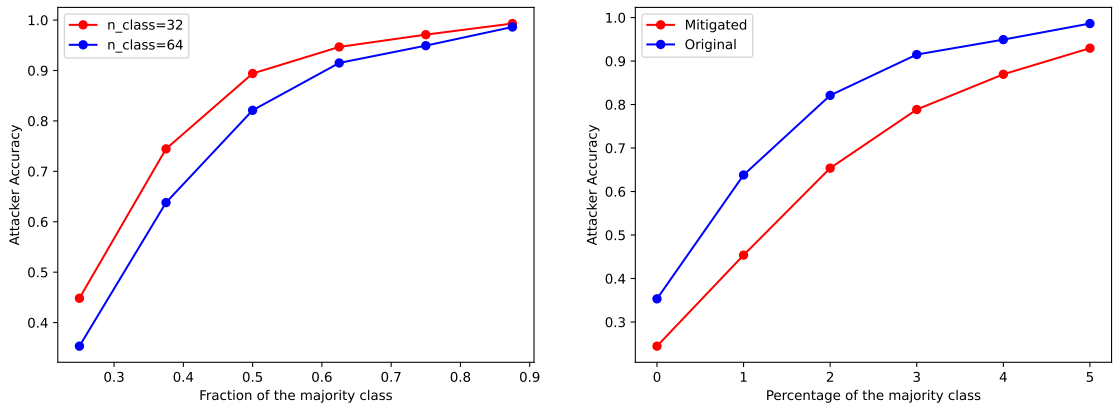
Figure 2 shows the performance of the attack. After training for 200 epochs, each model is evaluated using 50 images from the n selected classes used to train them. All of our attack models f_n have higher success rates than the random guess success rate. The accuracy of the attack drops as the number of classes included in the training set and used for evaluation increases. Having more classes means that the model has to be more discriminating, and the ResNet-20-like architecture is not a complex model, as demonstrated by its 91% accuracy on evaluation data from the CIFAR-10 dataset [14]. Because we did not perform hyperparameter finetuning during the training process, it is expected that our models' performance are no higher than 90%.

4.2 Encrypted Channel

		Dominant Class Fraction					
		0.25	0.375	0.5	0.625	0.75	0.875
# Classes	2	NA	NA	NA	0.960	1	1
	4	NA	0.593	0.938	1	1	1
	8	0.427	0.766	0.946	1	1	1
	16	0.500	0.774	0.877	0.971	1	1
	32	0.448	0.745	0.894	0.947	0.971	0.993
	64	0.353	0.638	0.821	0.915	0.949	0.986

Table 1: Accuracy of dominant class prediction as a function of the fraction of the dominant class within a batch and the maximum number of unique class labels in a batch.

In the encrypted channel setting, the attacker seeks to infer the dominant class of an input batch. Each batch is made up of the given fraction of images from a dominant class and randomly selected images from the remaining classes. We use the original support vector classification in python sci-kit learn package without further fine-tuning and cherry-picking as attacker model.



(a) Attack accuracy vs number fraction of the dominant class in each batch for $n \in \{32, 64\}$.

(b) Attack accuracy where experts $2k, 2k+1$ are co-hosted on the same machine.

Figure 3: Evaluation of attacker model accuracy

Compared to the unencrypted threat model, we use a different data collection process here. Whereas all 1000 classes of ImageNet images are used to collect data for the unencrypted threat model and there is no restriction on which classes can appear in any given batch during the data collection process, in the encrypted threat model, each batch that's processed by the V-MoE during data collection contains at most n unique classes where $n \in [2, 4, 8, 16, 32, 64]$ and the fraction p of the dominant class in each batch is fixed, where $p \in [0.25, 0.375, 0.5, 0.625, 0.75, 0.875]$. It is further guaranteed that the number of images from non-dominant classes in each batch is smaller than at of the dominant class. Similar to the unencrypted case, the data seen by each attacker model comes from the same n classes in the encrypted threat model. As a result, the dataset collected using $(n = 2, p = 0.875)$ is much smaller than the dataset collected using $(n = 64, p = 0.25)$.

If each ImageNet class has M images, then the number of batches that can be created is *at most* $n \cdot M \div (\frac{32}{p})$. There are $\frac{32}{p}$ instances of the dominant class in each 32-image batch, so there can be at most $M \div (\frac{32}{p})$ batches of data with the same dominant class. Therefore, with n distinct classes, there can be $n \cdot M \div (\frac{32}{p})$ batches of data in total. A side effect is that images may be repeated across different batches, where the non-dominant images in one batch becomes the dominant class for another. The number of images in the dominant class limits the number of batches that can be created, since non-dominant classes can be arbitrarily filled in by images from any other class. We fixed the size of the batch being processed by the V-MoE to 32 images for every instance of data collection.

The input for the attacker model is a set of 16-dimension vectors, where the value of the i-th dimension indicates the number of tokens i-th experts receive. Specifically, the first 8 items in the vector correspond to the experts on layer-5 and the last 8 items correspond to the experts on layer-7.

We evaluate the attacker model's accuracy for different combinations of maximum number of classes n and dominant class fraction p. The attacker model accuracy is shown in Table 1. We also compare the performance under 32 classes and 64 classes in Figure 3a.

5 Mitigation Techniques

5.1 Co-hosting

By default, every expert in an MoE model can be hosted separately to improve latency. This leads to a higher level of information leakage. Different placement strategies on the computing nodes where different experts may be co-located could reduce information leakage.

Assuming we have $K + 1$ compute nodes and the router is hosted on node-0. One such placement strategy that reduces information leakage can be defined as $\mathcal{S} : [1, 2E] \rightarrow [0, K]$, where expert v is hosted on node-$S(v)$.

In the encrypted channel, if an expert is co-located with the router, then we cannot infer much from that expert since the flow of information happen internally inside the TEE. If multiple experts are co-located on the compute node other than 0, the adversary can only observe the sum of their received number of tokens. After applyign the placement strategy, the number of tokens the k-th node receives becomes $\tilde{n}_k = \sum_{p,i} \sum_{v \in \{e^1_{p,i,1}, e^1_{p,i,2}, e^2_{p,i,1}, e^2_{p,i,2}\}} [\mathcal{S}(v) = k]$, and the observed information is degraded into $\tilde{\mathbf{O}} = (\tilde{n}_1, \ldots, \tilde{n}_K)$, where $(K \leq 2E)$.

For example, the attacker observes $\mathbf{O} = (n_1, n_2, \ldots, n_{16}) \in \mathbb{N}^{16}$ if every expert is hosted separately using our default configuration. If we combine every two experts (i.e., combine experts $2k$ and $2k{+}1$), the attacker's observation will collapse to $\tilde{\mathbf{O}} = (n_1{+}n_2, n_3{+}n_4, \ldots, n_{15}{+}n_{16}) \in \mathbb{N}^8$. We compare the attacker model accuracy of our original setting and the combined setting under 64 class in Figure 3b. The reduced success of the attack after applying the placement strategy shows that different placement strategies can lead to better privacy protection.

The ideal case for privacy is when everything is hosted one the same node, which implies $\tilde{\mathbf{O}} = \emptyset$. It's possible to find out the best placement strategy according to performance and privacy trade-offs by iterating through all possible placement strategies.

5.2 Traffic Engineering

There are two ways of mitigating privacy leakage through traffic engineering, one requires changes to the MoE model's router configurations, while the other is generic to distributed communications. The main factor that allows the attacker to infer information from the router-to-expert communication is that the router selects the top k experts for each token, which not only leaks privacy information of the input but also leads to potential load imbalances between different experts. One potential modification here is: instead of choosing experts for tokens, we can choose top-k tokens for experts, where each expert receive roughly the same number of tokens, meaning a number of tokens from each image will never be processed. This configuration leads to a better load balancing and privacy, but introduces potential accuracy degradation.

Other strategies to anonymize and obfuscate messages are often utilized to enhance privacy in distributed computing settings. Techniques such as standardizing message sizes via padding [16] and introducing decoy traffic [17] are used to confuse eavesdroppers. However, it is worth noting that these methods could bring additional performance overhead.

6 Conclusion

In this paper, we demonstrate the vulnerability of Vision Mixture of Experts (V-MoE) models to side-channel attacks. While trusted execution environments like GPU TEE has ensured the confidentiality and integrity of code executing within the TEE, they do not protect against threats arising from the communication between nodes. We show that both encrypted and unencrypted communication channels are susceptible to attacks that can infer sensitive information about the input data.

We propose several mitigation methods such as co-hosting experts and traffic engineering. Notably, we show that by placing appropriate experts together on a single computing node, information leakage can be significantly reduced. Future work can focus on developing more sophisticated algorithms for expert placements to reduce information leakage.

7 Decomposition of Work

Yi Zhou: Proposing and narrowing down project ideas. Early phase research on related works, proposing the idea of focusing on ML models. Assisting Michael diving into v-MoE and finalizing the v-MoE data collection setup. Encrypted channel attacker model design, implementation and evaluation. Mitigation technique related work research and proof of concept evaluation. Talk slides page 10-14. Writing of writeup section 2.3, 3.2, 4.2, 5.1, 5.2.
Mark Chen: I proposed initial ideas for the project and investigated some related works. Wrote abstract, introduction, some of the background, mitigation and conclusion sections for the paper. Covered the first few slides in the presentation.
Michael Li: I did a deep dive into the V-MoE paper's code implementation and determined how to make edits to their code base to enable data collection. I performed data collection for the encrypted and unencrypted threat models, which involved writing custom code for aggregating data into batches with dominant classes, and performed training and evaluation for the unencrypted channel adversaries. I plotted figure 2 and table 1 (data for table 1 was generated by Yi Zhou). I wrote section 4.1, part of section 4.2, and proof-read all the other sections. I drafted slides in the presentation (including threat model, evaluation, and V-MoE background).
Islina Shan: Proposed initial ideas for the project. Investigated privacy metrics including k-anonymity and l-closeness in the proposal. Helped set up computing cluster environment for running experiments and discussed experimental set-up and results. Drafted background section for the paper. Edited and covered main contribution, threat models, and methodology overview in the presentation.

References

[1] Yanqi Zhou et al. "Mixture-of-Experts with Expert Choice Routing". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 7103–7114. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/2f00ecd787b432c1d36f3de9800728eb-Paper-Conference.pdf.

[2] Jiaao He et al. "FastMoE: A Fast Mixture-of-Expert Training System". In: *CoRR* abs/2103.13262 (2021). arXiv: 2103.13262. URL: https://arxiv.org/abs/2103.13262.

[3] Noam Shazeer et al. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer". In: *arXiv preprint arXiv:1701.06538* (2017).

[4] Tyler Hunt et al. "Telekine: Secure Computing with Cloud GPUs". In: *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 817–833. ISBN: 978-1-939133-13-7. URL: https://www.usenix.org/conference/nsdi20/presentation/hunt.

[5] Carlos Riquelme et al. *Scaling Vision with Sparse Mixture of Experts*. 2021. arXiv: 2106.05974 [cs.CV].

[6] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. "Trusted execution environment: What it is, and what it is not". In: *2015 IEEE Trustcom/BigDataSE/Ispa*. Vol. 1. IEEE. 2015, pp. 57–64.

[7] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. "Graviton: Trusted execution environments on {GPUs}". In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 2018, pp. 681–696.

[8] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. "Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices". In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 2017, pp. 328–339. DOI: 10.1109/ICDCS.2017.226.

[9] Shen Li et al. "Pytorch distributed: Experiences on accelerating data parallel training". In: *arXiv preprint arXiv:2006.15704* (2020).

[10] Gilad Baruch, Moran Baruch, and Yoav Goldberg. "A little is enough: Circumventing defenses for distributed learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[11] Yayuan Xiong, Fengyuan Xu, and Sheng Zhong. "Detecting GAN-based privacy attack in distributed learning". In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE. 2020, pp. 1–6.

[12] *V-MoE*. https://github.com/google-research/vmoe/tree/main.

[13] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[14] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[15] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[16] Andrew C. Reed and Michael K. Reiter. *Optimally Hiding Object Sizes with Constrained Padding*. 2021. arXiv: 2108.01753 [cs.CR].

[17] Sambuddho Chakravarty et al. “Detecting Traffic Snooping in Tor Using Decoys”. In: *Recent Advances in Intrusion Detection*. Vol. 6961. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011. DOI: 10.1007/978-3-642-23644-0_12. URL: https://doi.org/10.1007/978-3-642-23644-0_12.