QuantAttack: Exploiting Quantization Techniques to Attack Vision Transformers Supplementary Material

A. Quantization Preliminaries

Quantization reduces the computational and memory demands of inference by using low-precision data types, such as 8-bit integers, instead of 32-bit floating points. This approach decreases memory usage and energy consumption, speeds up operations like matrix multiplication, and enables model deployment on embedded devices that often only support integer data types [2]. Two prevalent techniques in 8-bit quantization are zero-point quantization and absolute maximum (absmax) quantization. Both of these methods involve mapping floating-point values to a more compact int8 (1 byte) representation. This is achieved by normalizing the input values through scaling with a quantization constant.

A.1. Absolute Maximum Quantization

The absolute maximum method transforms inputs from a 16-bit floating point format into an 8-bit integer in the range of $[-127, 127]$. This transformation is achieved by multiplying the input matrix $\mathbf{X}_{f16}$ by a scaling factor S_{f16}. The scaling factor is computed as the ratio of 127 to the maximum absolute value in the entire tensor (infinity norm). The transformation formula is expressed as:

$$\mathbf{X}_{i8} = \lfloor S_{f16}\mathbf{X}_{f16} \rceil, \quad \text{where} \quad S_{f16} = \frac{127}{\|\mathbf{X}_{f16}\|_\infty} \tag{1}$$

Here, $\lfloor \cdot \rceil$ represents rounding to the nearest integer.

A.2. 8-bit Matrix Multiplication with 16-bit Floats

Considering hidden states $\mathbf{X}_{f16} \in \mathbb{R}^{s\times h}$ and weights $\mathbf{W}_{f16} \in \mathbb{R}^{h\times o}$, an 8-bit matrix multiplication resulting in 16-bit outputs is performed in the following way:

$$\mathbf{C}_{f16} \approx S_{f16}{\cdot}\mathbf{C}_{i32} = S_{f16}{\cdot}\mathbf{X}_{i8}\mathbf{W}_{i8} = S_{f16}{\cdot}Q(\mathbf{X}_{f16})Q(\mathbf{W}_{f16}) \tag{2}$$

Here, $Q(\cdot)$ represents the quantization operation (either absolute maximum or zeropoint).

A.3. Vector-wise Quantization for Matrix Multiplication

This method introduces a granular approach to quantization by assigning unique scaling constants to each row of $\mathbf{X}_{f16}$ and each column of $\mathbf{W}_{f16}$. The matrix multiplication result is then dequantized using the outer product of the scaling constants vectors $\mathbf{c}_{x_{f16}} \in \mathbb{R}^s$ and $\mathbf{c}_{w_{f16}} \in \mathbb{R}^o$:

$$\begin{aligned}\mathbf{C}_{f16} &\approx \frac{1}{\mathbf{c}_{x_{f16}} \otimes \mathbf{c}_{w_{f16}}}\mathbf{C}_{i32} \\ &= \mathbf{S} \cdot \mathbf{C}_{i32} \\ &= \mathbf{S} \cdot \mathbf{X}_{i8}\mathbf{W}_{i8} \\ &= \mathbf{S} \cdot Q(\mathbf{X}_{f16})Q(\mathbf{W}_{f16}),\end{aligned} \tag{3}$$

B. The LLM.int8() Quantization Technique

A significant problem for large-scale 8-bit transformers is that they have large magnitude features (columns), which are important for transformer performance and require high precision quantization. However, vector-wise quantization quantizes each row for the hidden state, which is ineffective for outlier features. Therefore, a new decomposition technique is developed, which focuses on high precision multiplication for these particular dimensions. A mixed-precision decomposition for matrix multiplication is proposed, where outlier feature dimensions are separated from non-outlier ones.

Formally, during inference, for every quantized layer, given the hidden states $\mathbf{X}_{f16} \in \mathbb{R}^{s\times h}$ and weights $\mathbf{W}_{f16} \in \mathbb{R}^{h\times o}$ with sequence dimension s, feature dimension h, and output dimension o, the main steps for efficient matrix multiplication are as follows (an illustration is shown in Figure 1):

1. **Outlier Extraction**: From the input hidden states $\mathbf{X}_{f16}$, extract all column indices that contain at least one outlier (*i.e.*, absolute values that are larger than a certain threshold τ) into the set $O = \{i \mid \exists |\mathbf{X}^i_{f16}| > \tau, 0 \geq i \geq h\}$.
2. **Mixed-Precision Multiplication**: The matrix multiplication process is divided into two segments. Outliers are multiplied using the standard matrix multiplication in float16, while non-outliers are first quantized (Section A.1) to their 8-bit representation and then multiplied in int8. This involves row-wise quantization for the hidden state and column-wise quantization for the weight matrix (Section A.3).

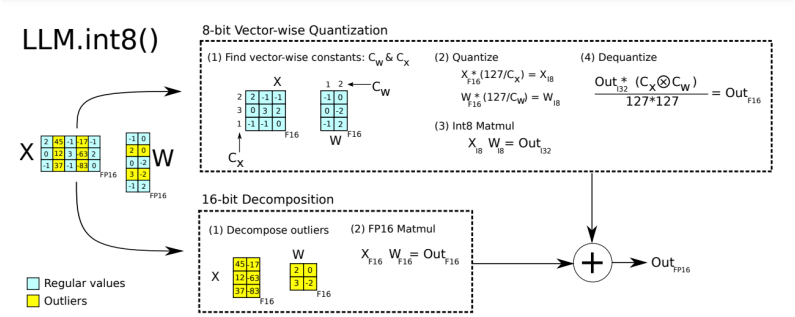


Figure 1. Illustration of mixed-precision matrix multiplication in the LLM.int8() quantization technique. The image is taken from [3].

Algorithm 1 LLM.int8() Matrix Multiplication

Require: Inputs $X_{f16} \in \mathbb{R}^{s \times h}$, Weights $W_{f16} \in \mathbb{R}^{h \times o}$, Threshold $\tau \in \mathbb{R}$

$O \leftarrow \emptyset$ {Initialize outlier set}
for $i \leftarrow 0$ **to** $h - 1$ **do**
  if $\max(|X_{f16}[:, i]|) > \tau$ **then**
    $O \leftarrow O \cup \{i\}$ {Add index to outliers}
$X_{i8} \leftarrow$ **quantize**(X_{f16}) {Quantize input}
$W_{i8} \leftarrow$ **quantize**(W_{f16}) {Quantize weights}
$M_{out} \leftarrow$ **matmul_f16**$(X_{f16}[O], W_{f16}[O])$ {f16 multiply for outliers}
$M_{non} \leftarrow$ **matmul_int8**$(X_{i8}, W_{i8}[\overline{O}])$ {Int8 multiply for non-outliers}
$M_{non} \leftarrow$ **dequantize**(M_{non}) {Dequantize result}
$C_{f16} \leftarrow M_{out} + M_{non}$ {Combine results}
return C_{f16}

3. **Dequantization and Aggregation**: The non-outlier results are dequantized back to float16 and combined with the outlier results to form the final output (Section A.2).

More formally, the matrix multiplication can be described as:

$$\mathbf{C}_{f16} \approx \sum_{h \in O} \mathbf{X}^h_{f16} \mathbf{W}^h_{f16} + \mathbf{S}_{f16} \cdot \sum_{h \notin O} \mathbf{X}^h_{i8} \mathbf{W}^h_{i8} \quad (4)$$

where C_{f16} represents the output tensor in float16, (X^h_{f16}, W^h_{f16}) represent the float16 input and weight for outliers, S_{f16} is the denormalization term for int8 inputs and weights, and (X^h_{i8}, W^h_{i8}) represent the int8 input and weight for non-outliers. The full procedure is shown in Algorithm 1.

C. Baseline Methods

We compare our attack to two baselines methods: Sponge Examples [11] and Standard PGD [7].

The Sponge Examples loss function can be formalized in the following way:

$$\mathcal{L}_{\text{Sponge Examples}} = -\sum_{a \in A} \|a_l\|_2 \quad (5)$$

where A is the set of all activation values, and a_l represents the activations of layer l.

The Standard PGD loss function can be formalized in the following way:

$$\mathcal{L}_{\text{Standard PGD}} = -\sum_{m=1}^{M} y_m \log(\hat{y}_m) \quad (6)$$

where y_m denotes the ground-truth and $\hat{y}_m$ the predicted score for class m.

D. Quantized Models Performance

Since the quantization technique used in this paper was originally proposed for the NLP domain, we first verified the performance on the quantized models. We compare the accuracy performance of the quantized and non-quantized models across 5,000 images randomly selected from the ImageNet-1K dataset's validation set. The results show that the quantized model exhibits the same level of performance as the non-quantized model. For ViT, the accuracy is 80.51% and 80.59% for the quantized and non-quantized models, respectively. For DeiT, the accuracy is 80.47% and 80.57% for the quantized and non-quantized models, respectively. This demonstrates that the LLM.int8() technique is applicable for the vision domain.

E. Models Architectures

In Section 5.2.5 we introduce four models differing in their tasks.

Open-World Localization (OWLv2) [8] is a zero-shot text-conditioned object detection model that combines visual and text features using a CLIP backbone [9], efficiently performing open-vocabulary object detection. In our evaluation, we use the base-size version. For the attack, we use

500 images from the COCO dataset [6], and the corresponding texts are simply the concatenation of the class categories in each image.

You Only Look at One Sequence (YOLOS) [4] is a transformer-based object detection model. It employs a bipartite matching loss and the Hungarian algorithm, resulting in a simplified yet effective approach for object detection tasks. In our evaluation, we use the small-size version. For the attack, we use 500 images from the ImageNet dataset [6].

Generative Image-to-text Transformer (GIT) [12] is a decoder-only transformer for vision-language tasks that utilizes CLIP's vision encoder for integrated text and image processing. In our evaluation, we use the base-size version. During our evaluation, we focus on its image captioning abilities, in which an image is provided as input and the model is asked to generate a corresponding caption. For the attack, we use 500 images from the ImageNet dataset [6].

Whisper [10] is a sequence-to-sequence model for automatic speech recognition (ASR) and speech translation. Since Whisper is fully differentiable, similar to the computer vision models above, it is also vulnerable to adversarial perturbations that tamper with the input audio by adding an imperceptible, carefully-crafted pattern. To demonstrate that our attack also extends to the audio domain, we employ a technique similar to the one employed for the computer vision models, in order to generate the adversarial examples. In our evaluation, we use the tiny-size version. We use 500 utterances from the FLEURS dataset [1] and examine our attack's effect on a quantized Whisper model.

F. Ablation Studies

F.1. Effect of accuracy loss component (λ)

In Section 4.2, we discuss the effect of the different components of the loss function (Equation 7 in the paper). The results presented in Table 1 show the impact to the different metrics of the values assigned to the λ weight. We found that increasing λ generally enhances the model's accuracy. For instance, when λ increased from 0 to 250, the Top1 accuracy improved from 26% to 94%, and the Top5 accuracy saw a similar rise from 47% to 99.8%. However, this is at the expense of "overall performance" (processing time, memory usage, energy consumption, and the number of outliers). Specifically, higher λ values tend to decrease outliers, as evidenced in Table 1, when $\lambda = 0$ the percentage change of the outliers is 1757.80% compared to 1611.1% when $\lambda = 250$. On the other hand, when setting $\lambda = 500$, the model's accuracy remains nearly unchanged, reaching 97.8%.

Eventually, we selected the setting that optimally balanced these considerations and chose $\lambda = 50$. This configuration led to a satisfying balance of a 90% accuracy rate

λ	%Energy [mJ]	%Throughput [ms]	%Memory [Mbits]	%Outliers	Accuracy	
					Top1	Top5
0	7.60%	9.00%	18.10%	1757.80%	26%	47%
25	7.50%	8.90%	17.60%	1725.60%	86%	97%
50	6.90%	8.80%	17.20%	1681.20%	90%	99.5%
75	6.70%	8.70%	17.20%	1667.00%	90%	99.5%
100	6.50%	8.50%	16.80%	1643.30%	91%	99.5%
250	6.40%	8.40%	16.50%	1611.10%	94%	99.8%
500	6.10%	7.20%	16.30%	1518.70%	97.8%	99.8%

Table 1. Comparative analysis of the weight values for the classification loss component on the ViT model. The values (except accuracy) represent the percentage change between the perturbed and original images. Grey line indicates the chosen value.

(Top1) and an outlier percentage of 1681.20% with the Top5 accuracy reaching 99.5%, reflecting our goal to enhance model accuracy without excessively compromising the attack's performance. This decision was driven by our aim to strike a balance between improving accuracy and overall attack performance as can be seen in Figure 2.

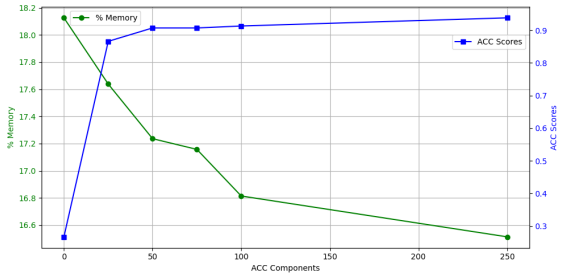


Figure 2. The results of our ablation study examining the effect of the λ value on the memory usage for the ViT module.

F.2. Hyperparameters ϵ, k, and x_{target}

In Section 4, we presented our attack's methodology, which consists of several hyperparameters. In this section, we present the effect of the different values assigned to these parameters.

F.2.1 Target Value x_{target}

In Equation 9 in the paper, we describe the quantization loss component ($\mathcal{L}_{\text{quant}}$), which consists of the target value x_{target}. The rationale behind choosing a value above the threshold is to guide the quantization process in a direction that enhances the attack performance. From Table 2, we see a clear trend where the overall performance of the attack improves as the x_{target} increases. When $x_{\text{target}} = 70$, we observe a good balance across various metrics. However, as the x_{target} further increases, for example when x_{target} is set at 100, there is a slight decrease in accuracy, despite similar trends in the other metrics. Taking the x_{target} to an extreme (1M), leads to a sharp decrease in accuracy, highlighting a clear point where increasing the x_{target} no longer benefits the model's performance, negatively affecting the performance.

x_{target}	%Energy [mJ]	%Throughput [ms]	%Memory [Mbits]	%Outliers	Accuracy
7	6.80%	6.80%	14.32%	1432.10%	98%
25	8.10%	7.60%	16.61%	1651.00%	96%
70	6.90%	8.80%	17.20%	1681.20%	90%
100	7.60%	7.30%	17.43%	1675.80%	90%
inf	6.60%	7.20%	16.15%	1612.10%	42%

Table 2. Comparative analysis of ViT model for different x_{target} values. The values (except accuracy) represent the percentage change between the perturbed and original images. Grey line indicates the chosen values.

F.2.2 Top-K Elements

In our study, we have conducted experiments with a range of K values to assess their influence on the performance of our attack. As mentioned previously, we consider $\mathbf{X}_l \in \mathbb{R}^{s\times h}$ as the input hidden state for the l^{th} quantized layer, with the bit-precision notation omitted for simplicity.

For each input matrix $\mathbf{X}_l$, we select the top K values, represented by $\mathbf{X}_l^{\text{top-k}} \in \mathbb{R}^{K\times h}$. The objective is to adjust these values towards a specified target value x_{target}, such that $x_{\text{target}} > \tau$.

The loss function for an individual layer is thus formulated as:

$$\mathcal{L}_{\text{single-q}}(\mathbf{X}_{l_q}^{\text{top-}K}) = \frac{1}{Kh} \sum_{\substack{x_h \in \mathbf{X}_{l_q}^{\text{top-}K} \\ |x_h|<\tau}} \left(|x_h| - x_{\text{target}}\right)^2 \quad (7)$$

In Table 3, we present a comparative analysis of the ViT model for different K values. The table highlights how the choice of K value affects the different metrics. Notably, $K = 4$ has better performance in three out of five metrics and ranked second in the remaining two. This suggests that $K = 10$ is a viable alternative under certain conditions. When considering the extremes, $K = 1$ demonstrates the highest accuracy and shows respectable results in processing time and memory usage, indicating its efficiency in specific contexts. However, a contrasting scenario is observed when all values from each column are considered $K = s$, leading to a noticeable decline in overall performance metrics. This indicates that while expanding the scope to $K = s$ increases comprehensiveness, it adversely affects efficiency and accuracy.

K	%Energy [mJ]	%Throughput [ms]	%Memory [Mbits]	%Outliers	Accuracy
1	8.11%	8.03%	17.10%	1529.27%	94%
4	6.90%	8.80%	17.20%	1681.20%	90%
10	5.98%	6.43%	15.40%	1547.86%	96%
all	3.55%	2.92%	9.23%	806.35%	73%

Table 3. Comparative analysis of ViT model for different K values. The values (except accuracy) represent the percentage change between the perturbed and original images. Grey line indicates the chosen values.

F.2.3 Perturbation Bound ϵ

The perturbation bound ϵ is the specified upper limit on the perturbation values in the PGD attack. In Table 4, we present the results for different ϵ values using the ∞-norm PGD attack. Predictably, when $\epsilon = \frac{8}{255}$, it results in fewer outliers (1314.50%) and overall shows less impact of the attack's performance, as it imposes a tighter constraint on the image perturbation limit. Conversely, a higher epsilon value leads to more outliers (2122.2%) and the best overall performance, however with a trade-off that leads to a visually perceptible perturbation. However, as the ϵ value increases the attacked image is more perceptible and less stealthy, which led us to choose a value that is in between the extreme values that still maintains high attack performance.

ϵ	%Energy [mJ]	%Throughput [ms]	%Memory [Mbits]	%Outliers	Accuracy
8/255	6.40%	6.50%	13.10%	1314.50%	90%
12/255	6.80%	8.00%	15.70%	1529.20%	90%
16/255	6.90%	8.80%	17.20%	1681.20%	90%
32/255	8.20%	10.90%	21.70%	2122.20%	88%

Table 4. Comparative analysis of ViT model for different epsilons parameters. The values (except accuracy) represent the percentage change between the perturbed and original images.

G. Additional Experiments

G.1. Loss on LayerNorm

In section 5.3 we provide a thorough analysis of how the normalization layers within transformer blocks impact the effectiveness of our attack. In an attempt to enhance our attack's performance, we introduced an additional component to our loss function aimed at directly minimizing the input's mean to the normalization layer. We replace Equation 7 in the paper with:

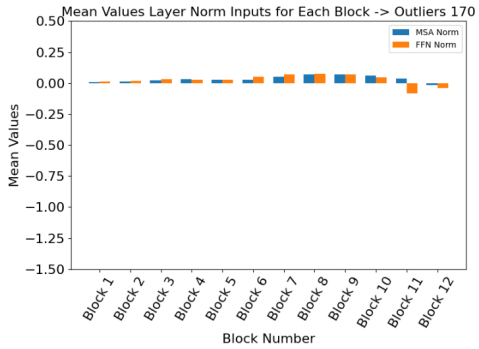
$$\mathcal{L} = \mathcal{L}_{\text{quant}} + \lambda_1 \cdot \mathcal{L}_{\text{cls}} + \lambda_2 \cdot \mathcal{L}_{\text{LayerNorm}} \quad (8)$$

where $\mathcal{L}_{\text{quant}}, \mathcal{L}_{\text{cls}}$ are described in Section 4.2 and $\mathcal{L}_{\text{LayerNorm}}$ is defined as follow:

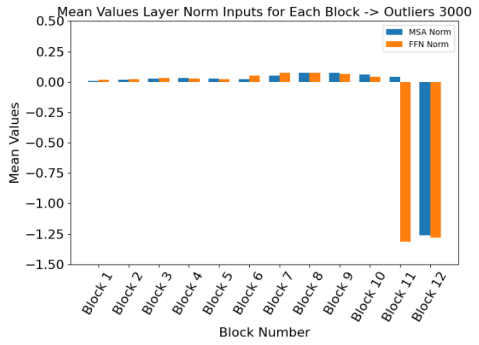
$$\mathcal{L}_{\text{LayerNorm}} = \frac{1}{L_{ln}} \sum_{i=1}^{L_{ln}} \mathrm{E}[\mathbf{X}_{l_{ln}}] \quad (9)$$

where $\mathbf{X}_{l_{ln}}$ denotes the input to to the $l_{ln}-th$ normalization layer, and L_{ln} denotes the total number of normalization layers.

As noted in the paper, this additional component did not improve our attack's performance. This phenomena can be explained by observing the inputs' mean values when using the original loss function (Equation 7 in the paper). From Figure 3 we can see that the original attack already affects the mean values in the affected layers, showcasing that an additional loss component aimed at effecting these values is redundant.



(a) Mean values of normalization layers' inputs per block for a clean image.



(b) Mean values of normalization layers' inputs per block for an attacked image.

Figure 3. Comparison of mean values of normalization layers' inputs per block between a clean and an attacked image when using the original loss (Equation 7 in the paper).

G.2. Transferability & Ensemble Across Different Quantization Methods

In Table 5, we analyze the transferability of our attack across different quantization techniques, focusing on model integrity (accuracy). We train perturbations on a model with one quantization technique and test them on the same model with a different quantization technique applied. Additionally, we demonstrate the benefits of using an ensemble strategy to enhance the robustness of the attack. Our results show that perturbations exhibit some degree of transferability. For instance, perturbations trained on static PTQ techniques (such as RepQ-ViT or PTQ4ViT) effectively transfer across these methods. However, perturbations trained on dynamic PTQ techniques do not transfer as well to static PTQ techniques and vice versa. The ensemble strategy, which assumes the attacker has partial knowledge of the set of possible quantization techniques, yields excellent results, further enhancing the attack's effectiveness.

H. Defense

In the countermeasures section, we discuss limiting the use of high-precision multiplications as a defense mechanism. We evaluated this approach on 10K images from the ImageNet validation set.

To limit the number of potential outliers, we first identified columns containing outliers. Then, we randomly selected a subset of these outlier columns based on a predefined ratio in the range 1-10% (the ratio is measured per layer). Finally, all remaining columns (not included in the subset) were clipped below the model's threshold τ.

As shown in Table 6a, the threshold mechanism does not negatively effect the model's accuracy when using a portion higher than 3%. On the other hand, in Table 6b we can see that when a defense is not used (outliers limit is set at 100%) the number of outliers is substantially higher compared to those when a defense is applied. This approach underscores the utility of our approach in enhancing the model robustness against availability-oriented attack on the dynamic quantization mechanism. Above all, while this defense demonstrates a solid mitigation, in real-world applications, the threshold still requires manual tuning according to the system's requirements.

Attack Type	Trained on: PTQ4ViT	Trained on: Repq-ViT	Trained on: LLM.int8()	Tested on	Accuracy
Transfer	✔			PTQ4ViT	10%
				Repq-ViT	62%
				LLM.int8	82%
		✔		PTQ4ViT	56%
				Repq-ViT	10%
				LLM.int8	81%
			✔	PTQ4ViT	84%
				Repq-ViT	87%
				LLM.int8	26%
Ensemble	✔		✔	PTQ4ViT	22%
				Repq-ViT	69%
				LLM.int8	28%
		✔	✔	PTQ4ViT	54%
				Repq-ViT	10%
				LLM.int8	26%
	✔	✔		PTQ4ViT	22%
				Repq-ViT	10%
				LLM.int8	81%
	✔	✔	✔	PTQ4ViT	25%
				Repq-ViT	10%
				LLM.int8	26%

Table 5. Transferability and ensemble analysis across different quantization techniques when using the ViT model.

I. Practical Implications

Following the practical implications discussed in [5], we consider two different scenarios in which our attack is applicable in:

- **Attacks on cloud-based IoT applications**: Typically, cloud-based IoT applications (*e.g.*, virtual home assistants, surveillance cameras) run their DNN inferences in the cloud. This exclusive reliance on cloud computing places the entire computational load on cloud servers, leading to heightened communication between

% Outliers Limit	Accuracy
1%	79.49
2%	79.67
3%	80.62
4%	80.75
5%	80.86
6%	80.86
7%	80.88
8%	80.88
9%	80.89
10%	80.89
100%[1]	80.89

(a) Clean images.

% Outliers Limit	Outliers
1%	178
2%	285
3%	380
4%	450
5%	503
6%	560
7%	646
8%	662
9%	737
10%	770
100%[1]	4000

(b) Adversarial images.

Table 6. Comparison of accuracy and outliers with and without defense at various outlier limits. This defensive measure successfully maintains high accuracy levels on clean images (a) while substantially reducing the occurrence of outliers on adversarial images (b). [1]Setting the limit at 100% is equivalent to disabling the defense mechanism.

these servers and IoT devices. Recently, there has been a trend for bringing computationally expensive models in the cloud to edge (*e.g.*, IoT) devices. In this setup, the cloud server exclusively handles complex inputs while the edge device handles "easy" inputs. This approach leads to a reduction in computational workload in the cloud and a decrease in communication between the cloud and edge. Conversely, an adversary can manipulate simple inputs into complex ones by introducing imperceptible perturbations, effectively bypassing the quantization mechanism. In this scenario, a defender could implement denial-of-service (DoS) defenses like firewalls or rate-limiting measures. In such a setup, the attacker might not successfully execute a DoS attack because the defenses regulate the communication between the server and IoT devices within a specified threshold. However, despite this, the attacker still manages to escalate the situation by: *(i)* heightening computational demands at the edge (by processing complex inputs at the edge); and *(ii)* increasing the workload of cloud servers through processing a greater number of samples.

- **Attacks on real-time DNN inference for resource- and time-constrained scenarios**: Quantization techniques can be harnessed as a viable solution to optimize real-time DNNs inference in scenarios where resources and time are limited. For example, in real-time use cases (*e.g.*, autonomous cars) where throughput is a critical factor, an adversary can potentially violate real-time guarantees. In another case, when the edge-device is battery-powered an increased energy consumption can lead to faster battery drainage.

Following the possible implications described above, we also discuss concrete use cases:

- **Surveillance cameras scenario**: consider a cloud-based IoT platform that uses quantized vision transformers to process and analyze images from a network of surveillance cameras. These cameras monitor various locations and send image data to a centralized cloud server for real-time analysis and anomaly detection.
  Attack impact: an attacker could exploit the quantization mechanism to force the model to use high-bit operations, increasing computational overhead and latency. This could lead to delays in detecting anomalies, potentially allowing security breaches to go unnoticed for longer periods. In a high-security environment, such a delay could have severe consequences, compromising the safety and security of the monitored locations.
- **Edge devices scenario**: edge devices, such as smart cameras or drones, often perform real-time deep neural network (DNN) inference to analyze data locally before sending summarized information to the cloud. These devices frequently use acceleration techniques to optimize performance and reduce energy consumption.
  Attack impact: by deploying an adversarial attack on the quantization mechanism, an attacker could significantly increase the computational load on the edge device. This could lead to rapid battery depletion, overheating, and reduced operational efficiency. In critical applications, such as search and rescue missions or autonomous vehicle navigation, such an attack could incapacitate the device, leading to mission failure or safety hazards.
- **Autonomous drones scenario:** consider autonomous drones that use quantized vision transformers for navigation, object detection, and environmental analysis. These drones are used in various applications, including delivery services, agriculture, and surveillance.
  Attack impact: an adversarial attack on the quantization mechanism could overload the drone's computational resources, leading to navigation errors, reduced flight time, or complete system failure. Such disruptions could result in operational inefficiencies or accidents, especially in complex environments where precise navigation is crucial.
  Autonomous vehicles scenario: consider autonomous vehicles that rely on quantized vision transformers for tasks such as object detection, lane keeping, and collision avoidance. These systems must operate in real-time to ensure safe and efficient navigation on roads.
  Attack impact: an adversarial attack on the quantization mechanism could introduce computational delays or errors in object recognition and decision-making processes. This might result in delayed or incorrect responses to environmental stimuli, such as failing to recognize a pedestrian crossing or misinterpreting traffic signals. Such disruptions could lead to traffic accidents, endangering the

safety of passengers, pedestrians, and other road users. In scenarios where quick decision-making is crucial, even a slight delay could have catastrophic consequences.

- **Wearable health monitors scenario**: consider wearable health monitors that use quantized vision transformers to analyze physiological data, such as heart rate, activity levels, and sleep patterns. These devices provide real-time health insights and alerts to users.
  Attack impact: an adversarial attack on the quantization mechanism could compromise the device's ability to process data efficiently, leading to incorrect health metrics and delayed alerts. This could affect the user's health management, potentially missing critical health events that require immediate attention.

J. Attack Visualizations

In Figure 4, we visualize the adversarial examples and their corresponding perturbation. In Figure 5, we visualize the adversarial examples from the baselines and the QuantAttack different attack variants.

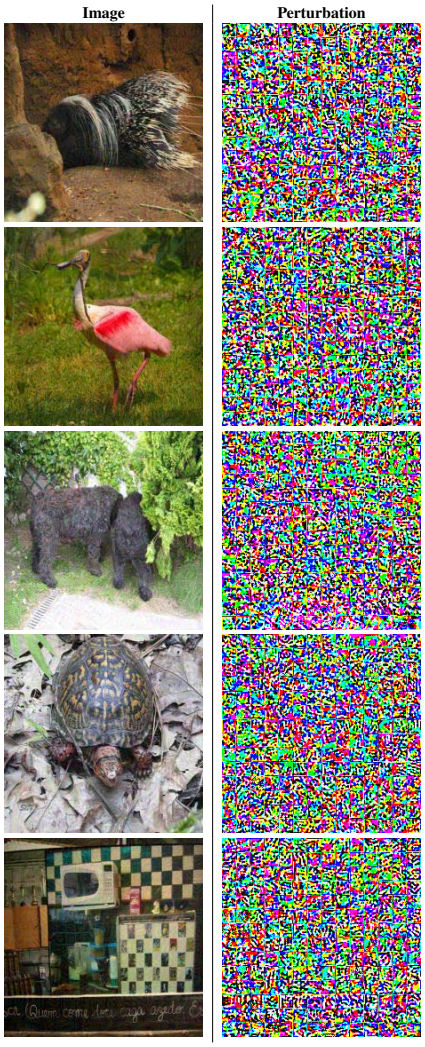


Figure 4. Adversarial examples with their corresponding perturbation.

Figure 5. Adversarial examples from the baselines and our QuantAttack attacks. The leftmost column shows the clean images. In the next three columns, we show adversarial examples from random, standard PGD and sponge examples, respectively. The last three columns include adversarial examples from the different QuantAttack variants.

References

[1] Alexis Conneau, Min Ma, Simran Khanuja, Yu Zhang, Vera Axelrod, Siddharth Dalmia, Jason Riesa, Clara Rivera, and Ankur Bapna. Fleurs: Few-shot learning evaluation of universal representations of speech. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 798–805. IEEE, 2023. 3

[2] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. A gentle summary of llm.int8(): zero degradation matrix multiplication for large language models, 2022. 1

[3] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022. 2

[4] Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu. You only look at one sequence: Rethinking transformer in vision through object detection. *Advances in Neural Information Processing Systems*, 34:26183–26197, 2021. 3

[5] Sanghyun Hong, Yiğitcan Kaya, Ionuţ-Vlad Modoranu, and Tudor Dumitraş. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. *arXiv preprint arXiv:2010.02432*, 2020. 5

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 3

[7] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 2

[8] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. *arXiv preprint arXiv:2306.09683*, 2023. 2

[9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[10] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023. 3

[11] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European symposium on security and privacy (EuroS&P)*, pages 212–231. IEEE, 2021. 2

[12] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language. *ArXiv*, abs/2103.01260, 2021. 3