Overload: Latency Attacks on Object Detection for Edge Devices

Erh-Chung Chen [1] [*] Pin-Yu Chen [2] I-Hsin Chung [2] Che-Rung Lee [1]

National Tsing Hua University [1] IBM Research [2]

s107062802@m107.nthu.edu.tw pin-yu.chen@ibm.com

cherung@cs.nthu.edu.tw ihchung@us.ibm.com

A. Comprehensive Analysis of NMS Algorithm

As described in earlier section, the steps of NMS can be divided into four major tasks:

1. Filtering low-confidence objects;
2. Sorting candidates by probabilities;
3. Calculating pairwise IoU scores;
4. Pruning inactive objects.

Let $\mathcal{C}$ be the set of objects fed into NMS. In this first step, the filtering step scans all objects in the set and deletes low-confidence objects, resulting in linear time complexity $O(|\mathcal{C}|$. In the second step, the time complexity of sorting is well known as $O(|\mathcal{C}|\log(|\mathcal{C}|))$. In the third step, NMS constructs a $|\mathcal{C}| \times |\mathcal{C}|$ matrix which stores the pairwise comparison results. IoU score computes the area of overlap between a pair of selected objects requiring a constant number of float operations. Therefore, the time complexity is $O(|\mathcal{C}| \times |\mathcal{C}|)$. In the last step, the major goal is to prune unqualified objects based on the IoU scores. The details are listed in Algorithm 1, where $\mathbf{S}_{|\mathcal{C}|\times|\mathcal{C}|}$ stores IoU scores. An object is marked duplicated if the IoU score $\mathbf{S}[i][j]$ is greater than the NMS threshold N_t. $\mathbf{r}$ is a utility array tacking whether the objects are marked. To obtain the worst case, we assume that no objects are duplicated, making the condition in line 5 always satisfied. As a result, the algorithm can be simplified to a two-layer loop that traverses half elements in the matrix $\mathbf{S}$. Therefore, the overall time complexity is independent of the rate of survival objects or the properties of boxes although some elements are marked removable during the pruning procedure.

Algorithm 1 NMS Pruning

```
1:  n ← |C|
2:  r = 0
3:  D ← {}
4:  for i = 1 to n do
5:      if r[i] ≠ True then
6:          D ← D ∪ {i}
7:          for j = i + 1 to n do
8:              r[j] = bool(r[j] + S[i][j] > N_t)
9:          end for
10:     end if
11: end for
```

B. Ablation Study

B.1. Impact of Different Objective Functions

In Section **??**, we mentioned that any monotonic increasing function can be used as the loss function. In this experiment, we evaluated the performance of four qualified functions: $\log(x)$, $\tanh(x)$, $x^2/2$, and $-\log(1-x)$. $\log(x)$. The derivative of $\tanh(x)$ is 1 at $x = 0$ and smoothly decreases as x increases. $x^2/2$ is a convex function with its minimum point at $x = 0$, and its derivative gradually increases. The limit of $-\log(1-x)$ as x approaches 1 from the left is positive infinity.

Table 1 shows that $x^2/2$ and $-\log(1-x)$ result in significantly fewer total objects compared to the other functions. To explain this phenomenon, we divide the objects predicted by the model into two sets: $\mathcal{S}^+$ and $\mathcal{S}^-$, where $\mathcal{S}^+ = \{x|\text{conf}(M(x)i) > T\text{IoU}\}$ and $\mathcal{S}^- = \{x|\text{conf}(M(x)i) \leq T\text{IoU}\}$. The original objective of the loss function is to maximize the confidence of individual boxes in set $\mathcal{S}^-$, increasing the total number of objects fed into NMS. However, $-\log(1-x)$ tends to increase the confidence of boxes in set $\mathcal{S}^-$ due to the rapid growth of its derivative when x is close to 1.0. As a result, although PGD maximizes the objective function defined in (**??**), $-\log(1-x)$ yields the worst performance. The behavior of $x^2/2$ is similar, as its derivative increases gradually. Therefore, an effective objective function should not only be a monotonic increasing function but also have a monotonically decreasing derivative.

[*]The primary research and contribution for this work were conducted during a visit to IBM Research.

Percentile	$F = \log(x)$			$F = \tanh(x)$			$F = x^2/2$			$F = -\log(1-x)$		
YOLOv5s												
	objects	boxes	time	objects	boxes	time	objects	boxes	time	objects	boxes	time
min	3228	392	52.8	9269	1081	25.0	3035	228	17.8	4556	511	19.3
0.10	18900	1570	52.8	19546	1829	45.9	5708	357	19.6	9553	555	23.9
0.25	23494	2817	91.8	17140	1471	148.3	6644	427	20.4	10079	647	25.0
0.50	22273	2617	167.4	19677	1961	192.1	6934	413	21.4	10955	579	26.6
0.75	22660	1630	208.5	21559	1448	227.1	8681	502	23.0	11529	644	27.9
0.90	22951	3162	241.2	22231	2585	241.6	10617	548	26.0	12422	790	29.1
max	23922	1386	241.2	22944	1382	253.0	13104	824	30.2	12632	813	39.7
YOLOv5n												
Percentile	$F = \log(x)$			$F = \tanh(x)$			$F = x^2/2$			$F = -\log(1-x)$		
	objects	boxes	time	objects	boxes	time	objects	boxes	time	objects	boxes	time
min	2890	763	13.3	4134	579	14.3	3435	210	13.0	3060	294	12.9
0.10	9012	1145	19.5	11950	1103	23.9	5295	339	14.3	8009	573	17.2
0.25	12985	1474	26.0	14064	1194	27.6	6074	355	15.0	8751	600	18.2
0.50	16755	1737	34.3	16128	1319	32.0	6920	444	15.8	9631	697	19.5
0.75	18910	1766	69.7	18276	1206	37.2	8079	448	17.1	10909	656	21.2
0.90	20901	1649	135.5	17507	1460	117.9	9361	612	18.7	11746	684	22.6
max	22303	1472	231.6	20902	1238	206.7	9946	585	25.3	15479	983	30.4

Table 1. The total elapsed time using different objective function on NVIDIA Jetson NX

B.2. Spatial Attention Evaluation

This experiment evaluates the influence of spatial attention. Table 2 shows the experimental results on Nvidia Jetson NX, where *SA* means the adversarial attack with the spatial attention and *PGD* means the native implementation of PGD. As can be seen, one can find that the spatial attention method can generate approximately 2,000 or more objects for the YOLOv5s and YOLOv5n models. This increase in object count is due to the iterative generation of objects from regions with fewer objects, facilitated by the proposed spatial attention technique.

Table 3 offers a performance comparison with different grid sizes, where *Grid* signifies that the image is tiled into $k \times k$ grids. In the case of 1×1 tiling, the configuration reverts to the original PGD attack, where all pixels are part of the same grid, sharing identical weights. Upon introducing spatial attention, specific areas can be highlighted, resulting in a performance gain. However, when the image is divided into a 20×20 grid, some tiles remain unhighlighted, hindering effective attacks on those tiles. The experimental findings suggest that an optimal configuration for spatial attention is around 5×5.

These results demonstrate the effectiveness of the spatial attention technique in generating a larger number of objects. The comparison between SA and PGD sheds light on the potential vulnerabilities and limitations of the YOLOv5 models when exposed to adversarial attacks with spatial attention.

C. Latency Attacks on Various Models

This experiment aims to comprehensively evaluate the performance of the proposed latency attack on various models. To extend our analysis, we conducted additional experiments on YOLOv3 model and two latest YOLO models, namely YOLOv7 and YOLOv8. Tables 4 presents the results obtained from these models, where the elapsed time was not measured as these models are not fully optimized for edge devices.

The obtained results reveal the effectiveness of our attack. At the 50th percentile, our attack generates 20,578 objects for YOLOv7 and 23,163 objects for YOLOv7-tiny. Similarly, for YOLOv8, our attack generates 8,313 objects at the 50th percentile. These numbers are in comparison to the maximum number of objects predicted by YOLOv7 (25,200 objects) and YOLOv8 (8,400 objects). The significant number of objects generated by our attack demonstrates its potency. Consequently, our findings indicate that both YOLOv7 and YOLOv8 models are susceptible to latency attacks. It is worth mentioning that as the total number of objects increases, the elapsed time during the attack is expected to rise.

We argue that object detection models with different architectures are also vulnerable to latency attacks. However, it is important to note that the proposed objective in (**??**)

Percentile	YOLOv5s						YOLOv5n					
	SA			PGD			SA			PGD		
	objects	boxes	time	objects	boxes	time	objects	boxes	time	objects	boxes	time
min	4098	1587	18.7	3228	392	18.3	5893	1131	14.7	2890	763	14.8
0.10	23112	2034	122.2	18900	1570	117.8	11925	1421	22.6	9012	1145	27.1
0.25	24248	4653	179.6	23494	2817	161.9	16405	2303	39.8	12985	1474	35.9
0.50	24000	2250	217.8	22273	2617	209.6	22363	2315	128.9	16755	1737	59.3
0.75	24335	4514	264.2	22660	1630	232.2	20923	2372	208.3	18910	1766	175.2
0.90	24778	2340	278.4	22951	3162	246.2	21947	2659	226.7	20901	1649	210.8
max	24859	2363	293.8	23922	1386	270.5	23396	2502	258.7	22303	1472	244.6

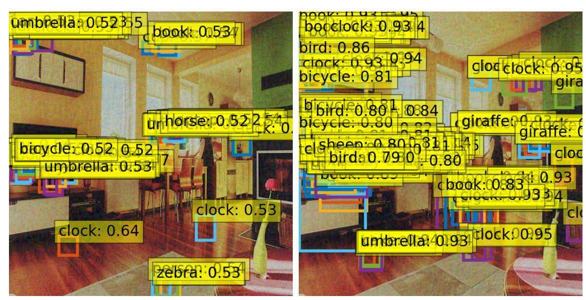
Table 2. Results of spatial attention evaluation

Percentile	YOLOv5s						YOLOv5n					
	Grid=1 × 1		Grid=5 × 5		Grid=20 × 20		Grid=1 × 1		Grid=5 × 5		Grid=20 × 20	
	objects	boxes	objects	boxes	objects	boxes	objects	boxes	objects	boxes	objects	boxes
min	3421	507	7465	1235	6220	1062	3421	507	7465	1235	6220	1062
0.10	6359	983	10739	1405	8193	1412	6359	983	10739	1405	8193	1412
0.25	10696	1203	11197	1535	9706	1595	10596	1203	11197	1535	9706	1595
0.50	12557	1485	16769	1770	21520	1827	12557	1485	16768	1770	21520	1827
0.75	17671	1848	17461	2110	19781	2117	17671	1848	17461	2110	19781	2117
0.90	20647	2155	17551	2305	18491	2315	20647	2155	17551	2305	18491	2315
max	21515	2723	20202	2688	19949	2638	21514	2723	20202	2688	19949	2638

Table 3. Performance comparison with different grid size.

is specifically designed for YOLO series and may not be the optimal objective for other architectures. To investigate this further, we conducted a small experiment using the SSD model [?]. Our observations revealed that the number of objects generated using our attack ranged from approximately 300 to 1,000. Since SSD normalizes the probabilities using the softmax function, ensuring that the sum of probabilities for all classes is 1.0, the attack's performance is highly influenced by the image's characteristics and the target class. These findings suggest that SSD models are indeed vulnerable to latency attacks, but further improvements can be made.

We would like to emphasize that the proposed loss function defined in (4) is tailored specifically for the YOLO series. As detailed in the background section, each model employs its own unique algorithm to process the locations and probabilities of the output objects. For instance, two-stage detectors divide the task into two phases, resulting in the inability to directly estimate the gradient. Some detectors calibrate the locations based on predefined anchors. Nevertheless, there could be significant benefits in adjusting spatial importance. Fig. 1 and 2 illustrate the outputs of adversarial examples generated by Retinanet [?] and FCOS [?], respectively. As can be seen, the predictions of adversarial examples produced by the standard PGD attack tend to



(a) (b)

Figure 1. The outputs of the adversarial examples by Retinanet. 1a and 1b are generated by the normal PGD attack and Overload attack, respectively.

cluster within a small region while the proposed attack ensures a more widespread distribution of objects in the spatial domain. We believe the spirit of Overload is applicable to most detectors integrated with NMS, but the implementation details should be further studied.

D. Transfer Attack Evaluation

The transferability of adversarial attacks, where an attack crafted for one model can be successfully applied to a

Percentile	YOLOv7				YOLOv7-tiny			
	Adversarial Examples		Original Examples		Adversarial Examples		Original Examples	
	objects	boxes	objects	boxes	objects	boxes	objects	boxes
min	18310	3139	0	0	18884	2398	0	0
0.10	19897	3765	6	1	23018	3259	97	9
0.25	20119	4069	18	2	23068	3390	32	3
0.50	20578	3681	30	3	23163	3313	12	1
0.75	21310	4991	10	2	23083	3519	28	3
0.90	20359	3912	27	3	23007	3708	74	9
max	22072	4243	12	1	22971	3399	36	2

Percentile	YOLOv8s				YOLOv8n			
	Adversarial Examples		Original Examples		Adversarial Examples		Original Examples	
	objects	boxes	objects	boxes	objects	boxes	objects	boxes
min	8244	1188	0	0	7881	1652	0	0
0.10	8273	1273	46	6	8044	1747	56	9
0.25	8250	1098	28	4	7839	1530	10	1
0.50	8313	1212	76	14	7759	1884	7	1
0.75	8333	1326	133	21	7886	1918	1	1
0.90	8333	1141	51	5	8017	1658	121	23
max	8347	1268	37	5	7968	1836	85	16

Percentile	YOLOv3				YOLOv3-tiny			
	Adversarial Examples		Original Examples		Adversarial Examples		Original Examples	
	objects	boxes	objects	boxes	objects	boxes	objects	boxes
min	11879	2107	0	0	5760	1554	0	0
0.10	13198	2558	76	7	5873	1674	0	0
0.25	13401	2565	62	6	5884	1619	56	6
0.50	14015	2562	134	9	5940	2061	30	3
0.75	14109	2707	205	17	5867	2380	125	24
0.90	14365	2839	84	7	5981	2528	16	4
max	14172	2651	39	3	5870	1543	21	2

Table 4. Latency Attacks on YOLOv7, YOLOv8, and YOLOv3 models

Percentile	YOLOv5s						YOLOv3			
	Adversarial Examples			Original Examples			Adversarial Examples		Original Examples	
	objects	boxes	time	objects	boxes	time	objects	boxes	objects	boxes
min	10778	1541	11.1	0	0	14.1	8443	1107	0	0
0.10	15771	1897	43.8	40	3	16.4	13138	1518	27	4
0.25	18167	2124	96.2	47	4	16.4	14091	1563	220	18
0.50	22755	2392	132.7	28	3	16.4	14796	1629	49	5
0.75	20338	1844	170.4	3	1	16.6	15418	1661	44	3
0.90	22384	2795	248.3	234	21	16.7	16027	1656	63	3
max	23579	1580	252.9	212	21	16.9	17684	1785	12	1

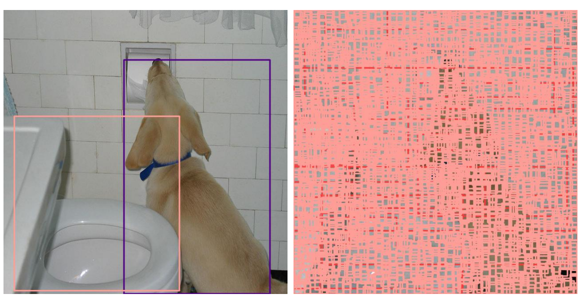
Table 5. Results of the ensemble attack

different victim model, is a common phenomenon in image classification tasks. However, when testing the transferability among different models in the YOLOv5 family for the latency attack, we did not observe this property. One possible reason is that the object detection network utilizes the Feature Pyramid Network (FPN), which com-

(a) (b)

Figure 2. The outputs of the adversarial examples by FCOS. 2a and 2b are generated by the normal PGD attack and Overload attack, respectively.

(a) Original image (b) Adversarial image



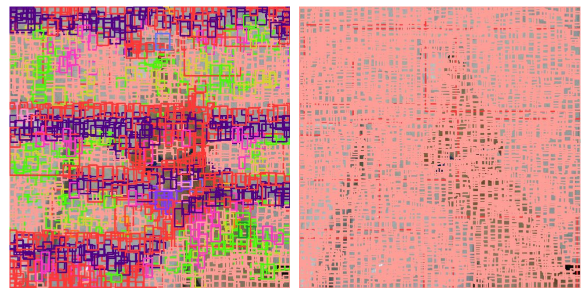
(c) The output of the original image (d) The output of the adversarial image

Figure 3. An example of Overload attack for object detection.

bines features extracted from both low-resolution and high-resolution sources. This integration of features from different networks may lead to divergence and hinder the transferability of the attack.

Nevertheless, we explored an alternative approach known as ensemble training to craft adversarial examples that can deceive multiple models. In the ensemble attack, gradients are obtained from either one candidate model or averaged across all candidate models in each attack step. We evaluated the performance of the ensemble attack using a combination of YOLOv3 and YOLOv5s models. Table 5 presents the results of the ensemble attack, omitting the execution times for YOLOv3 due to a technical issue with compiling the model to TensorRT format.

(a) Original image (b) Adversarial image of the ensemble attack



(c) The output of YOLOv3 (d) The output of YOLOv5s

Figure 4. An example of ensemble attack for object detection.

As observed, the ensemble attack successfully generates a significant number of objects for both YOLOv3 and YOLOv5s simultaneously. However, comparing these results with those in Table **??**, it appears that the strength of the ensemble attack is slightly weaker than that of the native attack. To provide visual context, Figure 3 and Figure 4 illustrate the original image, the corresponding adversarial image, and the results obtained from Overload and the ensemble attack, respectively.

These findings suggest that information from multiple models can be encoded within a single image, enabling the ensemble attack to deceive different object detection models. However, further investigation is needed to enhance the effectiveness and transferability of the ensemble attack against the latency-based defense.