

A Appendix

A.1 Prior arts tests

To validate the accuracy of prior arts of softmax approximation we have conducted several tests over DETR models. Since we are interested in the methods which are not using a division operation, we have considered prior arts where logarithmic transformation was applied. Thus, we have adopted available pre-trained models from <https://github.com/facebookresearch/detr> by substituting a conventional softmax layer by the methods described in [31], [34], [28], and [13]. All computations were conducted in FP32 precision.

A.1.1 Aggressive approximation

There are several methods, what strongly approximate the original softmax formula and which showed good results for conventional CV tasks: [34], [28], and [13]. Note, that Eq.(4) in [34] is mathematically equivalent to Eq.(9) in [13]; and Eq.(5) in [28] is mathematically equivalent to Eq.(18) in [13]. Unfortunately, if any of those methods is applied to DETR models, the quality of model collapsed completely, providing zero accuracy as shown in Figure 5.

```
Accumulating evaluation results...
DONE (t=5.77s).
IoU metric: bbox
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.000
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.000
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.000
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.001
```

Figure 5: Example of DETR (R50) model output due to aggressive approximation of softmax layer.

A.1.2 Exponentiation of logarithmic transformation

Mathematical formula for softmax computation by back exponentiation of logarithmic transformation is described as Eq.(2) in [31]:

$$\sigma(x_i) = \exp \left(x_i - \ln \left(\sum_{j=1}^N e^{x_j} \right) \right) \quad (i = 1, 2, \dots, N). \quad (11)$$

To mimic the usage of this method within 8-bit precision hardware, we have applied scaling with rounding in our code as below:

```
attn_output_weights = torch.round(Eq.(2) * prec)/prec,
```

with $\text{prec} = 2^w - 1$, where w is the number of bits of the used precision. Thus, for uint8 precision $w = 8$ and $\text{prec} = 255$. Note, that we have applied scaling with rounding only to the outer non-linear operation \exp , and if run on real hardware the same limitations would be applied to other inner operations (\ln , \exp), thus the accuracy will be even worse in the real case. In Table 3 there are shown the results of experiments. As it can be seen from the Table 3, the accuracy drop is high (3% to 32%), therefore we modify the original equation by bringing the input normalization by a \max -value as shown below and run another tests:

$$\sigma(x_i) = \exp \left(x_i - \max(x) - \ln \left(\sum_{j=1}^N e^{x_j - \max(x)} \right) \right) \quad (i = 1, 2, \dots, N). \quad (12)$$

Table 3: Experimental validation of prior arts over DETR models (Average Precision)

MODEL	METRIC	ORIGINAL MODEL, FP32	METHOD		ACCURACY DROP,	
			EQ.(2) IN [31]	EQ.(2)+ IN [31]	EQ.(2) IN [31], %	EQ.(2)+ IN [31], %
DETR (R50)	AP	0.42	0.349	0.395	7.1	2.5
	AP_50	0.624	0.564	0.607	6.0	1.7
	AP_75	0.442	0.350	0.41	9.2	3.2
	AP_S	0.205	0.113	0.175	9.2	3.0
	AP_M	0.458	0.373	0.431	8.5	2.7
	AP_L	0.611	0.579	0.592	3.2	1.9
DETR +DC5 (R50)	AP	0.433	0.248	0.304	18.5	12.9
	AP_50	0.631	0.44	0.519	19.1	11.2
	AP_75	0.459	0.24	0.303	21.9	15.6
	AP_S	0.225	0.039	0.093	18.6	13.2
	AP_M	0.473	0.234	0.325	23.9	14.8
	AP_L	0.611	0.473	0.512	13.8	9.9
DETR (R101)	AP	0.435	0.333	0.379	10.2	5.6
	AP_50	0.638	0.556	0.613	8.2	2.5
	AP_75	0.463	0.330	0.385	13.3	7.8
	AP_S	0.218	0.100	0.156	11.8	6.2
	AP_M	0.479	0.353	0.412	12.6	6.7
	AP_L	0.618	0.564	0.583	5.4	3.5
DETR +DC5 (R101)	AP	0.449	0.205	0.262	24.4	18.7
	AP_50	0.647	0.386	0.485	26.1	16.2
	AP_75	0.477	0.193	0.246	28.4	23.1
	AP_S	0.237	0.028	0.072	20.9	16.5
	AP_M	0.495	0.166	0.253	32.9	24.2
	AP_L	0.623	0.428	0.479	19.5	14.4

476 In Table 3 improved method is labeled as Eq.(2)+. Analysis of the table shows that even after usage
477 of max-based normalization as in Eq.(12), the accuracy drop still remains too high for practical
478 applications. The bold values in the table shows lowest accuracy drop per model per column. From
479 Table 3 an averaged accuracy drop w.r.t. to original FP32 models was calculated and shown in
480 Table 1.

481 A.2 Software models of the proposed methods

482 To validate the proposed methods by simulation, we have developed a software models in `pytorch`,
483 the pseudocode of which is shown below. These codes mimics the functionality of the proposed
484 HW-method to better understand the computational flow. This code in no matter is representing
485 performance, or latency of the proposed methods.

486 Algorithms 1 and 2 take several inputs:

- 487 • Data tensor x which is input values to be computed. The dimensions of the input tensor can
488 be any shape, in our code we resize it to 1-dimensional tensor first, and then restore it to
489 the original dimensions after the computation. In Algorithms 1 and 2 we assume that input
490 tensor x is already quantized by previous layer. However, our code allows quantization from
491 FP32 precision as well.
- 492 • Scale for de-quantization *scale*. As our code is designed to support different precisions, this
493 is the parameter to restore the softmax value back to floating-point precision. The value
494 of the scale for de-quantization depends on the selected precision (e.g., for int16 precision
495 $scale = 32, 768$). It can be selected as $scale = 1$ if no de-quantization is required (e.g., if
496 the next layer will compute the tensor in the same precision as softmax layer).
- 497 • For Algorithm 1
 - 498 – $LUT_{1/e}$ is 1D LUT where the reciprocal exponentiation values are stored for $\frac{1}{e^x}$
499 computation

Algorithm 1 REXP method

```
1: Input: data tensor  $x$ ,  $LUT_{1/e}$ ,  $LUT_{\alpha}$ ,  
   scale for de-quantization  $scale$   
2: Output: softmax tensor  $\sigma(x)$   
3: Normalize input data tensor:  $x \rightarrow (\max(x) - x)$   
4: for  $i = 1$  to  $size(x)$  do  
5:    $idx_{x_i} = MSB(x_i)$   
6:    $e^{x_i} = LUT_{1/e}[idx_{x_i},]$   
7: end for  
8: Accumulate normalization factor:  $\Sigma e^{x_i}$   
9:  $idx_{e^{x_i}} = MSB(e^{x_i}); idx_{\alpha} = MSB(\Sigma e^{x_i})$   
10: for  $i = 1$  to  $size(x)$  do  
11:    $\sigma(x_i) = LUT_{1/e}[idx_{e^{x_i}}] \cdot LUT_{\alpha}[idx_{\alpha}]$ .  
12: end for  
13: De-quantize  $\sigma(x) = \frac{\sigma(x)}{scale}$ 
```

Algorithm 2 2D LUT method

```
1: Input: data tensor  $x$ ,  $LUT_{exp}$ ,  $LUT_{\sigma}$ ,  
   scale for de-quantization  $scale$   
2: Output: softmax tensor  $\sigma(x)$   
3: Normalize input data tensor:  $x \rightarrow (x - \max(x))$   
4: for  $i = 1$  to  $size(x)$  do  
5:    $idx_{x_i} = MSB(x_i)$   
6:    $e^{x_i} = LUT_{exp}[idx_{x_i},]$ .  
7: end for  
8: Accumulate normalization factor:  $\Sigma e^{x_i}$   
9:  $idx_{e^{x_i}} = MSB(e^{x_i}); idx_{\Sigma} = MSB(\Sigma e^{x_i})$   
10: for  $i = 1$  to  $size(x)$  do  
11:    $\sigma(x_i) = LUT_{\sigma}[idx_{e^{x_i}}, idx_{\Sigma}]$ .  
12: end for  
13: De-quantize  $\sigma(x) = \frac{\sigma(x)}{scale}$ 
```

500 – LUT_{α} is 1D LUT with precomputed PDF normalizing constant values in chosen
501 precision. There are already several pre-defined versions of LUT_{α} with different
502 precision (e.g., int16, int8) in our code, see Section 5 for more details.

503 • For Algorithm 2

504 – LUT_{exp} is 1D LUT where exponentiation values are stored for e^{x_i} computation
505 – LUT_{σ} is 2D LUT for softmax computation with the precomputed values in chosen
506 precision. There are already several pre-defined versions of LUT_{σ} with different
507 precision (e.g., int16, int8) in our code, see Section 5 for more details.

508 The output of Algorithm 1 and 2 is the tensor with computed softmax values $\sigma(x)$. The shape of the
509 tensor is exactly same, as the shape of the input tensor x .

510 A.3 PTQ-D dynamic quantization

511 To obtain dynamically quantized models (referred as PTQ-D) we have followed PyTorch method-
512 ology described at <https://pytorch.org/docs/stable/quantization.html#> and https://pytorch.org/tutorials/recipes/recipes/dynamic_quantization.html.
513

514 We have used the default PyTorch quantization scheme, thus the linear layers of the
515 model have been quantized with the following properties: `dtype=torch.qint8` and
516 `qscheme=torch.per_tensor_affine`. As a result the size of quantized models was reduced, but
517 there is some accuracy drop for some of the models as shown in Table 4. This accuracy drop should
518 be taken into account when overall accuracy of the model (including approximation of softmax layer)
519 is analyzed.

Table 4: Properties of dynamically quantized PTQ-D models

MODEL	FP32, MB	PTQ-D, MB	SIZE REDUCE RATIO, %	ACCURACY DROP, %
DETR (R50)	166.69	128.49	77	0.0
DETR+DC5 (R50)	166.69	128.49	77	0.0
DETR (R101)	242.96	204.77	84	0.0
DETR+DC5 (R101)	242.96	204.77	84	0.0
TRANSFORMER (WMT14)	390.99	210.47	54	0.12
TRANSFORMER (WMT17)	390.99	210.47	54	0.14
BERT (SST-2)	437.98	181.43	41	0.58
BERT (MRPC)	437.98	181.43	41	0.66

Table 5: LUTs size used for DETR experiments

PRECISION	BITS PER ENTRY	CASE 1		CASE 2		CASE 3	
		SIZE OF LUTs	TOTAL SIZE, BYTES	SIZE OF LUTs	TOTAL SIZE, BYTES	SIZE OF LUTs	TOTAL SIZE, BYTES
INT16	15	1×13	538	1×13	666	1×13	1,050
		1×256		1×320		1×512	
UINT8	8	1×8	264	1×8	328	1×8	520
		1×256		1×320		1×512	

A.4 DETR quantization experiment

Table 5 shows the LUTs size for several pre-selected cases in int16 and uint8 precision. The first row shows the dimensions for $LUT_{1/e}$ (e.g., 1×13) and second for LUT_{α} (e.g., 1×256). Total required size in Bytes for both tables is also shown. Note, that the total size of LUTs in Table 5 is just estimation for comparison purpose, and can be slightly different due to real hardware specification.

In Table 6 and Table 7 below there are accumulated values of Average Precision (AP) and Average Recall (AR) from the experiments with DETR models, as described in Section 5.1. The behavior of Average Recall values is similar to Average Precision values. The bold values in the table shows highest values per model per method after applying dynamic quantization and softmax approximation. From Table 6 and Table 7 an averaged accuracy drop w.r.t. to original FP32 models was calculated and shown in Figure 2.

A.5 NLP quantization experiment

Table 8 shows the LUTs size for several pre-selected cases for NLP experiments in different precision:

- For 2D LUT method the first row shows the dimensions for LUT_e (e.g., 1×101) and second for LUT_{σ} (e.g., 11×60).
- For REXP method the first row shows the dimensions for $LUT_{1/e}$ (e.g., 1×13) and second for LUT_{α} (e.g., 1×16).

Total required size in Bytes for both tables is also shown. Note, that the total size of LUTs in Table 8 is just estimation for comparison purpose, and can be slightly different due to real hardware specification.

In Table 2 there are accumulated values from the experiments with NLP models, as described in Section 5.2. The bold values in the table shows highest values per model per method after applying dynamic quantization and softmax approximation. From Table 2 an accuracy drop w.r.t. to original (FP32) and quantized (PTQ-D) models was calculated and shown in Figure 3.

Table 6: Experimental validation over DETR models (Average Precision)

MODEL	METRIC	FP32	PTQ-D	INT16			UINT8		
				CASE1	CASE2	CASE3	CASE1	CASE2	CASE3
DETR (R50)	AP	0.42	0.42	0.413	0.417	0.418	0.411	0.417	0.417
	AP_50	0.624	0.624	0.618	0.621	0.622	0.616	0.62	0.621
	AP_75	0.442	0.442	0.434	0.439	0.44	0.434	0.439	0.439
	AP_S	0.205	0.205	0.19	0.198	0.202	0.19	0.197	0.199
	AP_M	0.458	0.458	0.453	0.456	0.457	0.451	0.455	0.456
	AP_L	0.611	0.611	0.609	0.609	0.609	0.608	0.608	0.608
DETR +DC5 (R50)	AP	0.433	0.433	0.382	0.394	0.411	0.376	0.392	0.401
	AP_50	0.631	0.631	0.603	0.613	0.623	0.599	0.61	0.615
	AP_75	0.459	0.459	0.396	0.41	0.431	0.386	0.411	0.419
	AP_S	0.225	0.225	0.164	0.176	0.199	0.159	0.174	0.181
	AP_M	0.473	0.473	0.417	0.432	0.449	0.411	0.431	0.44
	AP_L	0.611	0.611	0.578	0.589	0.600	0.575	0.592	0.601
DETR (R101)	AP	0.435	0.435	0.426	0.431	0.433	0.426	0.431	0.432
	AP_50	0.638	0.638	0.633	0.635	0.637	0.632	0.636	0.636
	AP_75	0.463	0.463	0.452	0.458	0.459	0.452	0.459	0.459
	AP_S	0.218	0.218	0.208	0.215	0.218	0.207	0.218	0.218
	AP_M	0.479	0.479	0.47	0.475	0.476	0.471	0.477	0.476
	AP_L	0.618	0.618	0.614	0.617	0.617	0.615	0.619	0.617
DETR +DC5 (R101)	AP	0.449	0.449	0.363	0.388	0.426	0.358	0.387	0.42
	AP_50	0.647	0.647	0.589	0.612	0.636	0.586	0.61	0.632
	AP_75	0.477	0.477	0.371	0.401	0.451	0.365	0.400	0.444
	AP_S	0.237	0.237	0.136	0.16	0.206	0.128	0.155	0.196
	AP_M	0.495	0.495	0.397	0.426	0.468	0.391	0.426	0.464
	AP_L	0.623	0.623	0.578	0.591	0.611	0.575	0.594	0.608

Table 7: Experimental validation over DETR models (Average Recall)

MODEL	METRIC	FP32	PTQ-D	INT16			UINT8		
				CASE1	CASE2	CASE3	CASE1	CASE2	CASE3
DETR (R50)	AR	0.333	0.333	0.33	0.332	0.332	0.329	0.331	0.331
	AR_50	0.533	0.533	0.525	0.53	0.531	0.524	0.529	0.53
	AR_75	0.574	0.574	0.568	0.571	0.573	0.565	0.571	0.572
	AR_S	0.312	0.312	0.296	0.31	0.308	0.301	0.305	0.307
	AR_M	0.628	0.628	0.624	0.626	0.627	0.621	0.626	0.625
	AR_L	0.805	0.805	0.806	0.803	0.803	0.804	0.803	0.805
DETR +DC5 (R50)	AR	0.342	0.342	0.312	0.319	0.328	0.31	0.318	0.324
	AR_50	0.551	0.551	0.498	0.511	0.529	0.492	0.509	0.519
	AR_75	0.594	0.594	0.539	0.553	0.571	0.533	0.55	0.562
	AR_S	0.344	0.344	0.266	0.283	0.307	0.261	0.279	0.288
	AR_M	0.646	0.646	0.591	0.605	0.624	0.586	0.605	0.617
	AR_L	0.814	0.814	0.785	0.791	0.802	0.778	0.79	0.803
DETR (R101)	AR	0.344	0.344	0.338	0.342	0.342	0.339	0.342	0.342
	AR_50	0.549	0.549	0.541	0.544	0.546	0.539	0.545	0.546
	AR_75	0.59	0.59	0.582	0.586	0.589	0.581	0.586	0.587
	AR_S	0.337	0.337	0.324	0.332	0.336	0.322	0.333	0.335
	AR_M	0.644	0.644	0.638	0.641	0.642	0.637	0.641	0.641
	AR_L	0.815	0.815	0.814	0.812	0.812	0.809	0.814	0.814
DETR +DC5 (R101)	AR	0.35	0.35	0.303	0.317	0.337	0.301	0.317	0.334
	AR_50	0.561	0.561	0.477	0.501	0.538	0.472	0.501	0.533
	AR_75	0.604	0.604	0.517	0.541	0.58	0.511	0.541	0.575
	AR_S	0.348	0.348	0.24	0.266	0.315	0.23	0.262	0.305
	AR_M	0.662	0.662	0.57	0.596	0.637	0.561	0.597	0.636
	AR_L	0.81	0.81	0.771	0.784	0.80	0.769	0.784	0.801

Table 8: LUTs size used for NLP experiments

PRECISION	BITS PER ENTRY	2D LUT		REXP	
		SIZE OF LUTs	TOTAL SIZE, BYTES	SIZE OF LUTs	TOTAL SIZE, BYTES
INT16	15	1×101 11×60	1,522	1×13 1×16	58
UINT8	8	1×101 11×60	761	1×8 1×16	24
UINT4	4	1×48 11×29	367	1×5 1×16	21
UINT2	2	1×12 11×8	100	1×3 1×7	10