

Peer Review

Review of: "Partial Convolution Meets Visual Attention"

Evgenii Arsentev¹

1. Independent researcher

1. What the work claims

The paper starts from a practical observation about FasterNet: partial convolution (PConv) runs a 3×3 convolution on only a slice of the channels and passes the rest through as identity, which is fast but leaves those untouched channels doing nothing useful. The authors propose to spend that idle branch on attention. Formally (Eq. 1, §3.1) the output is the concatenation of $\text{Conv}(X_{Cp})$ and $\text{Atten}(X_{\{C-Cp\}})$, with the split ratio fixed at $1/4$ in every variant (Table 6). Three blocks follow from this template: PAT_ch, which pairs the 3×3 convolution with a "Gaussian-SE" channel attention using both channel mean and variance; PAT_sp, a 1×1 convolution plus Hard-Sigmoid spatial gate placed after the MLP so that its 1×1 can be fused with the MLP's second 1×1 at inference; and PAT_sf, a partial self-attention with relative position encoding used only in the last stage. Stacked into a four-stage network with a deeper final stage, these give the PATNet family (T0 to L).

The headline claim is that PATNet beats FasterNet at every size on ImageNet-1K and on COCO detection and instance segmentation with Mask R-CNN, while keeping comparable or better speed. The abstract singles out PATNet-T2: 1.3% higher accuracy than FasterNet-T2, 25% higher GPU throughput, and 24% lower CPU latency.

2. Whether the conclusion holds on the data

Much here is genuinely good. The authors measure speed on two different GPU vendors (V100 and MI250) plus a single CPU core, rather than reporting FLOPs alone; for an efficiency paper, that is the right instinct. They also do something I wish were standard: Table 11 retrains FasterNet under their own recipe, so the reader can separate architecture from training tricks. And the ablations in Tables 3, 4, 5, and 10 are targeted at the actual design decisions, not padding. The core idea is simple, and that is a strength.

On the main comparison, the accuracy gains hold up well. Table 11 is the fairest evidence, and there PATNet-T2 reaches 80.2 against 79.2 for FasterNet-T2* (one point), PATNet-T1 78.1 against 76.5 (1.6 points), and PATNet-T0 73.9 against 71.0. At the large end, the margin shrinks: PATNet-M 83.1 against FasterNet-M* 83.0, and PATNet-L 83.9 against 83.6. I would say so in the text. A 0.1-point gain at the M size, with 61.3M parameters against 53.5M (Table 1, column Params), is a tie on accuracy and a win only on FLOPs and speed.

I checked the abstract's T2 numbers against Table 1. V100 throughput: $(3074 - 2455) / 2455 = 619 / 2455 = 25.2\%$, correct. MI250: $(4761 - 4189) / 4189 = 572 / 4189 = 13.7\%$, correct and matching §4.1. CPU latency is where it goes wrong. FasterNet-T2 is 43.7 ms and PATNet-T2 is 35.2 ms, so the reduction is $8.5 / 43.7 = 19.5\%$. The stated 24.1% is $8.5 / 35.2$, i.e., FasterNet is 24.1% slower than PATNet, which is not the same statement as PATNet being 24% lower. The accuracy gain, $80.2 - 78.9$, is 1.3 percentage points, not 1.3%.

The speed claim also needs narrowing. §4.1 says PATNet "consistently surpasses" FasterNet across all variants, and the conclusion of that paragraph says attention is added "without impacting throughput". Table 1 does not quite support either. For T0, V100 throughput is 7777 against FasterNet-T0's 8546, and CPU latency is 12.2 ms against 10.5 ms. For T1, V100 throughput is 4403 against 4648. So at the two smallest sizes, PATNet trades some V100 speed and (for T0) CPU latency for accuracy. That is a perfectly respectable trade. It just is not uniform dominance.

Table 4 tells the same story from inside the model. Going from the bare 2-2-6-4 row (6405 FPS, 25.7 ms, 76.0%) to the full model (4761 FPS, 35.2 ms, 80.2%) costs $(6405 - 4761) / 6405 = 25.7\%$ of throughput and raises latency by $9.5 / 25.7 = 37\%$. Each block earns its keep in accuracy (+1.4, +1.5, +1.3 points as PAT_ch, PAT_sp, and PAT_sf are added), but the cost is real. The PATNet versus FasterNet speed advantage at T2 comes largely from the narrower widths described in §3.3, not from attention being free. I would make that attribution explicit.

Three ablation readings deserve softer wording.

- Table 3 shows partial and full attention within 0.3 points of each other (80.2, 80.1, 79.9, 80.2). For full self-attention, the accuracy is identical at 80.2, with 14.5M against 12.6M parameters. That supports "partial attention matches full attention at lower cost." It does not support "superiority" in accuracy.
- The stage-depth comparison in Table 4 (last two rows) shows 78.8 to 80.2, but the two models differ by $12.6 - 9.7 = 2.9\text{M}$ parameters. Some of the 1.4-point gain may be capacity rather than placement. A parameter-matched 2-2-8-2 variant would settle it.

- In Table 10, plain SE-Net in the partial branch runs at 4910 FPS and 32.3 ms against 4761 FPS and 35.2 ms for Gaussian-SE, with 79.8 against 80.2. So the Gaussian statistics buy 0.4 points at a measurable speed cost; that is worth stating directly.

Finally, every result is a single training run. No seeds, no standard deviations. Several of the ablation differences (Table 3, Table 5 rows Conv 79.9 versus DWConv 79.6, Table 10) are 0.3 to 0.4 points, which is within the range that seed-to-seed variation on ImageNet can produce. I could not judge from the text whether they are stable.

COCO results (Table 2) look favorable: in the first AP column, PATNet-S has 42.7 against FasterNet-S 39.9, PATNet-M 44.3 against 43.0, PATNet-L 44.7 against 44.0. But as rendered on Qeios, the six AP columns of Table 2 are headed AP^b three times and AP^m three times, with the subscripts lost, so I am inferring that they are AP^b , AP^b_{50} , AP^b_{75} , AP^m , AP^m_{50} , AP^m_{75} . The text says PATNet keeps "similar latency," yet the table reports throughput.

3. Reproducibility

The training side is documented better than most. Table 6 gives widths, depths, partial ratio, and activation per variant, and Table 7 gives the full ImageNet recipe (batch size, learning rate, weight decay, stochastic depth, RandAugment, Mixup and CutMix, multi-scale training resolutions). Table 8 does the same for COCO. With these tables, I believe a competent reader could rebuild the training loop.

What I could not find:

- A code or weights link. I searched the Qeios text and found none. For a backbone paper whose main claims are speed numbers, released code is close to essential.
- The exact form of Gaussian-SE. §3.2 says it uses channel mean and variance, but how the two statistics are combined (concatenated, summed, passed through separate FCs) is shown only in Figure 3(a), and as rendered on Qeios, the figure is not legible to me. The number of heads and the RPE variant used in PAT_sf are also not stated in the text.
- Benchmark protocol. The text gives batch 256 for GPUs and one core of an EPYC 73F3 for CPU, which is good. Missing: inference precision (FP32 or FP16), framework and library versions (PyTorch, CUDA, ROCm), number of warm-up and timed iterations, and whether the PAT_sp fusion and BN folding were applied to all models or only to PATNet.

- Training compute: number and type of GPUs for ImageNet and wall-clock time. Table 8 implies 8 GPUs for COCO (16 images at 2 per GPU), but ImageNet is not specified.
- Seeds and repeat counts, as noted above.

One item in Table 7 looks inconsistent. "# of forward pass" is 188k for every variant. With roughly 1.3M training images (§4.1) and 300 epochs, batch 4096 gives $1.3M / 4096 \approx 317$ steps per epoch, so about 95k steps; batch 2048 gives about 635 steps per epoch and about 190k. So 188k fits the M and L variants (batch 2048) and is about twice too high for T0 through S (batch 4096). It may have been carried over from another table, and I would recheck it.

4. What specifically to fix

1. Abstract and §4.1, "Results": CPU latency of PATNet-T2 is 19.5% lower than FasterNet-T2 (8.5 / 43.7 from Table 1), not 24.1%. And write "1.3 percentage points" rather than "1.3%".
2. §4.1, "consistently surpasses" and "without impacting throughput": qualify both. Table 1 shows PATNet-T0 slower than FasterNet-T0 on V100 (7777 vs 8546 FPS) and on CPU (12.2 vs 10.5 ms), and PATNet-T1 slower on V100 (4403 vs 4648). Table 4 shows attention itself costs about a quarter of throughput at T2.
3. Table 3 and the paragraph above it. Replace "superiority" with something like "comparable accuracy at fewer parameters", since accuracies are 80.2, 80.1, 79.9, 80.2.
4. Table 4, last two rows: I would add a 2-2-8-2 configuration widened to roughly 12.6M parameters, so that the stage-depth effect is separated from the 2.9M parameter difference.
5. Tables 3, 4, 5, and 10 report a single "Throughput (FPS)" and "Latency (ms)". The values (4761, 35.2) match the MI250 and CPU columns of Table 1; please say so in the headers.
6. For the main T2 results and at least the Table 3 and Table 5 ablations, report mean and standard deviation over three seeds. Several differences there are 0.3 points.
7. Table 2: restore the subscripts in the AP column headers (three columns read simply AP^b and three AP^m). Also, replace "similar latency" in the results paragraph with "similar throughput", and fix the citation for ResNet101, which points to [20] (MobileViT) instead of the ResNet paper.
8. §3.2, PAT_ch: write out the Gaussian-SE computation as an equation (how mean and standard deviation are fused, the reduction ratio of the FC layers). The current sentence "Unlike SENet, it only considers the mean information" reads as if "it" were your module; I think it refers to SENet.

9. §3.3 says the last stage is twice as deep as the first two. That matches T1, T2, and S (2-2-x-4), but Table 6 gives T0 as 1-2-6-4 and M and L as 2-3-16-4 and 2-3-20-4. A short note would remove the apparent contradiction.
10. Small table inconsistencies: PATNet-L has 104.4M parameters in Table 6 but 104.3M in Tables 1, 9, and 11; in Table 11, the second FasterNet-S row is missing its asterisk; Table 7 "# of forward pass" (see section 3 above).
11. Release code and pretrained weights, and add a benchmarking paragraph: precision, framework versions, warm-up and timed iterations, and whether fusion and BN folding were applied equally to all baselines.
12. Text: "Depthwise onvolution" in the abstract, "PA_sp" in §1, "we employee" in §3.2, "FastNet" in A.2 and A.4. The Table 1 caption mentions bold results, which do not survive in the Qeios rendering.
13. Conclusion: "optimal balance" is stronger than the evidence. "A better accuracy-speed trade-off than FasterNet at the evaluated sizes" is what the tables show.

5. Recommendation

Minor revisions. The idea is clean and useful: put the idle channels of partial convolution to work with cheap attention. The experiments are real, cover two GPU vendors and a CPU, and include a same-recipe FasterNet baseline that strengthens the accuracy claim considerably. What needs changing is mostly calibration. One latency percentage is computed against the wrong denominator, the speed claims are a little broader than Table 1 supports at the small sizes, a few ablations are read more strongly than single-run differences of 0.3 points allow, and the Gaussian-SE module and benchmarking protocol need to be written down precisely. None of this requires new architecture work; a parameter-matched stage ablation and a few repeated seeds would be the largest additions.

Declarations

Potential competing interests: No potential competing interests to declare.