VRandme commented 4 days ago

I regret not being able to participate in the discussion laid out in #121 as it happened.
As it stands, the original author of TResnet **@mrT23** was present.

Since this post is after the fact, with a TResnet(from the original author at that!) was pulled into here,
I just would like to lay my opinions on it and ask some questions that hopefully **@mrT23** would be able to answer.

As summarized in the paper, I view that the fundamental contributions of TResnet boils down to the following

1. Stem : SpaceToDepth
2. Blocks selection
3. Inplace-ABN
4. Dedicated SE
5. Antialiasing.

I will address these in increasing complexity.
5. Antialiasing (https://github.com/adobe/antialiased-cnns), is a well known and tested method of increasing accuracy and consistency. It was also used in assembled cnns

4. Dedicated SE : **@mrT23** made great efforts to streamline and optimize Squeeze and Excite. I would like to find out how it fares against Efficient Channel Attention. In theory, ECA or (my own cECA) would be able to be optimized similarly to show better parameter and computational efficiency and accuracy.
   As it is, Tresnet is not amenable to drop in replacements of attention, but it could be rendered as such easily.

5. Inplace-ABN.
   I wonder if such a method could be applied to EvoNorm(https://arxiv.org/pdf/2004.02967.pdf).
   considering EvoNorm is itself an attempt to gather activation layers and Normalization layers and search for them in an end to end manner, its possible that it is not necessary for EvoNorm.
   I havent seen or conducted for myself enough testing on EvoNorm to tell for myself,

6. block selection.
   The recent RegNet paper(https://arxiv.org/abs/2003.13678) showed that even for bottleneck layers a bottleneck of 1 (no bottleneck channel expansion) could be effective and uses such layers extensively to construct what is ostensibly a more simple resnet that is more effective.
   However, it has only been tested (as per the original paper) in limited capacity without all the bells and whistles of modern CNNs so it remains to be seen what kind of performance it would show WITH all the bells and whistles.

Furthermore, while the RegNet paper compares the bottleneck block (1x1, 3x3 with or without expansion followed by another 1x1 with residual connections.) with the vanilla block(one 3x3 with

or without residual connection), it is not a proper comparison.
The real Vanilla resnet block would have to have TWO 3x3 layers with a residual layer. Like TResnet.
It might be valuable to see what TResnets could do with all stages with basic blocks or bottlenecks without channel expansions. Such a comparison would require concomitant hyperparameter tuning to adjust channel widths and layer counts but maybe RegNet scaling might show valuable pointers.

1. SpaceToDepth Stem

The Space to Depth stem is valuable tool to increase GPU throughput. The fact that it maintains or even increases accuracy is cherry on top.

My concern is that SpaceToDepth is hard to visual conceptually. I fear that this might lead to it being difficult to visualize functionally. For example, visualizing intermediate layer activations is an important tool to understand why a model functions the way they do. I'm not sure how the initial non visually intuitive stem would affect following layers from an interpretability standpoint.

In a similar vein, I'm concerned that SpaceToDepth might hinder TResnet's ability to be integrated to image segmentation or detection pipelines for the above reason.

One of the reasons that EfficientNets have took so long to be utilized in many image detection frameworks to displace ResNets was that a meaningful feature extractor was difficult to code.

There have been some attempts (like the version that exists in this very repo) but the difficulty of such endeavor, alongside difficulties in GPU throughput and fragile training, made it hard to vanquish ResNets.

I would love if **@mrT23** could provide insights to these issues.

To be frank, my interests faded in (T)ResNets after seeing the RegNet paper. I hope that the eventual code and model releases will rekindle it.

In the end, TResnets are insightful, powerful, effective and efficient models in their own right. My points come more from curiosity than criticism and I hope **@mrT23** understand my appreciation for their work and efforts (especially wrt incorporating their contributions to this beneficial code base).

mrT23 commented 4 days ago

Hi **@VRandme**
you raised a lot of issues, so i have lots of answers :-)

part1:
Antialiasing - i learned about the idea from Assembled-cnn paper (although they did not invent it themselves). tried several hyper-parameters and basically went with their design, while introducing code optimizations. its contribution to imagenet is medium - some improvement, but it has non-negligible GPU cost. However, it does wonders for transfer learning to fine-grain datasets, and that's the main reason i used it.

Dedicated SE - again, didn't invent the wheel here or something here. but i do think i have some important optimizations to get better speed-accuracy tradeoff. In general, SE layers are in my opinion the biggest advancement in deep learning architecture in recent years. they give an excellent speed-accuracy tradeoff. i tried to replace SE by plain ECA and in Tresnet, and it lowered the scores a bit. might revisit it in the future

Inplace-ABN - inplace ABN is a marvelous thing. if PyTorch had some sense, they would set it as their default option. with in place ABN you can use batch size twice as large. twice!
for higher resolutions, the increased batch size is a major major strength.
i didn't read EvoNorm yet, thanks for the reference.

block selection - i read RegNet but hadn't tried their models yet. i recommend to be very very cautious about it. it seems a lot less practical than the explosive headers. EfficientNet models also claimed to reinvent the wheel, and they turned out to be terrible models.
i have more to say about it, but my post is getting longer and longer. always remember to compare inference and train times, not just inference times.
but if RegNet is indeed a good design (i am more skeptical than you), than most of TResNet enhancements will still be relevant to it.

SpaceToDepth -
SpaceToDepth design is in my opinion the future.
it has two major advantages:
(1) the stem design of ResNet model is terrible. it is quite expensive and ineffective, since you can loose "information" during the quick downscaling it does. once an information is lost in the stem cell, it can never be regained.
my intuition for using SpaceToDepth was that no information can be lost in it.
it lets the blocks (with the residual connection, that also prevents information loss) to do the actual processing, and keep us protected from information loss

(2) you get an image, do SpaceToDepth, and then just do a bunch of repeated blocks. repeated (simple) blocks is an efficient and ultra-fast GPU design.
for a future design, i am thinking about an even more extreme version of SpaceToDepth, with less or even no further resolution reductions. just SpaceToDepth and one repeated block.

regarding performances:
We were able to get SOTA results with SpaceToDepth on fine-grain datasets, where the resolution is very important. i didn't mention it in the article, but we are also winning kaggle competitions with TResNet (and SpaceToDepth), where resolution is also ultra-important.

hence i (strongly) don't see why SpaceToDepth would damage scores of image segmentation or detection pipelines. we are currently testing TResNet on detection.

regarding interpretability, you might be right, i can see why the activation maps of SpaceToDepth can be harder to interprete. is interpretability really that important ? i have never used it

VRandme commented 3 days ago

@mrT23 First of all, thank you for the reply. I hope you realize that I know that developing, training, and deploying ImageNet scale models is not easy task and I respect and thank you for that.
Put in that context, my criticisms border on nitpicking but i appreciate that you put the effort to respond.

Antialiasing : I was unaware of its benefits in Transfer learning and Fine graining. I was more interested in Consistency/robustness. The fact that such benefits exist fit very nicely with your model (good for Transfer learning or fine graining on GPUs).

Dedicated SE : thank you for sharing your anecdotal evidence on ECA. Although ECA is very efficient, It might be less versatile than SE since they have different mechanisms. ECA might have to increase channel widths and parameters to effectively compare with SE in many other yet to be tested scenarios, and then its back down the hyperparameter rabbit hole. There's nothing wrong with sticking with well accepted best practices, which you not only adopted but improved upon.

inplace ABN is a marvelous thing. very very true.
block selection - you might very well be right, and the delayed release of actual models of regnets is making me nervous. Reminds me of the delay between the EfficientDet paper and model release.

SpaceToDepth : .... "SpaceToDepth design is the future." I actually agree!

to contextualize my opinions :
I **don't** see why SpaceToDepth would damage scores of image segmentation or detection pipelines, **either**. What I **DO** think is that the stem might make it harder to develop and maintain code for TResNet on detection. However, since you are currently working on it, you are the best person to answer that question and it seems you are not anticipating any problems. which is good.

In the end, even if the activation maps for SpaceToDepth is hard to interpret, if it proves to be effective (which I think it will), then visualization tools and approaches will grow to work with such a stem.
I only hope it won't be too difficult to do.
Again thank you for your code and your discussion.

mrT23 commented 2 days ago

Hi Chris
i enjoyed reading your insights and suggestions, and in current corona days i have ample time to answer them thoroughly :-)
As you can see, i can be critical toward other deep learning models, but it also means i have to be able to accept criticism and suggestions to my work. That's the only way to get better.

One last insight regarding SE and ECA -
In general, the bigger a deep learning model gets, the better it performs. The magic of SE layers

is that they are able to increase significantly the number of parameters of a model, without increasing significantly the runtime.
that's why i was suspicious towards ECA - it adds much fewer parameters than SE, so i would expect it won't give the same score boost. I can say ECA outperformed my initial expectations. However, there is an inner "lie" in ECA - the throughput improvement compared to SE is very small. why - the major runtime consumption in a SE layer is the global average pooling, not the fully connected. ECA also has global average pooling, so its runtime cost is very similar to SE.

anyway, i gave a talk last week about "The dark magic behind deep learning" that contains further insights regarding TResNet models, as well as other issues.
you are welcome to take a look
https://drive.google.com/file/d/1xPfa3XpqTZTeCrpuJtosqEGNDbmD5M9Q/view

all the best
Tal

VRandme commented 2 days ago

@mrT23
Thank you very much for your insights!
I have a few questions tho.

1. Is the "soft-triplet loss" you recommend the same as the "SoftTriple Loss" laid out in this paper?
   https://arxiv.org/pdf/1909.05235.pdf

2. May I share your presentation with other researchers?(I was thinking on sharing it on the PyTorch KR community which is a pytorch facebook group in korea)

3. You mention AutoAugment. Many agree that it is a powerful tool but learning the policies are the difficult part. For well researched datasets such as ImageNet there are predetermined (by other researchers) policies. However for datasets where such appropriate settings are not known, or in cases the models used deviate so much from the original networks used on ImageNet to learn the policies this becomes an issue. Recently I've found many papers using a later method called RandAugment to vastly simplify or outright automate the policy learning process.
   Although AutoAugment comes out on top in the best case optimized scenario, RandAugment shows competitive or better performance in many E2E tasks especially in the forementioned scenarios.
   Do you have any opinion on this?

4. Also, this discussion is going deep into TResNets rather than how it relates to this repo in particular. I would be happy to open an issue on https://github.com/mrT23/TResNet or any other forum that you seem fit.

I didn't expect you to respond at all at first but now I'm getting awesome insights.

mrT23 commented 2 days ago

- Triplet loss was designed and implemented by a co-author of the article - Hussam lawn
  he brought it from the world of person Reid:
  https://arxiv.org/pdf/1910.07038.pdf
  At first i was skeptical if it is applicable also to plain classification, but in the end, without it we wouldn't reach SOTA score on fine-grain datasets (Stanford-cars and Oxford Flowers)

- you may share the presentation, it is from a public talk i gave

- regarding 'AutoAugment' Vs 'RandAugment' - the "search" part in 'AutoAugment' is in my opinion not true (i am trying hard to be polite :-)).
  google just run thousands of tests on ImageNet, with different policies, and chose the best one. they wrapped it under an "auto-searching" header so it could be an academic article.
  for ImageNet, i personally train with AutoAugment, its a good ImageNet policy.
  for other datasets, lately i switched to costume GPU augmentations that i developed, which give more control and adaptability. the are similar in spirit to what FastAI suggest, but more diverse. i haven't worked with 'RandAugment'.

is it possible to transfer an issue from one repo to another ? if so, i would be happy to bring it to https://github.com/mrT23/TResNet .

Tal

rwightman commented 17 hours ago

GitHub did eventually implement issues transfer but I believe it's still limited to repo in same org. Cut and paste of comments could be done but tedious. Often best to close issue in one repo and then create new one in another and paste link to old in comments for reference.

Some interesting discussion here. A comment on ECA, for the overhead, it works quite well with ResNet like network architecture as you both know. I thought it might be a great fit for a lighter MBConv/InvertedResidual net like EfficientNet but it decreased the performance over a baseline with no SE. I tried all reasonable locations in the block too.

mrT23 commented 16 hours ago

thanks **@rwightman**
i will cut and paste to TResNet repo.

please close the issue