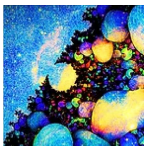
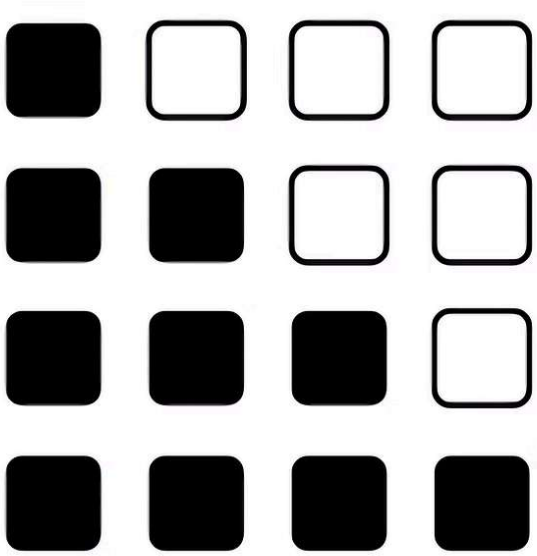


E Ethan Smith 5 days ago 10 min read

Softmax Attention is a Fluke

Updated: 4 days ago



Attention is the magic ingredient of modern neural networks. It is the core of what has launched performant language models into the spotlight starting with GPT, and since then, it has extended its hands across all modalities.

There are a number of desirable properties that make attention a first-class building block. Namely:

- It handles variable sequence lengths with ease
- It allows for a global receptive field without needing to scale parameters
  - as opposed to convolutional neural networks
- It enables dynamic routing of information between tokens
- It is parallelizable and incredibly GPU-friendly

The core principle of projecting tokens into another space, calculating their correlations, and using those as weights to define mixing strategies of the values is solid.

But here I want to very boldly claim that our specific choice of the well-known attention equation is a fluke. In particular, softmax appears to come out of nowhere.

Why Softmax?

Numerous papers have developed alternatives that did not end up making it to the mainstream for lack of competitive performance. So why dare to make the 1001th attempt to contend with this? Is it possible we got it right the first time?

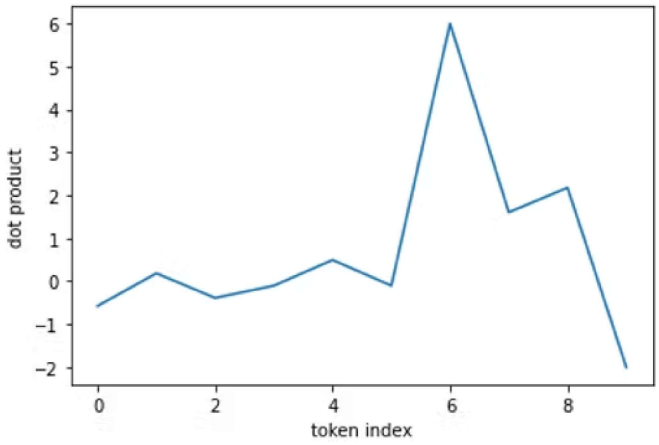
To be fair, most attention alternatives aim to circumvent the O(n^2) scaling of computation in sequence length by attempting to linearize the computation or use cheaper, more flexible activation functions. Attempts to skirt around the cost may naturally lend themselves to drops in capability. Perhaps the O(n^2) is inevitable.

However, let's stay with the core circuit of attention. The part I'd like to question is why specifically the exponential function, normalized across the sequence.

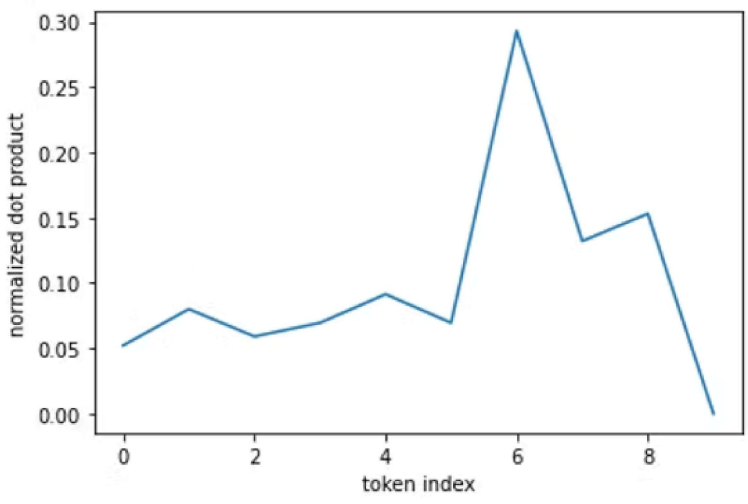
It's a natural choice. Softmax appears all around neural networks. It is a sensible means of turning real-valued vectors into a row of probabilities. It is also nicely differentiable and has the ability to non-linearly "silence" and filter out noise to reveal signal.

Attention, I would argue, makes good use of the latter traits. Here's an example of the suppression it can provide.

Let's imagine here that we have taken one vector and dot'ed it with 10 others, loosely measuring their correlation scaled by norms. This is a plot of their interactions.



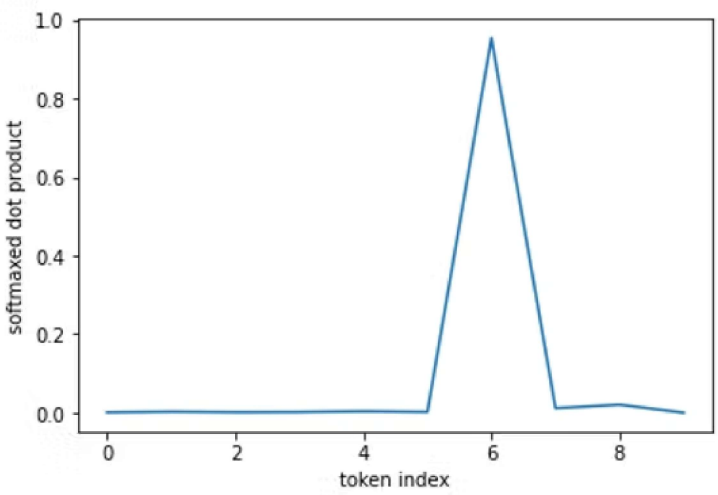
If we were to take a weighted sum of the 10 vectors with these coefficients, we could end up with a vector whose norm is quite large, where we generally like to have a consistent steady range of values for our neural network to run smoothly and not explode. So we can start by subtracting by the mean and sum normalizing it.



This is okay, but note that our final output will be a pretty mixed blend of all the tokens. It'll be 30% token 6, approximately 15% token 7 and 15% token 8 and so on.

But let's say we wanted to pull out the token with the largest interaction, or maybe a few, and not have noise from the other tokens mixed in. This is a common desire in database lookups (an established analogy for the role attention and dot products play in neural networks) where we might take be interested in the top-k vectors.

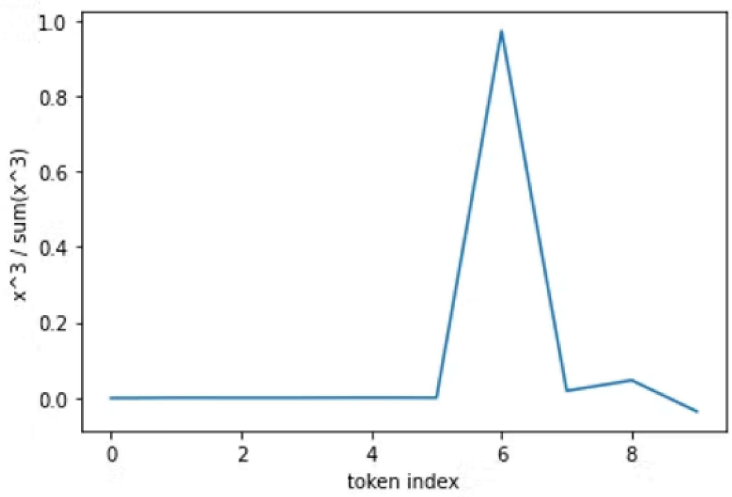
We'll want some means of setting the other positions to zero, but just doing **max(x)** or manually setting other positions to zero is not differentiable. Enter **softmax**.



Now if we take a weighted sum over the tokens, we almost recover the function of **max(x).** The output will be approximately ~99% token 6. This function has a useful winner-take-all nature to it via the nonlinear exponentiation followed by normalizing across the sequence. The growth of mass in one value means a loss in others, even being able to drive its comrades close to zero.

I've made a case up until now for why we really like softmax. It's noise suppression is useful. Though the other aspect of softmax is its root in forming probability distributions. **For that, there isn't exactly a reason we need to think of attention mixing weights as probabilities.**

There are plenty of other kernels that can yield a similar effect. For example, x^3 actually gives a pretty similar result for this case



This is the argument of this work, which reveals a number of polynomial kernels that can also perform well on toy tasks.

Additionally, do we need mixing coefficients to be exclusively positive? Why?

This was an argument I made here, where I argued that negative correlations may be just as much informative as positive ones, and it is perhaps the small correlations instead that should be suppressed. This later became a paper under a similar motivation.

Lastly, another work explored breaking free from typical normalization, allowing attention to take weighted sums outside of its convex cage.

So the questions I see are:

1. Why e^x? Especially if we don't need to use the values as probabilities?
2. Do we need sum-to-one normalization?
3. Why only positive values?

None of these, however, feel like bugs.

We may find further merit in searching the space of alternatives that deviate from these conventions, but after all, as I said, softmax was a fair choice for a first iteration due to its common usage and it was sufficient to bring us where we are today.

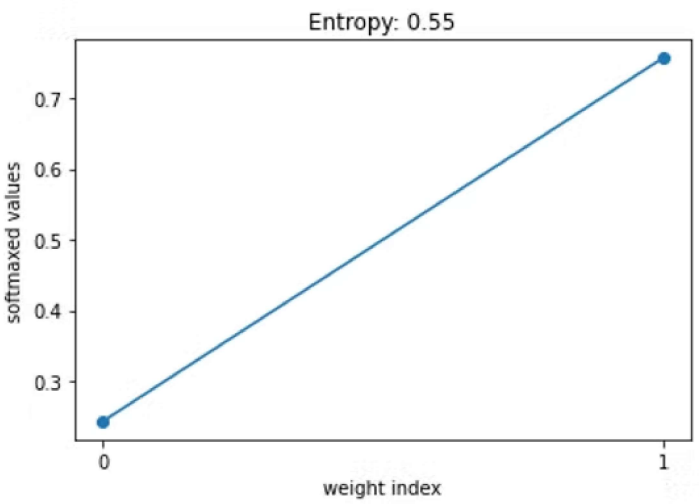
However, there is one aspect around how normalization and scaling is treated that I do feel *is a bit of a bug* with attention. Particularly in cases where our sequence length can vary, like with causal language models.

Normalization Across Variable Length Sequences

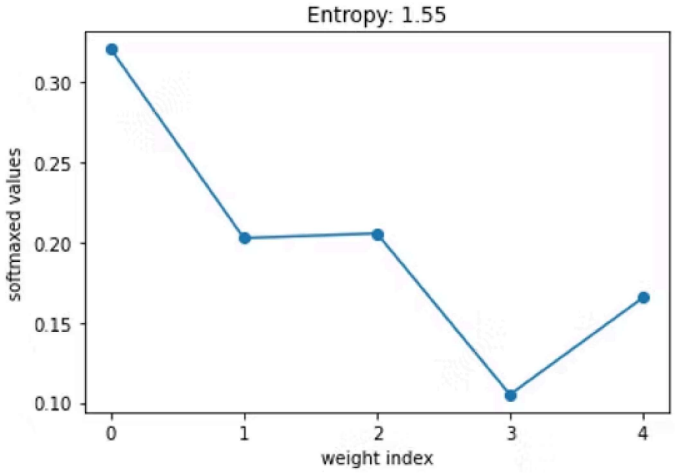
Loosely summed up in this tweet here.

The one part I find particularly fragile about attention is the dynamics around normalization, and how it can vary depending on the length of the sequence. Firstly, let's see a toy example of this in action. We'll sample logits randomly from the Gaussian distribution across different lengths and apply softmax to them. Observe what happens to the size of the largest value, entropy, and overall sharpness of the distribution.

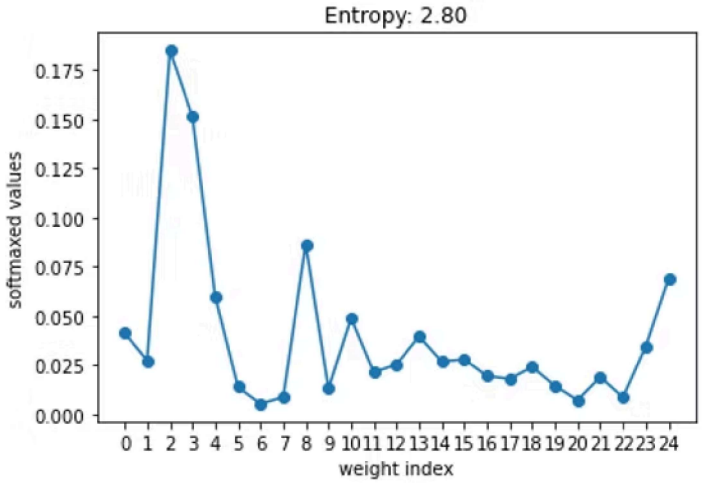
At 2 elements (Note: if we only had one element, 100% of the mass would go to it always, it would be a weighted sum over 1 element making attention pointless)



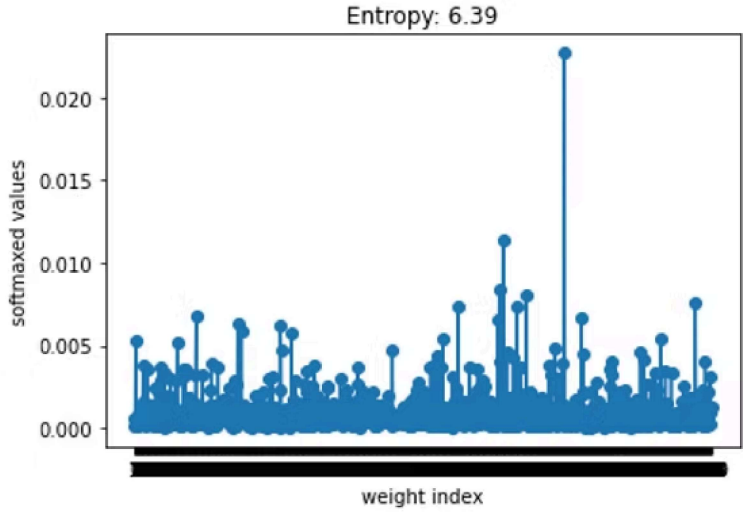
At 5 elements



At 25 elements



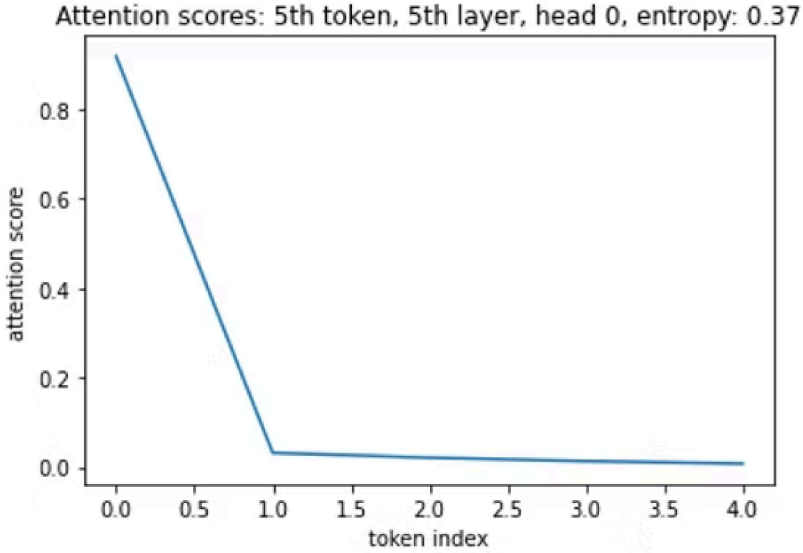
At 1000 elements



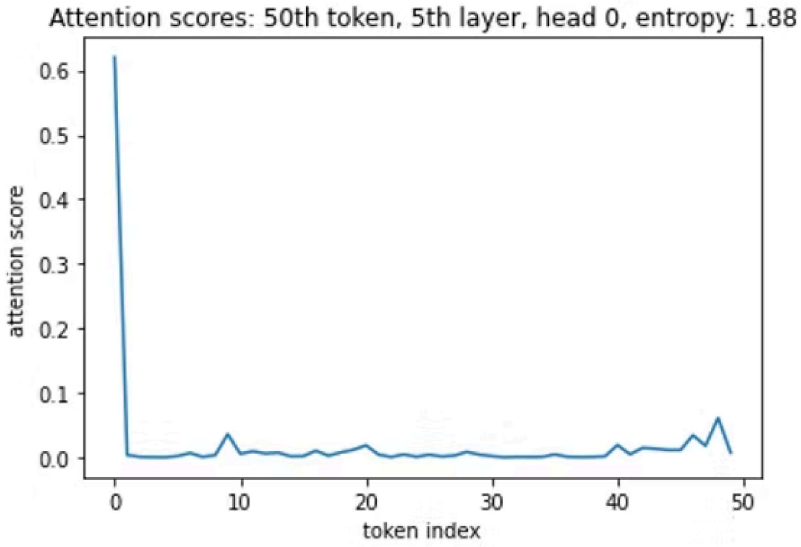
The distribution becomes progressively more flat, entropy increases, and the largest value becomes much smaller.

This is not just a reality of Gaussian variables, but is also observed with attention maps from our models for a generated piece of text, namely GPT2-large here.

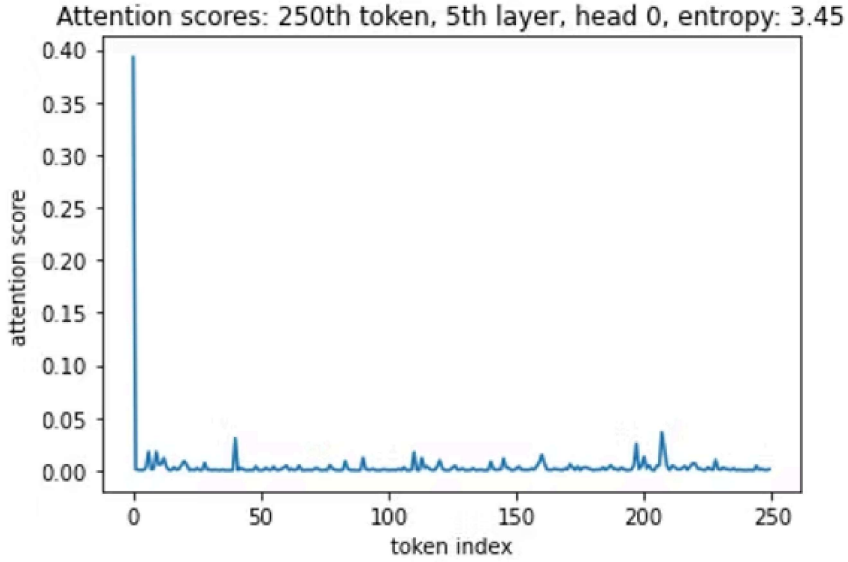
At length 5



At length 50



At length 250



Ignore the first token for now, that's something we'll talk about later. With growing sequence length so does the requirement to produce outlier-ly large values to overwhelm all the others and achieve sharpness. We end up with many small spikes rather than few large ones. Now entropy naturally increases with the number of choices, though again, is it necessary to apply the probability lens to attention?

It took reading a few works before what's happening really cemented in my mind, and now it's impossible to look away from.

Achieving Null Attention

The first was Softmax-plus-one, which argues attention should be able to do "nothing." I don't consider this a huge issue, and not the aspect I think of as the aforementioned bug, but a solution here would feel like a "nice to have."

This blog post criticizes the dependence the denominator has on the numerator. It could be the case that the a token would like to choose inaction, or have weighting coefficients that sum to less than 1 to reduce the magnitude of effect of a given attention layer.

Recall softmax looks like this

$$(\text{softmax}(x))_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

In the current formulation, this isn't particularly possible*. We can try to reduce our token-token interactions as much as possible, even making them exclusively negative, and attention would still have an effect. In fact, one could imagine an attempt at this kind of adaptation could backfire. If all correlations drop evenly, we'd approach a uniform distribution of attention as opposed to isolating a few candidates, which is exactly what brought us to using softmax in the first place!

The "fix" is to add a bias to the denominator, slightly decoupling the denominator's dependence. It would now be possible to output very small values in the numerator but have the denominator remain as a constant value.

$$(\text{softmax}_1(x))_i = \frac{\exp(x_i)}{1 + \sum_j \exp(x_j)}$$

If we imagine that

Then

$$\frac{\exp(x_i)}{1 + \sum_j \exp(x_j)}$$

Becomes approximately

$$\frac{0}{1 + 0}$$

Allowing for the capability to assign 0 weights everywhere, entirely nullifying attention. Though this can also appear more partially where exp(x) airs around small values below one allowing us to say, have our total weights sum to arbitrarily something like 0.2.

I have yet to see this implemented anywhere, and **in my opinion the missing piece to practicality is that the denominator bias should be learned rather than fixed to 1. 1 is an arbitrary choice and may ask attention weights to output around a certain range of values that may be unnatural for it. (though this too, as we'll see might not even be sufficient due to how the denominator's size also depends on the exact length of our sequence as well, we may prefer a learned bias that is a function of the sequence size)**

But why would attention want to do nothing? I paid for the layer; am I wasting my FLOPs?

Inaction is actually a good bias to have in neural networks in general. It is often a good heuristic to firstly assume the identity function, no effect, and then increase the effect from there. This is partially the principle of why we use residuals everywhere nowadays; learning the identity function directly by gradient descent is actually quite difficult. We can instead hardcode the identity function and ask that the networks learns to predict what should be added to it, the residual.

I left an * above on the remark that inaction is impossible for attention layers. Despite a lack of conventional means of choosing to do nothing, we'll often find that nature (neural networks) finds a way anyway in a form that is equal parts clever and aggravating. What you'll find is that tokens will often put a really large portion of their attending mass to a meaningless token like a padding, or a comma, or beginning of sequence marker. This paper aptly referred to this phenomena as *"Attention Sinks."* A token is forced to attend to its bretheren up until it fulfills the entirety of its attention budget or mass. Though it can choose to just dump the majority of it onto a token that confers no effect.

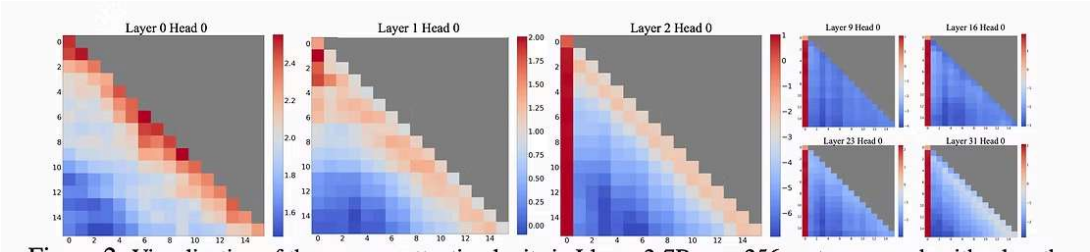
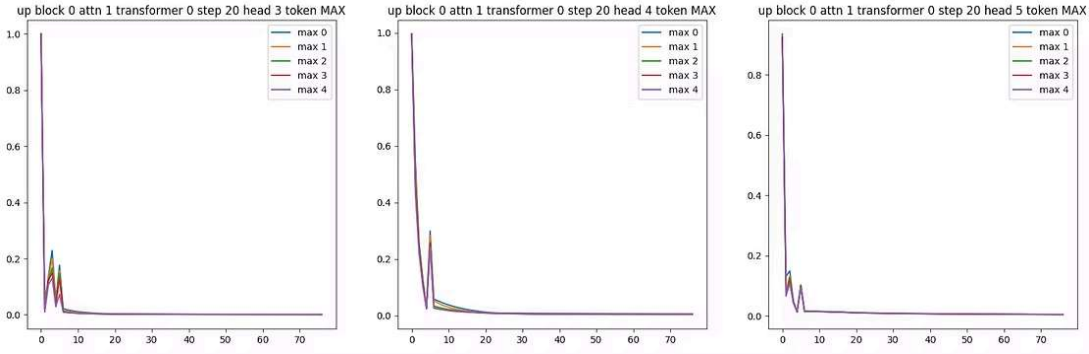


Figure 2: Visualization of the *average* attention logits in Llama-2-7B over 256 sentences, each with a length of 16. Observations include: (1) The attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern, with recent tokens receiving more attention. (2) Beyond the bottom two layers, the model heavily attends to the initial token across all layers and heads.

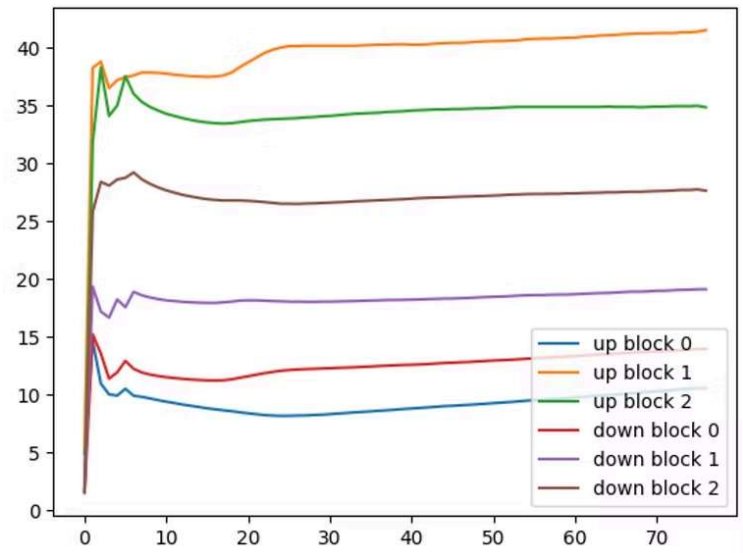
I found similar effects very extensively in text2image models and a bit also in text encoding models.

This shows the attention of the image to tokens of the text prompt, where y axis is the amount of attention mass and the x axis is the token index. Here we are displaying, for every text token of the prompt, what is the maximum amount of attention it has ever received from the set of all image tokens. Across different attention heads



We can see there is a massive amount of attention on the very first token of the sequence which is the BOS token. Importantly, because CLIP here encodes the text prompt with causal masking, the BOS token is *entirely* invariant to the text prompt itself. It is the same regardless of what you put in and thus confers no information about the request. So the remaining hypothesis is that similar to Attention Sinks, we are attending to "dummy token" just to reduce the weights we place on the other informative tokens, reducing the magnitude of the output, which we would probably rather have a mechanism for scaling more directly!

Incredibly interestingly, we can also look at the norms of the projected values of all the text tokens. The norm of the BOS token approaches 0 meaning it gets quite close to being a vector of all zeros. If we place the majority of weight on this vector of zeros, our resulting output is pretty close to a vector of all zeros as well!



Token norm after projecting through the value matrix. X axis represents token index. Y axis represents token norm

Though I cannot help but feel this is maladaptive in the ways this requires the value projection matrices to adapt to the BOS token vector, and that there are better ways to achieve this effect.

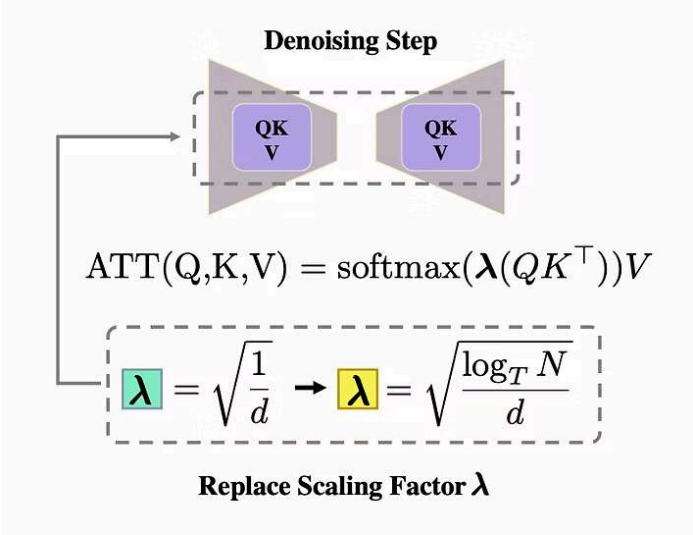
Sequence-Length Dependent Temperature.

The other paper that made me realize the importance this played was Training-free Diffusion Model Adaptation for Variable-Sized Text-to-Image Synthesis. And this where I begin to see a bit of a potential bug in attention.

Firstly, they reiterate our previous observations that the entropy/sharpness of attention depends on the sequence length.

> To accomplish these, our approach centers around the concept of entropy, which measures the spatial granularity of information aggregation. Specifically, when the attention entropy rises, each token is attending to wider spatial information, otherwise the opposite. Bearing that in mind, we establish the statistical relationship between the resolution (the number of tokens) and the entropy of attention maps. Our finding signifies that the attention entropy varies in correspondence with the quantity of tokens, which implies that models are aggregating spatial information in proportion to resolutions. We then establish our key interpretations based on the proportionality. In particular, since narrower attention is paid during the synthesis of low resolution images, models encounter difficulties

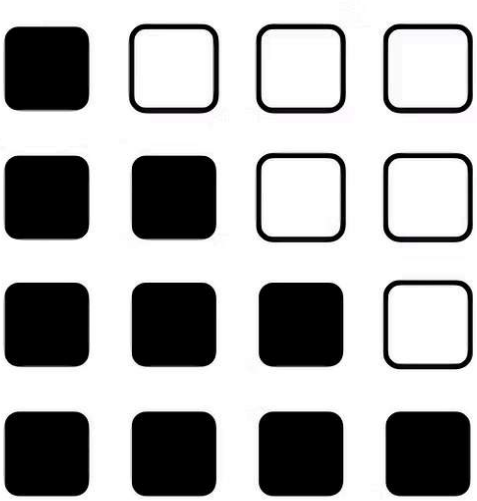
However, if we train on only one fixed sequence length, a common practice for image models at the time, we might expect the model to adapt the norms of the Queries and Keys to account for this. But when we go to test out other resolutions, we're now extrapolating to a domain we are ill-adapted for. Their proposal is to train on one resolution as normal, but then at inference time, rescale the temperature scaling to attempt to match the expected entropy we would have at our learned, trained resolution.



However, the need for a relative anchor makes it a bit of a band-aid approach and its common to train across multiple resolutions simultaneously. We can't match our scaling to that of a trained length if this is a moving target. We'd really like to have this built into the attention equation itself at train time.

On the topic of handling multiple sequence lengths simultaneously, language models, which train with causal masking, zero out the logits of tokens to the right (future) for a given token.

So technically, if we're training at a sequence length of say, 4096, attention is effectively batching queries that attend to a sequence of length 1, a sequence of length 2 ... all the way to a sequence of 4096 simultaneously.



As stated before, our query and key projection matrices can adapt themselves to output queries and keys that have norms allowing to achieve desired levels of entropy.

But here's the problem, the same matrices are being used for all tokens. Some sequence lengths may prefer larger norms and some smaller, but we can't adapt in both of these directions at once over these single shared weights! This is not to say we need a set of weights for each possible length, but there may be something we can do.

> Hypothesis: This may be correlated to issues of entropy collapse and instability in attention layers seen in LLM training. The sequence of 4096 may request a gradient direction leading to lower entropy via higher Q and K norms while the sequence of 5 may request higher entropy. Conflicting update directions may guide training towards a poorly adapted and fragile compromise.

A proposal:

1. The scaling factor of attention should be a function of sequence length.
    a. This could be learned or scaled by an equation, leaning towards the equation approach as it seems there is a good way to understand this.
2. There should be a bias in the denominator that is either a learned constant, or possibly also a function of sequence length.
    a. This I have less intuition on though I lean towards a learned approach as it may depend on the typical range of values the logits take on.

With respect to 1, Petar Veličković had a paper on this very matter, for how attention varies with sequence length and how we can recover sharpness through scaling.

In conclusion, while we have now detailed explanations for how attention performs after the fact, its inception was somewhat a fluke in terms of justifying some of the specifics of its design. A common and effective trend in ML, to try first and find out later.

Attention is obviously performant as is, but it appears there's potentially some places to grow if we forego the probability framework softmax lends itself to and possibly resolving something I would even deem a bug.

Recent Posts See All

```
@misc{smith2024attndiffusion,
  author = {Smith, Ethan},
  title = {Softmax Attention is a Fluke},
  url = {https://www.ethansmith2000.com/post/softmax-attention-is-a-fluke},
```