Logit and Attention Lens

In this section, we have focused on two approaches: a) Logit Lens, and b) Attention Lens. Let's begin with a bit of a background on the two approaches first.

a) **Logit Lens** (nostalgebraist, 2020) is a method that directly accesses each layer's outputs and allows us to study what the model predicts at each processing step. It is based on the insight that the hidden vectors passed between the intermediate layers of a model have the same dimension as the output vectors, and applying the same unembedding to these intermediate vectors, we obtain the predictions made by these layers. Thus, we are able to 'see-through' the model's operations and study how the output is formed across the layers. We can for example see how much each layer contributes to the final output, how the predictions change across the layers, what the top predicted tokens at any stage are as well as how likely the model considers them to be.
b) **Attention Lens**, however, refers to the method where we plot attention patterns of the various induction heads, then try to experiment with certain systems having predetermined mathematical aspects/qualities hoping to get some representation of said aspects/qualities from the induction heads simply by observing the attention patterns. We also experiment with "value-weighted" attention patterns and observe some interesting phenomenon that remains unexplained. Basically, this method lets us see how or what the model is "thinking" when figuring out the symbolic expression of a certain n-dimensional trajectory.

We have not used hooks to obtain the attentions and intermediate tokens and logits, but instead developed our own functionalities within our own fork of the original ODEFormer library. Some key functions have been developed in this library that allow us to directly plot token charts, and obtain intermediate tokens/logits, and also attentions (value-weighted as well as the usual ones). Please note that the library lacks

updated documentation which we plan to add in the coming few weeks. Now, with this amount of context and understanding, let's discuss the results we have till now.

Intermediate Results

1. Sign Prediction in Simple 1D Systems

While we did not stare at the token charts hard enough, we do have a few interesting discoveries here. Consider the figure below (Fig. LALens1).

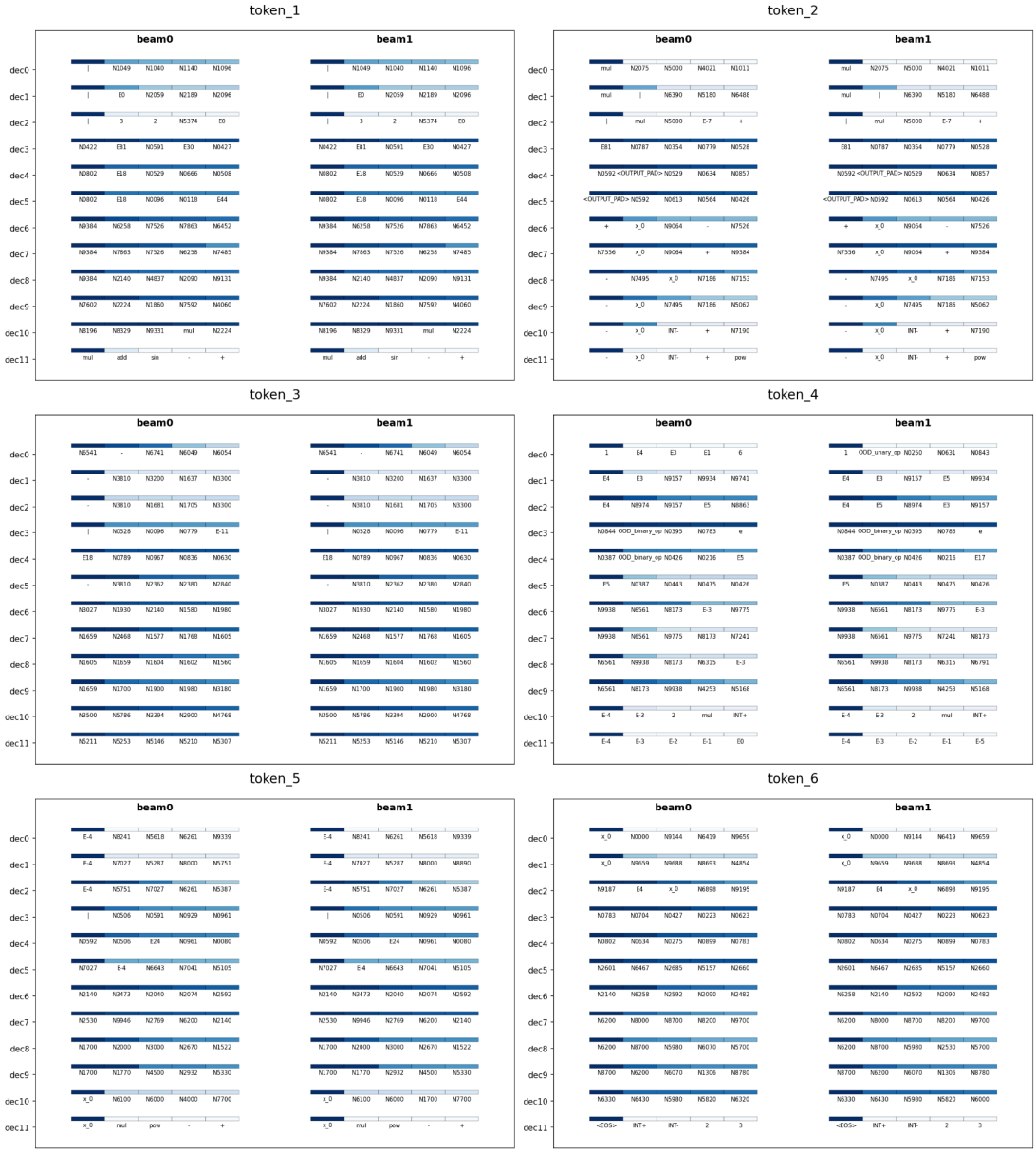

Fig. LALens1: A compact chart showing token evolution across layers and beams.

The figure shows how tokens and the corresponding confidence scores vary across beams and decoder layers. For this experiment, we had a simple 1-dimensional decreasing exponential system primarily because we had observed earlier this system results in a simple 6-token output. Also, we opted for a low beam size (2) and temperature (0.1) for simplicity purposes. A general observation is that in the initial layers across all beams, the model is pretty confident on the wrong token. In the successive layers the probability is spread across several tokens which is why we see a more uniformly blue distribution, and in the last few layers the model becomes more confident on the right token. However, in the token_3 subplot, the prediction of the constant does not become very confident even in the last few layers like in the other subplots. This is an obvious result because these constants are in the order of 10^{-4} and small changes in magnitude does not really affect the overall expression that significantly.

Next, we zoom in a bit on the sign prediction. We have observed that in cases of both decreasing and increasing exponential systems, decoder layer 6 predicts '+' which is corrected to '-' in case of a decreasing system only in decoder layer 8. This is a bias towards '+' presumably because most numbers in the training data were positive. We have observed the attention patterns in the various heads as well and found some heads to be capturing the direction of maximum to minimum magnitude or vice versa in the trajectory which according to us was key in determining the sign.

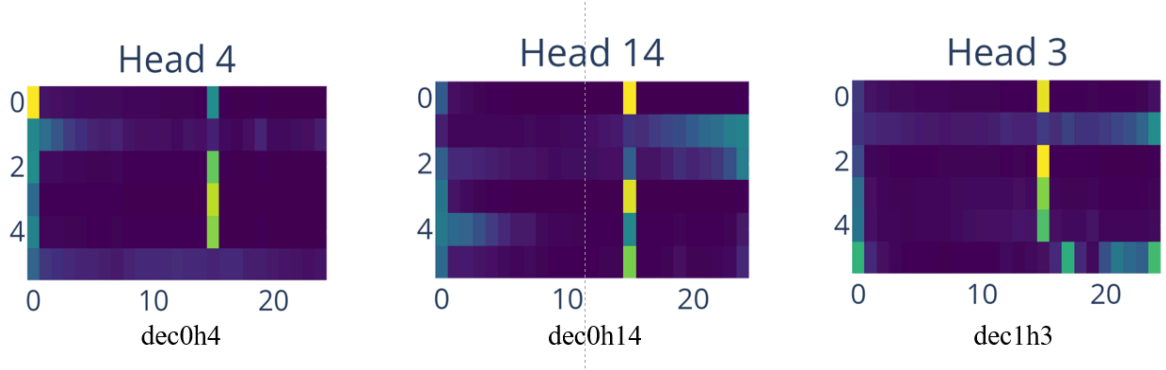

Fig. LALens2: The second row from the top shows interesting patterns.

In Fig. LALens2, we can see that the rows corresponding to the value 1 on the y-axes show that pattern of minimum to maximum or vice versa. Row 1 represents the token

‘+’ or ‘-’. These three patterns are from the attention heads of layers in which the ‘+’ or ‘-’ was being predicted for the first time. You can also notice a dominant column of attention in all three plots. This will be discussed later. We have also looked at the logits of the ‘+’ and ‘-’ tokens and their evolution through the layers.

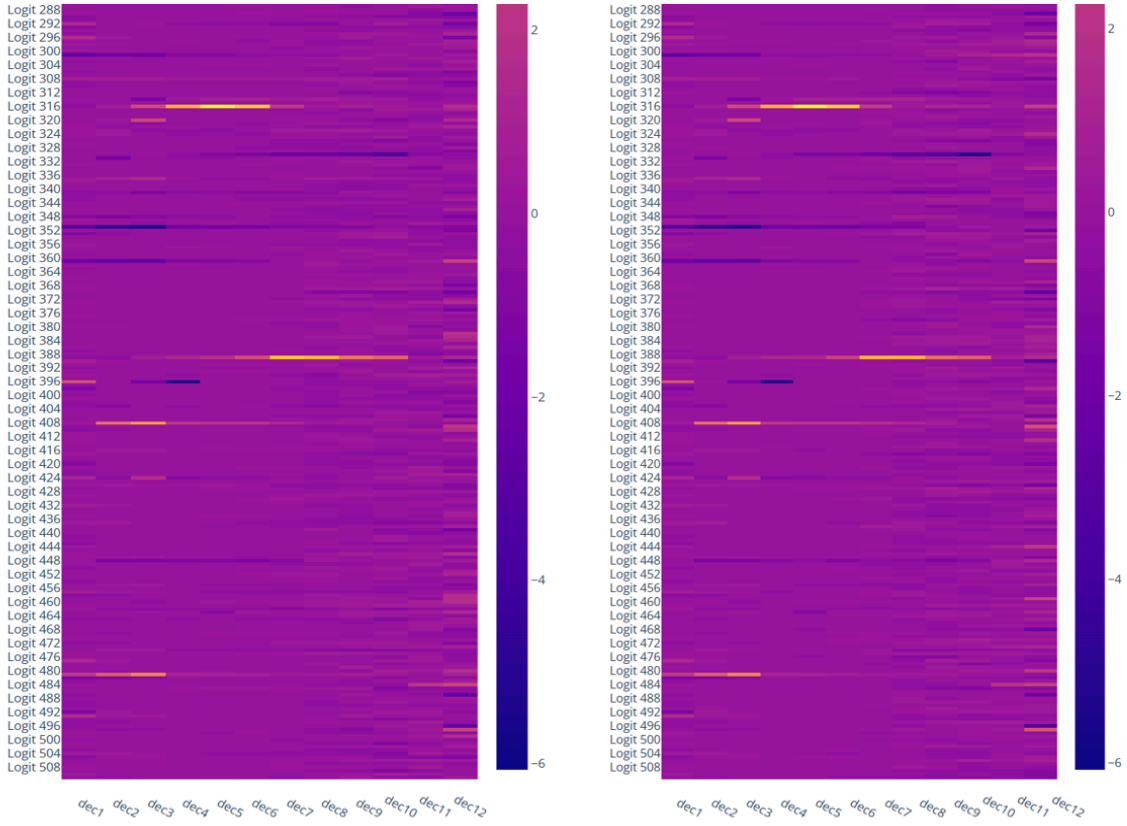

Fig. LALens3: Left one is for the increasing system, while the right one is for the decreasing system.

There is no immediate difference between the two plots, however, since ‘+’ was being predicted from layer 6 onwards, and logit 389 also seems to be more activated from layer 6 onwards (here 7 because naming begins from 1). Thus, we conclude that most of the information about the sign is being carried by logit 389 alone.

Now, let’s discuss the attention patterns in more detail. Broadly, we have found concrete proof that null attentions or attention sinks exist in the ODEFormer, a point that is being mostly attended to but seem to not be any special at first glance, and attention

heads that capture various mathematical qualities of the input trajectory that is helpful for the model to predict the symbolic expression.

2. N1010 as "Default" Token

During our experiments, we have seen (oftentimes with higher beam sizes) that after we get an "<EOS>" token in one beam, but other beams have not yet yielded an "<EOS>" token, the beam gives an "add" token or another "<EOS>" token as the next token. If all other beams have still not yielded "<EOS>", then the beam that has finished predicting an expression will continue to give the "N1010" constant token as a kind of "default" token. For example, consider the following three plots in Fig. LALens4 below for predicting three consecutive tokens.

	beam 0	beam 1	beam 2	beam 3	beam 4	beam 5	beam 6	beam 7	beam 8	beam 9	beam 10	beam 11	beam 12	beam 13	beam 14
decoder 0	mul	x_0	x_0	x_0	x_0	add	add	x_0	add	x_0	add	add	+	+	INT+
decoder 1	mul	x_0	x_0	x_0	x_0	N0540	E6	x_0	E6	x_0	E6	E6	+	+	INT+
decoder 2	|	E4	E4	E4	E4	N0589	N0589	E4	N0589	E4	N0589	N0589	+	+	2
decoder 3	E81	N0623	N0623	N0623	N0623	N0779	N0779	N0623	N0779	N0623	N0779	N0779	N0606	N0606	N0243
decoder 4	N0634	N0634	N0634	N0634	N0634	E18	E18	N0634	E18	N0634	E18	E18	N0426	N0426	2
decoder 5	N0383	N6258	N6258	N6258	N6258	N0991	N0991	N6258	N0991	N6258	E18	N0991	N1647	N1647	2
decoder 6	E-7	N6258	N6258	N6258	N6258	N9384	N9384	N6258	N9384	N6258	N9431	N9384	N1101	N1930	2
decoder 7	N7556	N8000	N1011	N8000	N1011	N9011	N9011	N8000	N9011	N8000	N7526	N8400	N1101	N1850	2
decoder 8	N5065	N1097	N1174	N1097	N1174	N6884	N6884	N1097	N6884	N1097	INT-	N6884	E-5	N1819	2
decoder 9	pow	N6330	N8780	N6330	N8700	-	-	N6330	-	N6330	1	-	N2890	N1030	2
decoder 10	x_0	|	|	|	|	-	-	|	-	|	INT+	-	N1380	N7900	2
decoder 11	x_0	<EOS>	<EOS>	<EOS>	<EOS>	-	-	<EOS>	-	<EOS>	INT+	-	N1016	N3552	2

	beam 0	beam 1	beam 2	beam 3	beam 4	beam 5	beam 6	beam 7	beam 8	beam 9	beam 10	beam 11	beam 12	beam 13	beam 14
decoder 0	x_0	N0216	E7	N0216	N0216	-	-	N0216	-	N0216	INT-	-	N9077	N3440	2
decoder 1	x_0	E7	1	E7	E7	-	-	E7	-	E7	INT-	-	N9691	N4456	2
decoder 2	E4	|	E4	|	|	-	-	|	-	|	1	-	N5323	N4456	2
decoder 3	N0623	N0422	N0422	E81	N0422	|	|	E81	|	E81	N0243	|	N0844	|	|
decoder 4	N0275	N0802_binary	N0802_binary	N0802_binary	N0802_binary	N0096	N0096	N0802	N0096	N0802	1	N0096	E5	E5	|
decoder 5	N6447	N0802_binary	N0802_binary	N0802	N0802	N1867	N1867	N0802	N1867	N0802	1	N1867	E5	E5	E2
decoder 6	N6179	N6258	N6258	N6258	N6258	N1930	N1930	N6258	N1930	N6258	1	N1930	E-4	E-4	N3760
decoder 7	N1041	N7526	N7526	N7526	N7526	N2939	N2939	N7526	N2939	N7526	1	N2939	E-4	E-4	N1079
decoder 8	N6179	N7526	N1174	N7526	N1174	N1010	N1010	N7526	N1010	N7526	1	N1010	E-4	E-4	N1050
decoder 9	N1060	N1770	N1770	N1891	N1770	N1830	N1830	N1891	N1830	N1770	1	N1830	E-5	E-4	N6330
decoder 10	pow	x_0	x_0	N6040	x_0	N7060	N7060	N6040	N7060	x_0	1	N7060	E-3	E-4	add
decoder 11	pow	<EOS>	<EOS>	add	<EOS>	N6420	N6420	add	N1300	<EOS>	1	N1300	E-3	E-4	add

	beam 0	beam 1	beam 2	beam 3	beam 4	beam 5	beam 6	beam 7	beam 8	beam 9	beam 10	beam 11	beam 12	beam 13	beam 14
decoder 0	pow	N1010	N1010	N1010	N1010	N1700	N1255	N1010	E-4	N1010	1	N1333	E-3	E-4	add
decoder 1	pow	N1010	N1010	N1010	N1010	N1700	N1410	N1010	N1009	N1010	1	N1310	E-3	E-4	N0540
decoder 2	5	N1010	N1010	N1010	N1010	N1438	N1547	N1010	N2000	N1010	1	N1547	N1700	N5751	N0589
decoder 3	N0223	N1010	N1010	N1010	N1010	N0257	N0257	N1010	N2015	N1010	1	N0257	N6592	|	N0779
decoder 4	N0857	N1010	N1010	N1010	N1010	N0189	N0257	N1010	N2015	N1010	1	N0257	N6592	|	E18
decoder 5	N0060	N1010	N1010	N1010	N1010	N0257	N0759	N1010	N2000	N1010	1	N0825	N6258	N6258	E18
decoder 6	N6452	N1010	N1010	N1010	N1010	E-2	E-65	N1010	N2000	N1010	N8200	E-65	N2839	N1041	E-7
decoder 7	add	N1010	N1010	N1010	N1010	E-2	E-45	N1010	E-2	N1010	N1041	E-45	N2970	N1041	N8733
decoder 8	N1111	N1010	N1010	N1010	N1010	E-2	E-2	N1010	E-2	N1010	N1040	E-2	E-5	N4137	N9954
decoder 9	add	N1010	N1010	N1010	N1010	E-2	E-2	N1010	E-2	N1010	mul	E-2	E-4	N1013	+
decoder 10	add	N1010	N1010	N1010	N1010	E-2	E-2	N1010	E-2	N1010	mul	E-2	mul	mul	+
decoder 11	add	N1010	N1010	N1010	N1010	E-4	E-4	N1010	E-3	N1010	mul	E-2	mul	mul	+

Fig. LALens4: N1010 is predicted after expression prediction ends in some beams.

In the figure's first subplot, we can see that beams 1-4 and 7 and 9 gave the "<EOS>" token as output, but other beams did not. Due to this, when the other beams carry on with the next token prediction, these 6 beams either give the "<EOS>" or "add" token as output. And if the other beams still have not finished, these 6 beams continue giving the "N1010" token as a "default" token till all other beams give the "<EOS>" token hence completing the process of token generation, after which post-processing takes place.

3. Existence of Null Attentions

We have observed that Attention sinks are present in the attention plots. These are input trajectory points that are more attended to compared to other points, but they do not transfer much information/value as seen from the value-weighted attention plots. What is value-weighted attention? Taken from "A Mathematical Framework of Transformer Circuits" (Elhage et al.) this is a method where we scale the attention in the plots by how much information is transferred which is obtained from the value vectors.
Value-weighted attention shows us exactly how much value is being transferred by attending to the input points. So, if a certain point is being mostly attended to by all or most tokens (or the points themselves in case of encoder self-attention) in the usual attention plots but not much in the value-weighted ones, it means that point is not contributing much to the prediction despite being mostly attended to. Here, we say that point is an attention sink.

However, even interesting is the discovery of a different class of points which we call **MAT points** (mostly-attended-to points).

4. The MAT Saga

These are points that are mostly attended to by all (or most) tokens as seen in the decoder's value-weighted cross-attention plots. The fact that these points light up almost similarly in the value-weighted plot as in the normal ones, show that these points are indeed transferring information that is needed by the model (specifically the decoder layers) to correctly predict the symbolic expression. For example, let's consider the

attention plot in Fig. LALens5. In the top subplot, we can clearly see that there are a few input points/columns that are more lit up than the others. These points are also transferring information so the attention on them is not completely useless.

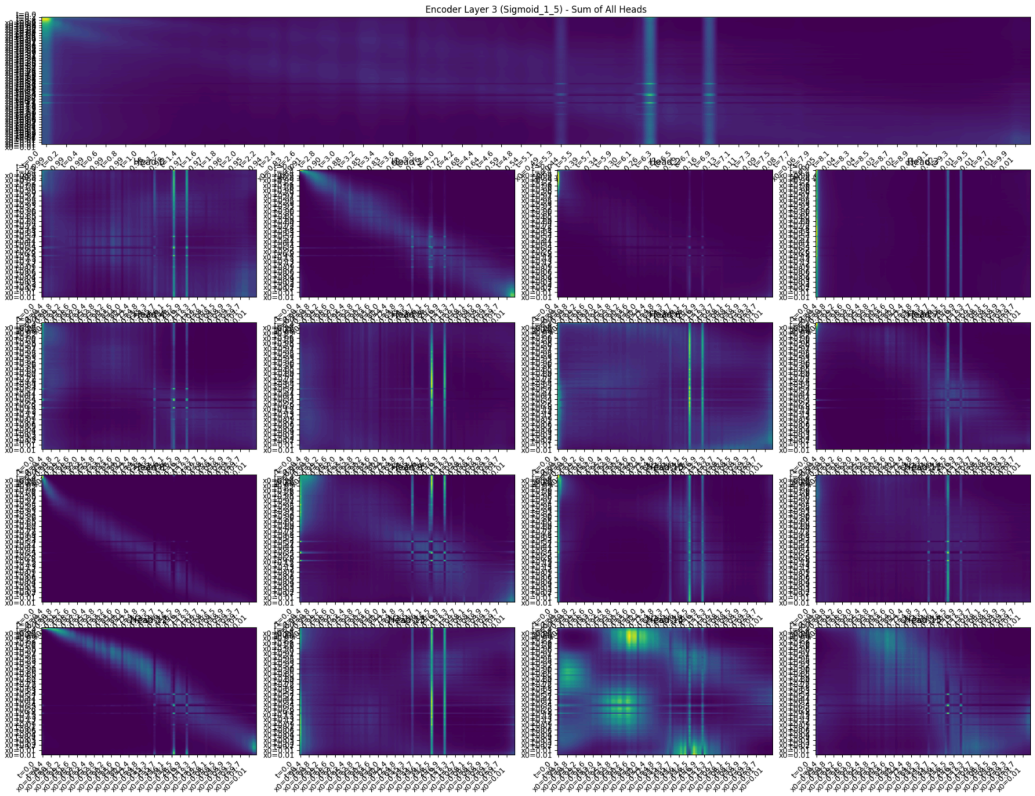

Fig. LALens5: The above one is the normalized summation (a combined view) of all 16 heads.

Lighting up is fun but now you might be questioning what is that one defining mathematical quality of the MAT point/column. We have plotted the norm of all of the points of the harmonic sine-cosine system, and from the plot Fig. LALens6, we see that the MAT point has a very low norm.

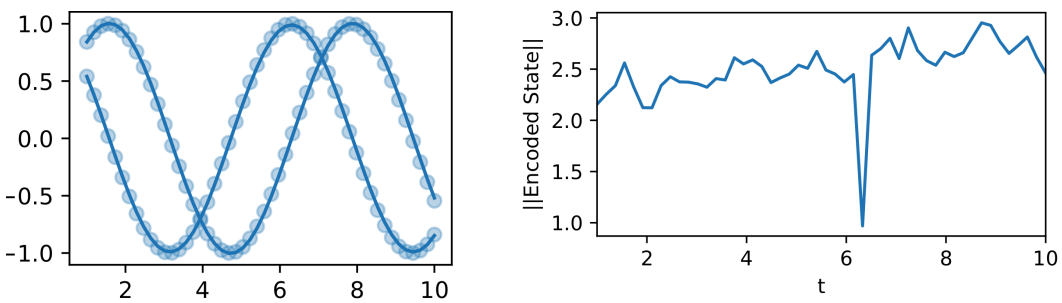

Fig. LALens6: MAT column has a lower norm value.

Let's also take an example now to show the difference between value-weighted attentions and normal attentions. In Fig. LALens7, we clearly see more information transferred by other points but the MAT point is still clearly visible. The attention plots in the figure correspond to the system in the previous figure.

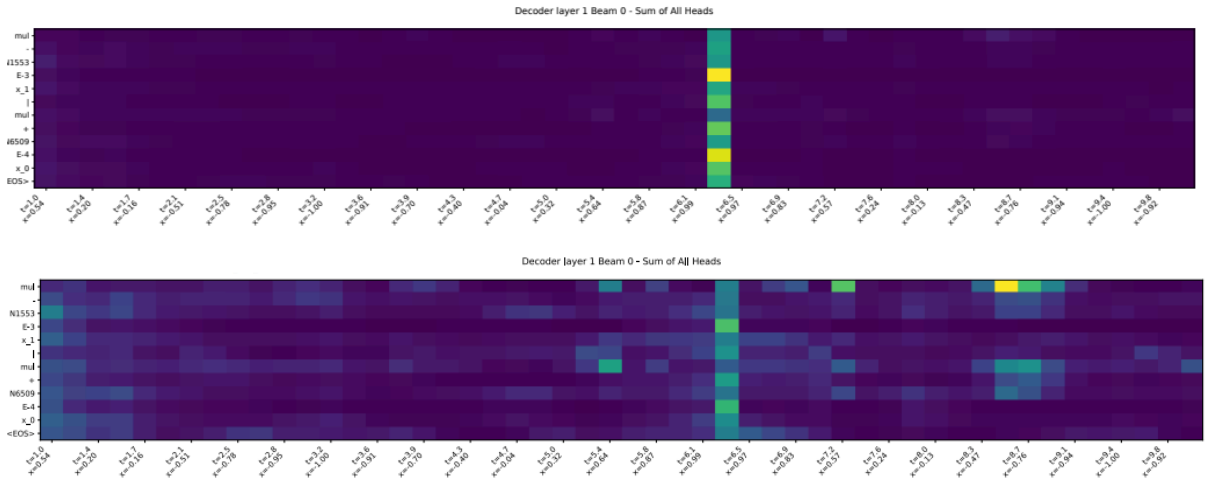

Fig. LALens7: The bottom one is the value-weighted attention plot.

We initially observed that a few points or just one point (most of the time) was mostly attended to by all input points or tokens, but this was random. The randomisation was done in the ODEFormer to get random points as input in case the original input was very long. Due to us experimenting with very short trajectories, we did not require the randomisation and on turning it off, we observed that the MAT point is constant. What do we mean? Take 'x' number of input points from various systems across dimensionalities and feed it to the ODEFormer to get the predicted symbolic expression. In all cases, the MAT point remains the same. Change the number of input points to 'y' and the MAT status will shift. This shifting is also predictable according to what we have seen in our experiments. Roughly the 60% point in the input trajectory is going to get the MAT status. So if we have 25 input points in the trajectory, point 15 is going to become the MAT point. One can think at this moment that this sounds like there was some split internally in the trajectory in the ratio of 60:40, and that the MAT point lighting up seems like the Attention sink of the latter 40% of the trajectory since sinks are mostly the first point. This is a great idea but what we have observed in some cases is that there is no abrupt change in the attention pattern before and after the MAT point.

In some cases we see that attention is being gradually increased as we move closer to the MAT point from the left, and then decrease in a same gradual manner when we move past the point. If the hypothesis that there is some kind of internal training and testing "split" we should not have seen that gradual attention build-up around the MAT point. And this pattern is only visible in the value-weighted plots, suggesting that even though points around the MAT point are not that attended to, they do transfer helpful information.

Additionally, we conducted a few tests to get more insights on the nature of the MAT point. Please note that the attention in the following plots are not value-weighted to clearly differentiate the MAT point.

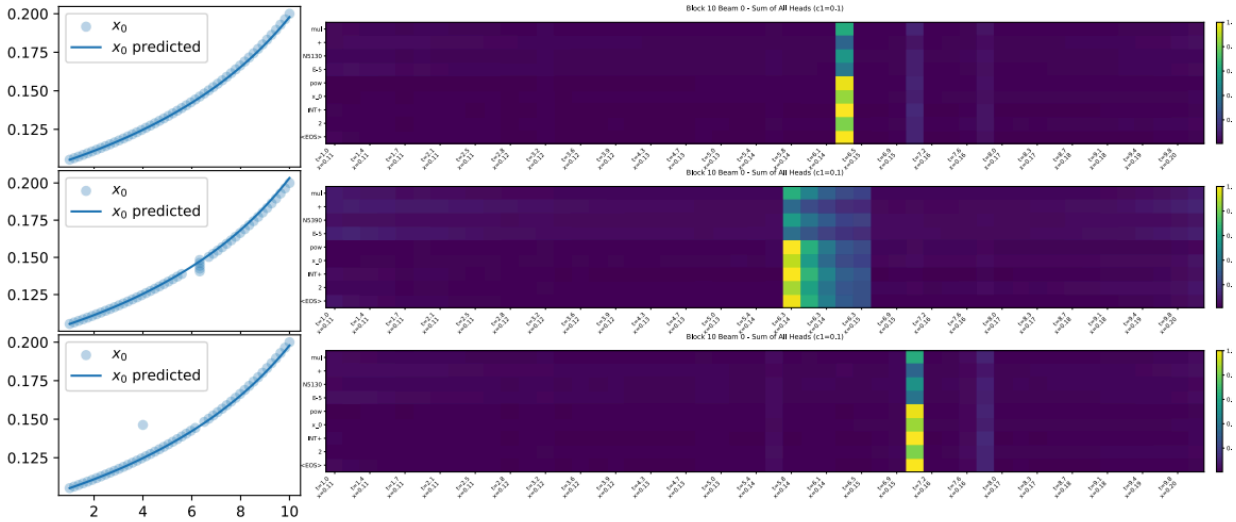

Fig. LALens8: The MAT status seems to depend on a certain time step and not trajectory value.

Before the experiment we believed that the MAT status is because of the value of the point in the trajectory. However, when we modified the trajectory to have different values for points for the same time step (i.e. made the slope infinity for some portion of the input trajectory) the model seemed to be attending to all of the points more than the others. As seen in the second subplot of Fig. LALens8, the model tries to give the MAT status to all of the points at that one particular time step. However, it does not do that and instead we see a gradual decay in the attention the points receive as we move right. Now, let's move to the third experiment and subplot, where we changed the MAT

point's value to be something entirely different. In this case, we see some other point get the MAT status. But wait a second! It's what we call the second MAT point that got the MAT status after we shifted the original first MAT point by some degree. What is the second MAT point you ask? When we look back at the first subplot we see not only the MAT point having higher attention than the others, but there are two more points that get somewhat more attention than the others. We call these the MAT candidates. And as we saw, changing the MAT point's value to some extent results in one of the candidates getting the MAT status instead. Now, a great question would be by how much should the MAT point be shifted?

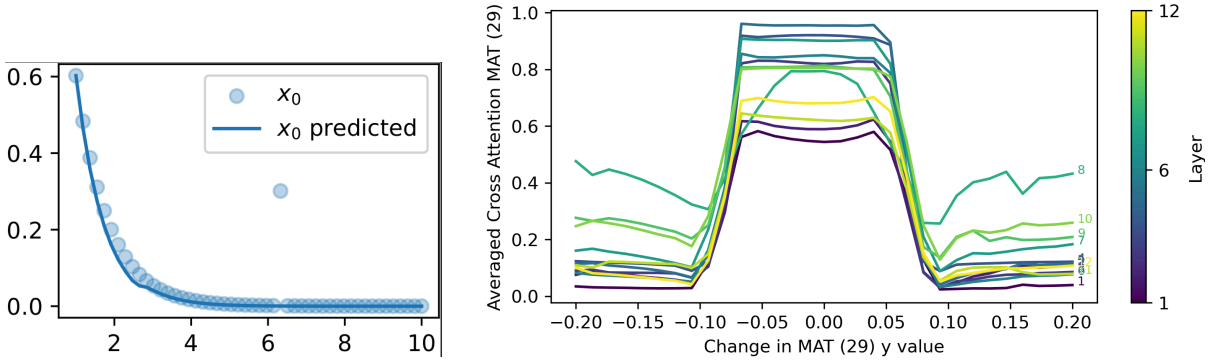

Fig. LALens9: The MAT status seems to depend on a certain time step and not trajectory value.

Let's take a look at Fig. LALens9. Changing the MAT value by anything lesser than or equal to approximately |0.05| retains most of the usual cross-attentions and does not result in the shifting of the MAT status to some candidate. Increasing the value more than |0.05| however shows a significant dip in the averaged cross-attentions of the MAT point, and the MAT status is given to some other candidate point.

One last thing that we observed in some value-weighted attention heads in several systems, is that there exists some heads in which the MAT point is less attended to than the other points. These heads mostly capture some mathematical aspect of the input trajectory, but even if the MAT point should have some attention logically (perhaps it is the maxima or minima or something else), it does not.

5. Attention Heads capture Mathematical Features

Just like we have seen development of low level and high level feature development in the attention heads of vision transformers, we observe that similar phenomenon occurs in ODEFormer as well. The ODEFormer too has specialized attention heads that capture some mathematical aspects/qualities/features of the n-dimensional input system. We are going to look at the self-attentions of the encoder blocks only since it has input points in both axes and we are interested in seeing what data is "encoded" into the trajectory in the encoder layers, before the decoder layers engage in cross-attention while predicting tokens. We treat the ODEFormer as a multi-modal transformer, because it takes input in the trajectory space and gives output in a well-defined token space.

Now, let's take a closer look at some of the features in some systems.

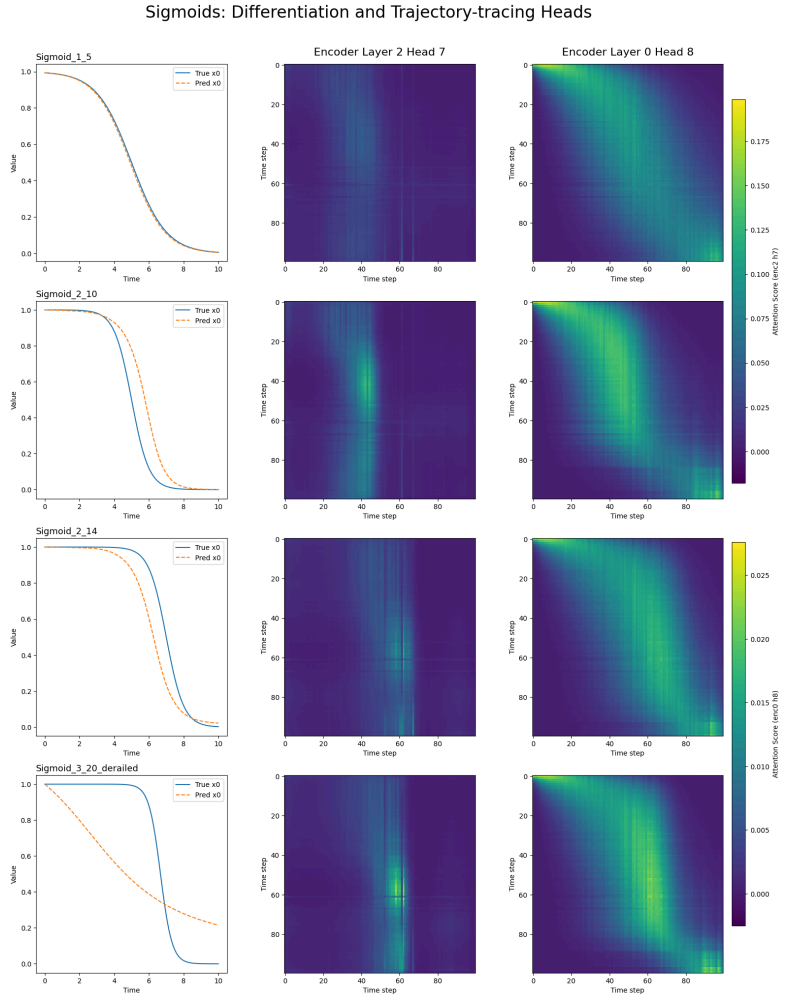

Fig. LALens10: Specialized Attention Heads spotted during the Sigmoid Experiment.

Some context first: In Fig. LALens10, Sigmoid_a_b means the input trajectory was that of a sigmoid following the equation:

$$trajectory = \frac{1}{1 + e^{(a*times-b)}}$$

When changing the values of a and b, i.e. changing the position and steepness of the sigmoid curve, we noticed that one head, encoder layer 2 head 7, seemed to pick up on differentiation or the portion of the curve where the slope is not 0. Besides this, we also noticed that encoder layer 0 head 8 seems to trace the input sigmoid trajectory itself at first glance. On changing the sigmoid curve to go from 0 to 1 (instead of 1 to 0 in the figure) the tracing head does not seem to perfectly trace the trajectory, but the pattern does shift with shifts in the transition portion.

In Sigmoid_3_20 you might notice that the prediction doesn't really look like a sigmoid curve at all. This is true and we have found there to be some degree to which the ODEFormer can predict the sigmoid curve accurately. Beyond a certain degree of steepness, the ODEFormer derails quickly and badly, telling us that the transformer is unable to handle quick and non-periodic transitions in the input trajectory. Though the model managed to capture significant sigmoid-like features in case of Sigmoid_3_20, it failed completely in case of Sigmoid_8_25 (which is way steeper). There are no meaningful patterns in the attention heads, just noise.

While differentiation is a high-level concept, tracing is a relatively low-level low-effort concept, and we hypothesize this to be the reason why a head in layer 0 traces the trajectory, but differentiation is captured by a head in layer 2. To support this, we also found heads that capture the maximum and minimum in a trajectory in layer 0. These findings might not be true for all systems, but from what we have observed these are true for some simple systems.

While observing the cross-attentions after feeding the ODEFormer the trajectory in Fig. LALens6, we observed a few more attention heads that seemed to capture the minima and maxima.

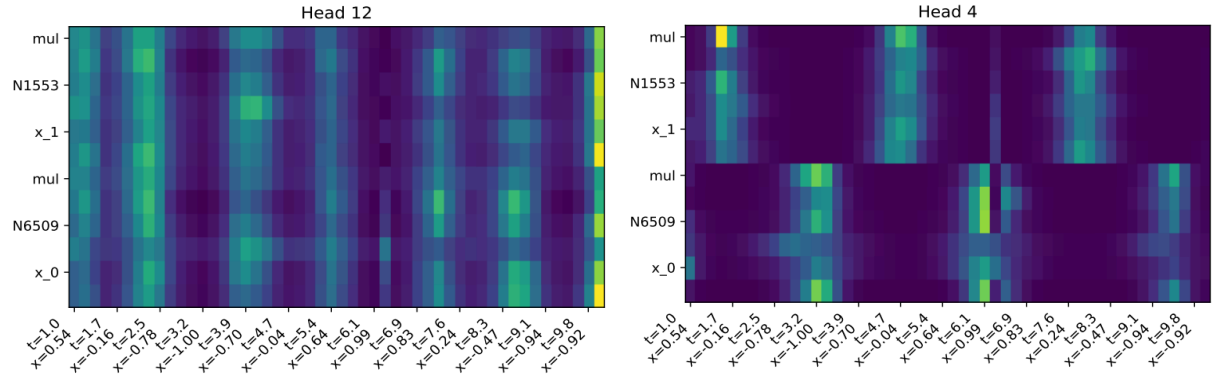

Fig. LALens11: Heads capturing maxima and minima with/without dimension separation.

While the left attention head in Fig. LALens11, decoder layer 5 head 12, captures the minima and maxima but shows no separation of dimension considering that the input trajectory belongs to a 2D system. On the right however, we see that decoder layer 6 head 4 captures the minima and maxima with the dimension separation. While the pattern in the left head is rare and only comes before we see the pattern in the right head around the middle layers. The pattern with the dimension separation is more common in these kinds of systems.

6. Self-Inhibiting Behaviour of the Encoders

We plotted the OV matrix as obtained by multiplying the output weight matrix with the value weight matrix for each encoder layer, and the results are in Fig. LALens12. The most striking feature in all four encoder layers is the presence of a leading diagonal that is significantly more negative than the other values. In addition to this, the diagonal seems to become more and more negative as we move from layer 0 to the final layer of the encoder.

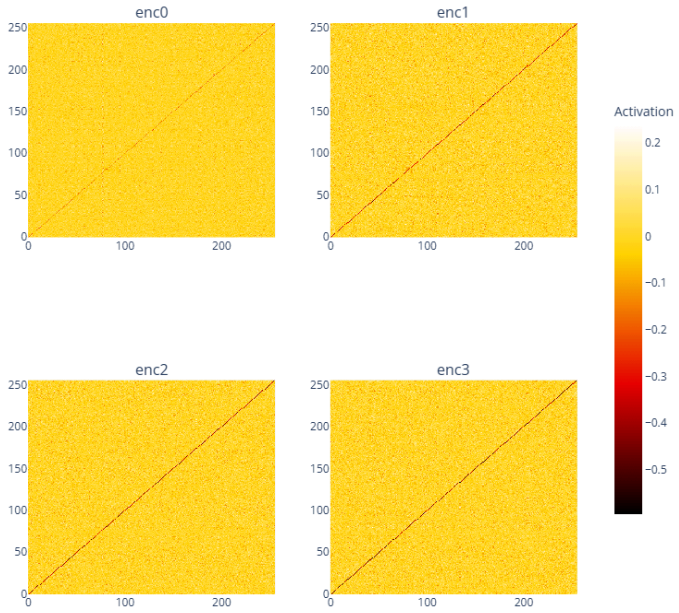

Fig. LALens12: Presence of a significantly more negative diagonal in the OV matrices.

This coincides with the formation of the MAT point, which begins forming from the second encoder layer, and only becomes more and more pronounced as we go towards the final encoder layers. We think this negative diagonal to be some form of self-inhibition, something like decreasing the effect of itself, and we are yet to ascertain the importance of this self-inhibiting behaviour in the formation of the MAT point (if at all).

This strikingly negative leading diagonal is present in the OV plots of the encoder layers, across different systems.

Future Work

Even though we have made some interesting discoveries, we are yet to find concrete evidence of how the ODEFormer exactly predicts the symbolic expression. We have found low and high level features in the attention heads, and dimension separation, and the very intriguing MAT point, but we see much more interesting patterns in the attention heads in case of 2D systems from ODEBench, that we will make sense of in the coming few weeks. We plan to use other methods as well in addition to what we are

currently using. Methods that allow us to piece together all our tiny interesting discoveries and help us write a complete story. We are already working on a few different ideas: re-constructing trajectories from the latent space, isolating attention heads to observe their singular effect on the trajectory, conducting more in-depth ablation studies to help us understand the role of the encoder layers. The future seems to be awesome!