Attention Revisited

Laurel Koenig

Abstract

One of the shortcomings of current deep learning architecture is that it fails to solve complex problems, particularly problems requiring extrapolation. Cognitive architecture has been a solution to such problems in the past, but it requires extensive training. There has been recent work with transformers that suggests that such architectures are capable of forms of extrapolation. [1] However, there are other studies that conflict with this and instead suggest that the transformer, arguably the most successful deep learning architecture of the last five years of research, lacks extrapolatory capabilities. [2]

Upon examination, the repository was found to be incomplete, so we recreated the missing components and repeated the testing done in "Attention Is Not Enough".[2] The results were mostly consistent with the previous study indicating that their finding were likely correct.

I. Introduction

DESPITE being created in our image, computers struggle with thought patterns that tend to come naturally to people. Where we unknowingly learn and connect dots subconsciously, our current deep learning architectures struggle and often outright fail at tasks requiring extrapolation. This fluid ability to apply ideas learned from past experiences is known as generalization. Interpolation is a form of generalization that determines an unknown within a set. In mathematics, spline functions are a form of interpolation because they use a set of points to determine an unknown function within those points. Deep learning has been wildly successful in solving tasks involving interpolation.

Extrapolation, on the other hand, is a point of struggle for deep learning architectures. Extrapolation is also a form of generalization, but unlike interpolation, extrapolation focuses on extending beyond an unknown. Reasonably applying past experience to never before seen circumstances to make predictions. In the past it has been said that extrapolation isn't possible to replicate through machine learning, but more recent studies have shown some promise. [3]

One solution is to use biologically plausible working memory (WM) models. To provide detail and insight into the function of the brain, these WM models make use of many biological mechanisms. While these are extremely effective, they need extensive training as well as increased memory and central processing unit (CPU) requirements.

Some studies have suggested that transformer architecture is capable of such generalization. [3] However, other recent studies have shown that the transformer breaks down when given more complex extrapolation tasks. [2]. In this study we repeat the tests performed in "Attention in not Enough" to evaluate their findings about the limits of the transformer.

II. Background

A. Artificial Neural Networks

Artificial neural networks (ANN) are machine learning approaches inspired by the physical properties of biological neural networks such as the brain. Typically, they are composed of layers of nodes, or artificial neurons, that are interconnected with activation weights. Each layer does something slightly different to the data so that it might take many layers for a network to properly perform a complex task.

When a network is trained, it processes examples of the task it should be solving and corrects the weights between nodes based on the examples. Weights are initialized randomly before training, and then changed slightly with every training example. How the weights should change each time is determined by the gradient part of the optimizer.

B. Long Short-Term Memory

Long Short-Term Memory (LSTM) is a particular type of Recurrent Neural Network (RNN) that was developed in response to the exploding and vanishing gradient problem. [4] RNNs maintain information past a single time step which allows that information to accumulate and impact the current time step. The vanishing gradient problem is an issue that can occur during the training of RNNs where the network weights stop changing causing slow and ineffective training. The exploding gradient problem is the opposite; the error norm is too large and that causes large updates to the weights. LSTM networks attempt to circumvent this problem with clever use of input, output, and forget gates so that only useful information is retained by the network. [4]

C. Transformers

Transformers are an encoder/decoder architecture reliant on attention mechanisms that were proposed in the 2017 paper "Attention Is All You Need".[3] By leveraging the self-attention mechanisms this architecture is designed to process sequences by looking at everything at once rather than one thing at a time. It is also known to be more parallelizable than recurrent models, and has been shown to do exceptionally well on tasks like language translation with minimally comparative training. [3]

1) Attention: Attention mechanisms are named as such because they draw their inspiration from cognitive attention in that they should use a gradient descent to determine what part of the data should have more attention devoted to it. More tangibly, they can be thought about as mapping key-value pairs and a query to an output. The keys, values, query, and output are all vectors and a weighted sum of the values is used to compute the output. To assign a weight to a value a compatibility function of the related key and query is used. [3]

D. Encoder/Decoder Architecture

A nice way to think about how encoder/decoder architecture works is to imagine that the encoder and decoder are two different people trying to communicate a message across languages. The input is the first language, the state or encoding is the shared language, and the output is a third language. In this analogy the encoder only knows the input language and the state language while the decoder only knows the state language and the output language. To continue this analogy, if the encoder and decoder were people working together regularly they would be expected to get better at understanding the intricacies of their shared language and how each of them uses it. They would then be expected to get more accurate translations. Because of this encoders and decoders always work and train together as pairs.

1) Teacher Forcing: When models use the output from previous steps to inform the current step it creates a training problem where a wrong guess can cause the model to end up way off track. Teacher forcing is a method of training where the model checks to see if it is correct at every step rather than waiting until the end. When it does this it computes each step from the correct answer rather than the previously computed one. This means that even if it get off track early in a training sequence the model weights aren't hurt by that.

This is, of course, not a truly accurate portrayal of how much a model has learned though, so we turn off teacher forcing during testing. Turning off teacher forcing on encoder/decoder models is done in two main ways. The first is through masking, and the second is to train using a coupled model but test using separate encoder/decoder models.

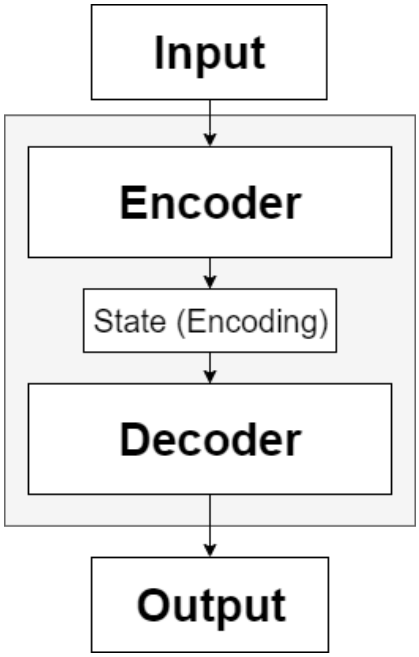

Fig. 1: Encoder/Decoder Architecture

E. Previous Results and Paper

In "Attention Is Not Enough" they used a nested encoder/decoder structure to test the limits of the transformer. The created four different configurations of inner and outer LSTM and Transformer components and then trained and tested these on four different tasks designed to test extrapolation. They used a WM as the control against which to test their models. [2]

While going through their repository it was discovered that parts of the code base were missing. Specifically, the code for creating and saving the outer models along with the script for one of the four nested models could not be found. Despite the code to generate it being missing, all of the outer models had been saved along with their weights. The missing nested script, however, prevented a simple verification of their results.

III. Re-implementation

First, all of the duplicated portions of the scripts from the prior implementation were moved into their own files. This included the transformer class blocks for both inner and outer, embedding classes, as well as a number of key functions including, but not limited to, functions for processing the data and calculating the accuracy.

After that the remaining scripts were rewritten with the new import files. They were written as notebooks first for easier testing and access to model metrics. Once the notebooks were working in a satisfactory manner scripts were created from these notebooks to allow for more fluidly running different tasks.

Following that began the tedious task of reconstructing the missing model.

A. Model Details

All of the models are built using a nested encoder decoder structure. (For readability that will written as Inner/Outer when referring to a model.) In this nested structure the outer encoder makes integer embeddings for each filler token. The outer models are all pre-trained for encoding and decoding filler tokens. These embedded tokens are then passed to the inner encoder decoder where sequence embeddings are made. Then the process is reversed with the inner decoder attempting to parse the sequence embeddings into token embeddings that the outer decoder can translate into the original sequence. All of the configurations are created with both coupled and separate versions of the inner encoder/decoder. This allows us to train with teacher forcing and

then turn it off and test without it. All the models use the same training and testing data. All the models load the transformer block classes, the embedding classes, and any functions that were previously duplicated from the same files.

Two different architectures were tested in four different inner/outer configurations. Graphical representations of the outer models can be found in Appendix A, and the same can be found for the inner models in Appendix B. Many of the key parameters used can be found in Table I. All of the models were made using Tensorflow/Keras (version 2.7.0) and Tensorflow addons (version 0.17.1-dev).

TABLE I: **Key Model and Training Parameters**[2]

Name	Value	Description
LSTM/LSTM		
Outer size	100	Dimension size of filler token embeddings
HIDDEN_SIZE	300	Dimension size of sequence embedding
α	0.001	Learning rate for Adam optimizer
EPOCHS	1600	Number of epochs
BATCH_SIZE	100	Batch size
Transformer/Transformer		
Outer size	300	Dimension size of filler token embeddings
Outer Num_heads	32	Number of attentions heads in outer transformer block
Inner Num_heads	4	Number of attention heads in inner transformer block
embed_dim	128	Embedding size for each token
Inner ff_dim	32	Hidden layer size in feed forward network in inner transformer
rate	0.1	Dropout rate for both inner and outer transformers
α	0.001	Learning rate for Adam optimizer
EPOCHS	250	Number of epochs
BATCH_SIZE	50	Batch size
LSTM/Transformer		
Outer size	300	Dimension size of filler token embeddings
HIDDEN_SIZE	300	Dimension size of sequence embedding
Num_heads	4	Number of attention heads in each transformer block
ff_dim	4	Hidden layer size in feed forward network in transformer
rate	0.1	Dropout rate for transformer
α	0.001	Learning rate for Nadam optimizer
EPOCHS	40	Number of epochs
BATCH_SIZE	25	Batch size
Transformer/LSTM		
Outer size	64 or 256	Dimension size of filler token embeddings
HIDDEN_SIZE	300	Dimension size of sequence embeddings
α	0.001	Learning rate for Adam optimizer
EPOCHS	1600	Number of epochs
BATCH_SIZE	100	Batch size

1) LSTM/LSTM: The first of the four configurations tested was the inner LSTM with an outer, pretrained LSTM model. In "Attention Is Not Enough" this model was tested with both one-hot and integer embeddings but no significant difference was found. Because of their results, this model uses integer embeddings. This configuration uses the Adam optimizer with a mean squared error (MSE) loss function for the embedding token layer and a binary corssentropy (BCE) loss function for the start/stop token layers. It was trained according to the parameters in Table I.

2) Transformer/Transformer: This model is an inner transformer encoder/decoder modeled separately and then coupled for training. It has outer transformer blocks that have been pre-trained. It uses the Adam optimizer with an MSE loss function for the embeddings and a BCE loss function for the start/stop tokens. It was trained according to the parameters in Table I.

3) LSTM/Transformer: This configuration loads pre-trained transformer models for the outer encoder and decoder. It is then composed of an inner LSTM encoder/decoder model. Notably different, this configuration uses the Nadam optimizer. It does still use MSE and BCE loss functions in the same way though. It was trained according to the parameters in Table I.

4) Transformer/LSTM: This was the configuration missing from the original codebase. It loads a pre-trained LSTM outer model first, and then creates a coupled and uncoupled version transformer encoder/decoder. This also uses the Adam optimizer with a mean squared error (MSE) loss function for the embedding token layer and a binary crossentropy (BCE) loss function for the start/stop token layers. It was trained according to the parameters in Table I.

IV. Testing and Results?

A. Testing

The models, both inner and outer, were built to have start/stop tokens input along with the data. This means that, in theory, the models could take sequences and filler tokens of various lengths. That theory wasn't tested, however, as the data used for this was a set length. Specifically, the filler tokens were five letters each and the sequences were composed of three roles. The outer models were both trained to 100% accuracy with a corpus of 1000 filler tokens, and then the training and testing data for the inner models was created from that corpus.

In the paper "Attention in Not Enough" four different architectures, several of which involved transformers, were tested against a working memory cognitive architecture. This also attempted to test extrapolation capabilities by abstracting a word's role in simple sentences. There were four different versions of this test increasing in difficulty of extrapolation. The first three tests were originally developed for testing indirection on a biologically plausible model. [5] The fourth was created by Jovanavich to test a working memory model. [6]

1) Standard Generalization (SG): The training set presents the model with every filler being used in every role, as well as every filler being used in a sequence together. For example, if the fillers are ”dog”, ”tooth”, and ”ball” and the roles are where they appear in a sequence then the training set could include ”dog tooth ball”, ”tooth ball dog”, and ”ball dog tooth”. The testing will have unique role-filler pairs. With our example a testing phrase could be ”tooth dog ball”.

2) Spurious Anticorrelation (SA): The training set includes all fillers being used in every role, but not all in the same sequence together. For example, if the fillers are ”dog”, ”tooth”, ”ball”, and ”pig” and their roles are where they appear in the sequence then the training set could include, ”dog tooth ball”, and ”tooth ball pig”. The testing will have unique role-filler pairs that have not been in the same sequence together. With our example a testing phrase could be ”dog ball pig” because ”dog” and ”pig” weren’t put together in the training set. Note that this test should have sets where ”dog”, ”tooth”, ”ball”, and ”pig” have a chance to be trained in every role.

3) Full Combinatorial (FC): The training set has fillers that are not used in every role. For example, if the fillers are ”dog”, ”tooth”, ”ball”, and ”pig” and their roles are where they appear in the sequence the training set could include ”pig tooth ball”, ”pig dog ball”, and ”pig tooth dog”. The testing set will have fillers used in roles that are not in the training set. With our example a testing phrase could be ”dog tooth ball”. In this ”dog” was tested in the first role but new filled that role in the training set.

4) Novel Filler (NF): The testing set includes fillers not in the training set at all. For example, some of the training sequences could be ”dog tooth ball” ”tooth ball dog”, and ”ball dog tooth”. Following this example a testing sequence could be ”pig dog tooth” as ”pig” is not in the training set.

B. Results

For optimal results every test should be run on every model a number of times and then averaged. This would parallel the original project as well as give a more accurate picture of how well the models are learning these tasks. [2] In this case, we have run each test on each model once and the results can be seen in Table II.

	LSTM/LSTM		LSTM/Transformer		Transformer/LSTM		Transformer/Transformer	
Test	Word Accuracy	Letter Accuracy	Word Accuracy	Letter Accuracy	Word Accuracy	Letter Accuracy	Word Accuracy	Letter Accuracy
SG	98.333	99.555	73.333	92.777	97.666	99.0	100.0	100.0
SA	98.666	99.666	80.0	92.777	99.666	99.944	100.0	100.0
FC	5.333	28.444	20.0	55.0	52.666	65.555	100.0	100.0
NF	35.666	53.5	50.0	67.777	52.333	58.055	59.999	72.277

TABLE II: Results from one pass of training and testing

V. Discussion

The results from one pass of testing are largely consistent with the results reported in ”Attention Is Not Enough.” The inner model weights are trained from fresh random weights each time they are tested though, so some variance is expected. This is why the original experiment was performed multiple times with the results being averaged. Table **??** shows a comparison of the previous study. [2]

	LSTM/LSTM		LSTM/Transformer		Transformer/LSTM		Transformer/Transformer	
Test	Word Accuracy	Letter Accuracy	Word Accuracy	Letter Accuracy	Word Accuracy	Letter Accuracy	Word Accuracy	Letter Accuracy
SG	-0.1003	-0.0561	-2.0000	0.4440	-0.6340	-0.0833	0.0000	0.0000
Deviations	1.0651	0.2612	6.5168	1.9387	1.0899	0.7430	0.0000	0.0000
SA	-0.7340	-0.1451	4.6670	0.5550	0.7327	0.3940	0.0000	0.0000
Deviations	0.4899	0.1614	6.1262	2.0951	1.0520	0.5238	0.0000	0.0000
FC	-8.6003	-12.5893	3.3340	-0.6111	-32.1673	-26.1061	0.0667	0.0222
Deviations	4.1333	7.0561	2.2222	5.8352	9.1388	6.1480	0.2108	0.0703
NFs	-18.2673	-11.6611	-4.0010	-3.2785	-3.0003	-37.3894	2.4660	0.8604
Deviations	4.4642	4.0325	4.0976	3.7263	3.8151	4.2712	6.9143	4.6052

TABLE III: Comparison of the results from ”Attention Is Not Enough” with the results from this study [2] The deviations were taken from ”Attention Is Not Enough”. The differences were calculated by subtracting the averaged results from the previous study from the one pass results from this one.

All of the different configurations performed within the expected deviation on the SG and SA tasks; the LSTM/Transformer and Transformer/Transformer configurations both performed as expected across the board. The LSTM/LSTM and the Transformer/LSTM configurations were both notably outside of the expected deviations however. It’s possible that these would still average out to be consistent with the previous study considering that this study only did one pass of testing. It’s also possible that there was an error in construction. Since the Transformer/LSTM configuration was based partially on the LSTM/LSTM

configuration, specifically the way the encodings were processed and the testing process were both based on the original LSTM/LSTM script. Either way, more investigation is required.

As you can see in Table II all of the configurations did excellently on the SG and SA tasks. This indicates that they have interpolatory properties. That said, the LSTM/Transformer configuration performed worse on those tasks than other configurations. This does indicate that the outer transformer encodings may be hard for the LSTM architecture to process. There could be other causes for this configuration to under perform which is worth looking into.

The LSTM/LSTM model performed quite badly on the FC and NF tasks. This was expected, and indicates that the tasks are hard enough to properly test for extrapolation. The two configurations with both LSTM and Transformer components did better but not excellently. The Transformer/Transformer model did much better than the others on the FC task, but still dropped down to the only 60%/72% word/letter accuracy which is not marginally better than the other models with transformer components. This indicates that the transformer does have better extrapolation potential than LSTM components but it's still not competitive with with WM models as indicated by "Attention Is Not Enough". [2]

A. Future Work

Because of the time constraints on this project, the testing performed was not as extensive as the previous project. To further verify that the current code base is consistent with the previous one the tests need to be performed multiple times and then averaged. In "Attention Is Not Enough" each test was run ten times on each model so we should do that at a minimum. [2] This is best done by creating an automated script that runs the models and compiles the data.

Other than the missing configuration script, the codebase from "Attention Is Not Enough" was also missing the scripts used to generate, train, and save the outer models. Since we had the saved models and weights these weren't imperative for testing. For the sake of prosperity, there is intention to recreate these.

1) ESPN: Emergent Symbol Binding Network (ESBN) is a newly proposed deep learning architecture intended to solve extrapolation tasks. The ESBN network attempted to demonstrate extrapolation by solving four different tasks that require the application of an abstract rule. In all four cases the testing involved symbols not seen during the training. The study tested these tasks in comparison to several different architectures including transformers along with some more traditional architectures. They also compared the performance of each architecture with how much of the training set was withheld. While the ESBN network had 100% test accuracy with as few as 2 training examples the other architectures, with the exception of recurrent neural nets, also performed extremely well given at least 50% of the training set. This indicates that either the extrapolatory potential of transformers and traditional deep learning techniques has been underestimated or the tests used on the ESPN network were not rigorous enough. [1]

We wish to explore the extrapolation potential of ESPN networks compared to transformers and cognitive architecture.With these results reinforcing the previous conclusion is would be safe to guess that the ESBN network was not tested to it's limit. In the future there are plans to create an implementation of the ESBN network and use similar testing to that done here in hopes of separating the capabilities of the different models.

APPENDIX A
GRAPHS OF THE OUTER MODELS

A. Outer LSTM Models

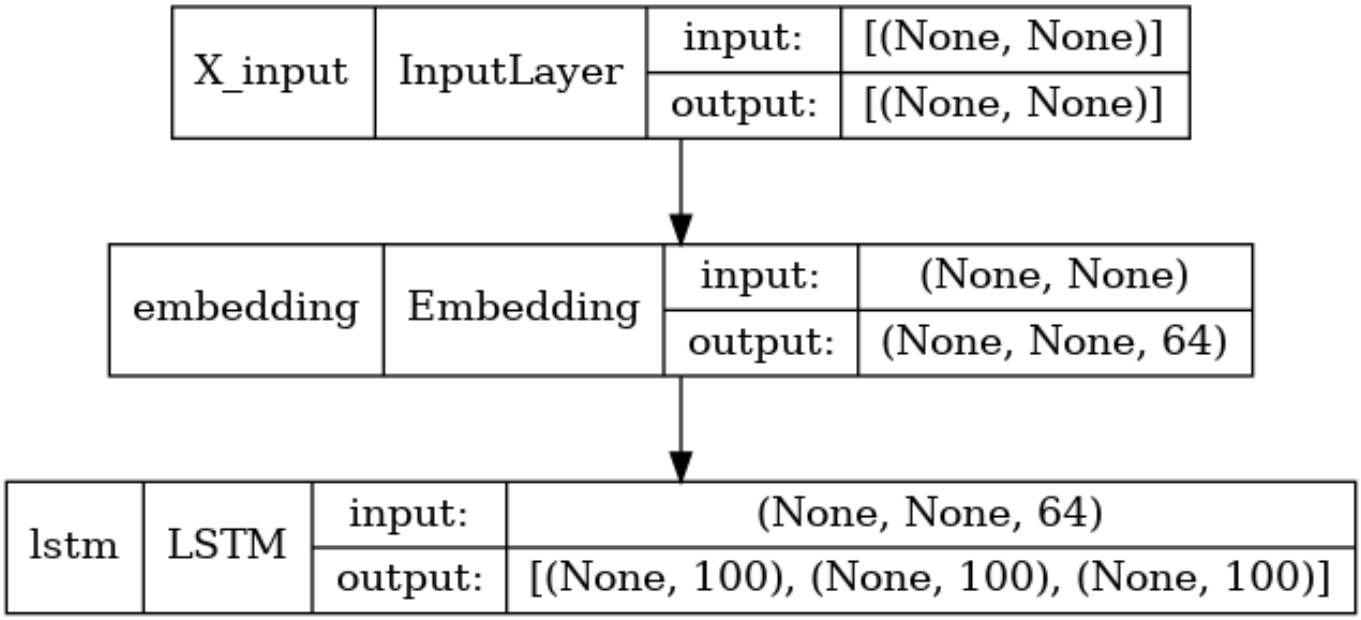

Fig. 2: Outer LSTM Encoder

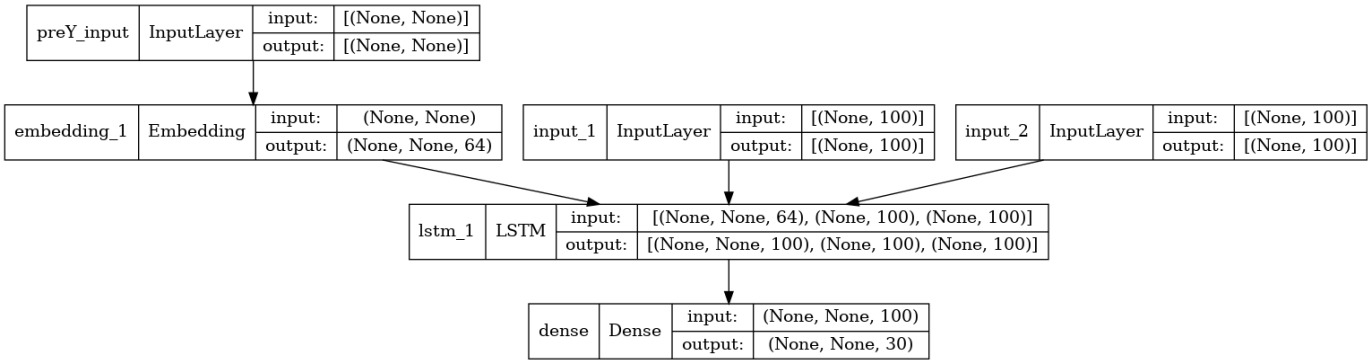

Fig. 3: Outer LSTM Decoder

B. Outer Transformer Models

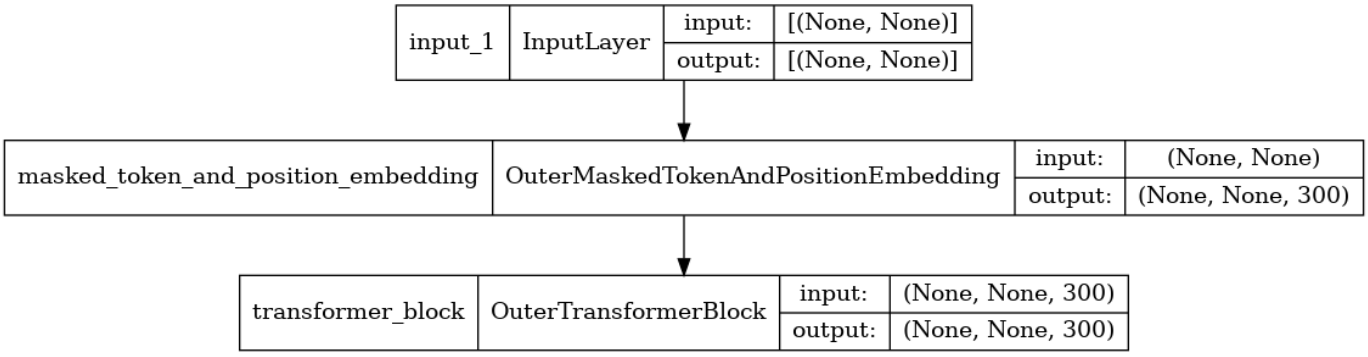

Fig. 4: Outer Transformer Encoder

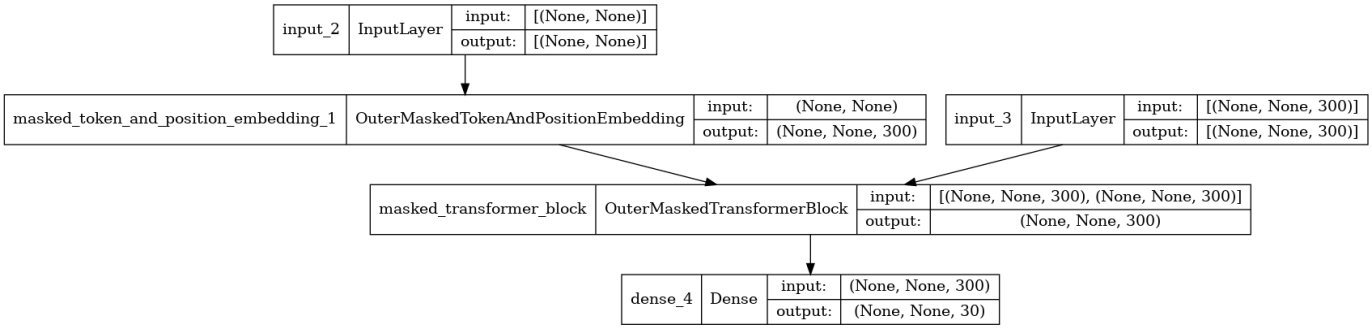

Fig. 5: Outer Transformer Decoder

APPENDIX B
GRAPHS OF THE INNER MODELS

A. LSTM/LSTM

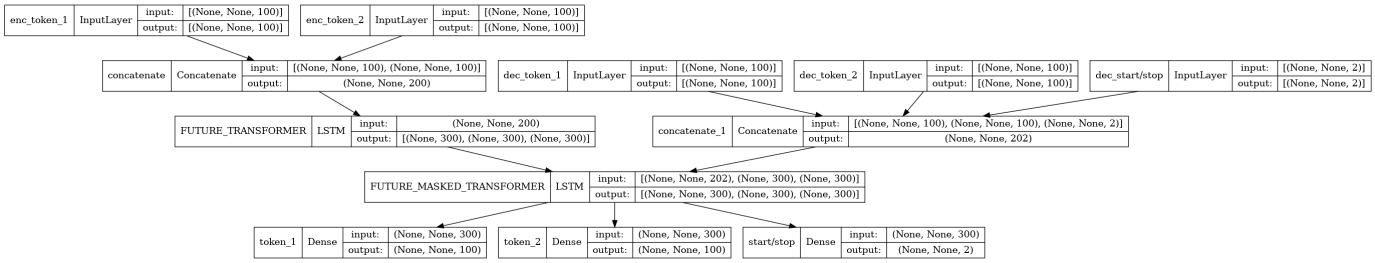

Fig. 6: The Coupled Inner LSTM model for the LSTM/LSTM configuration

B. Transformer/Transformer

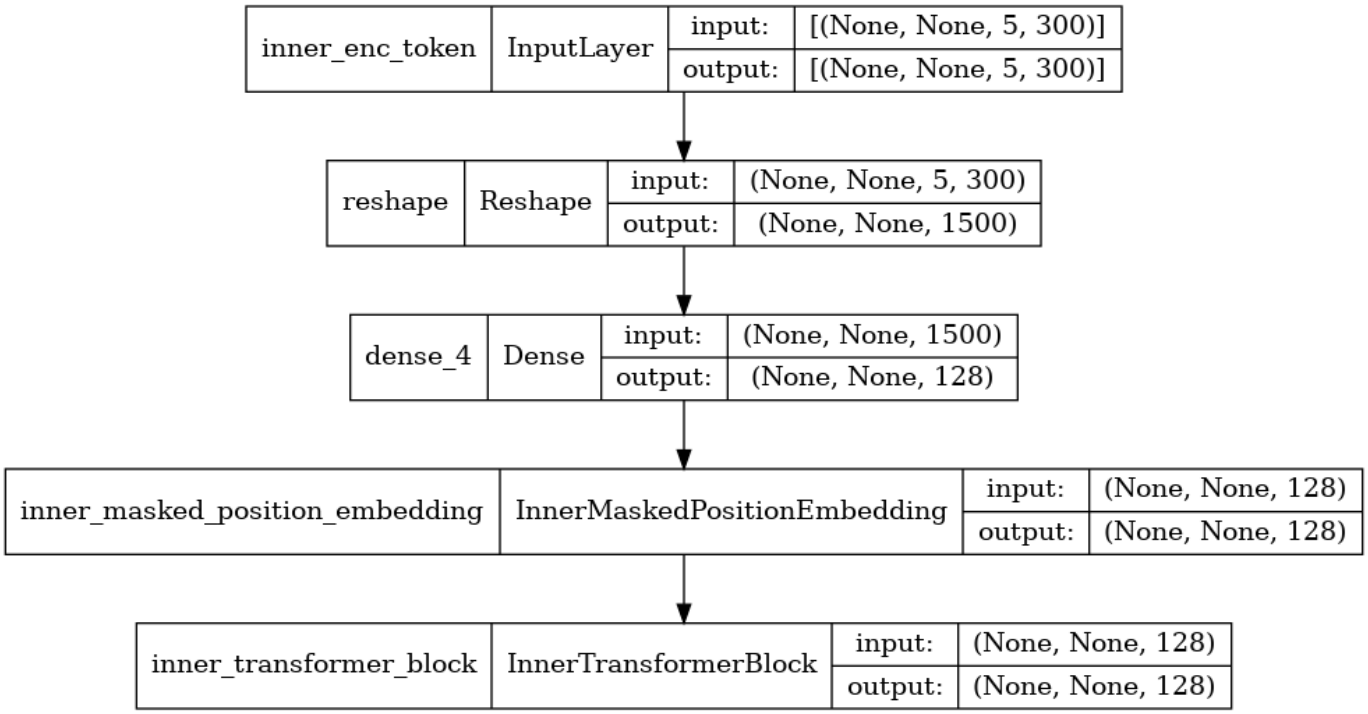

Fig. 7: Inner Encoder for the Transformer/Transformer configuration

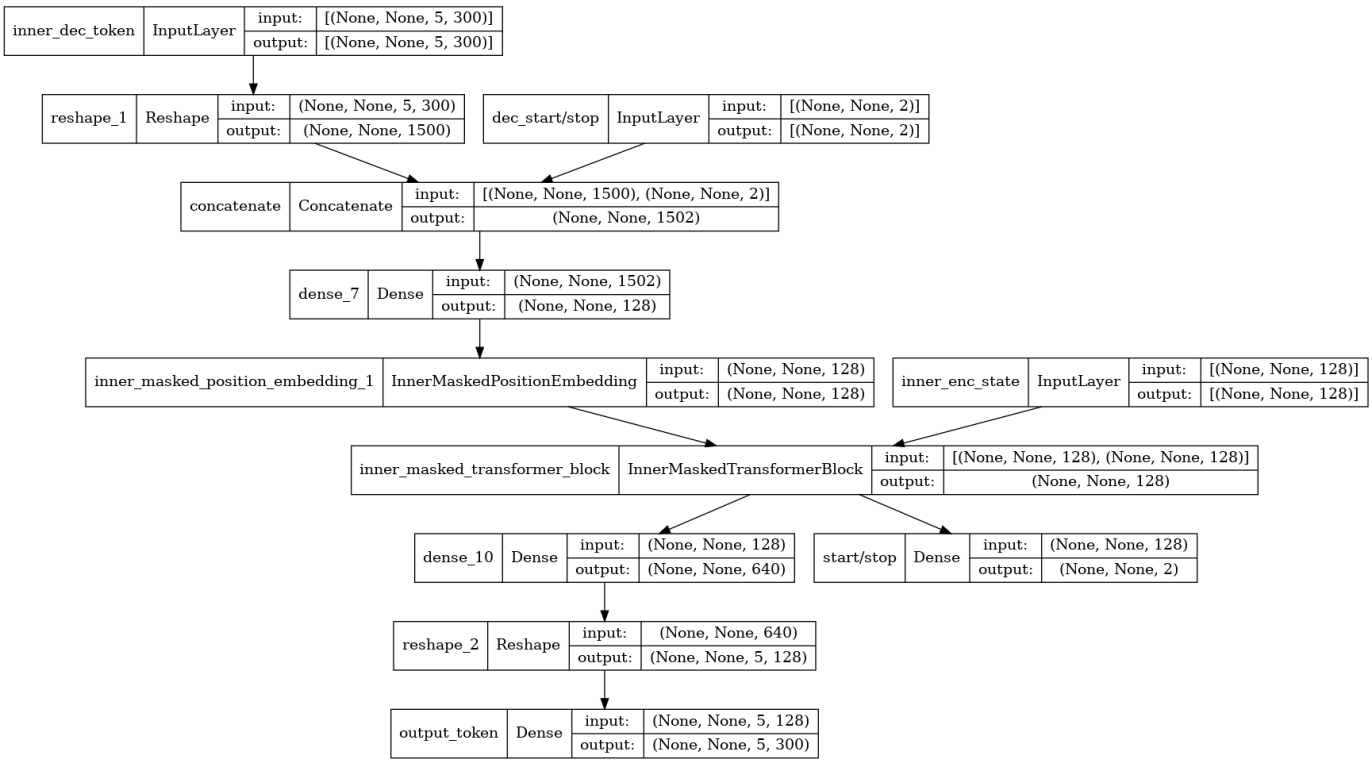

Fig. 8: Inner Decoder for the Transformer/Transformer configuration

C. LSTM/Transformer

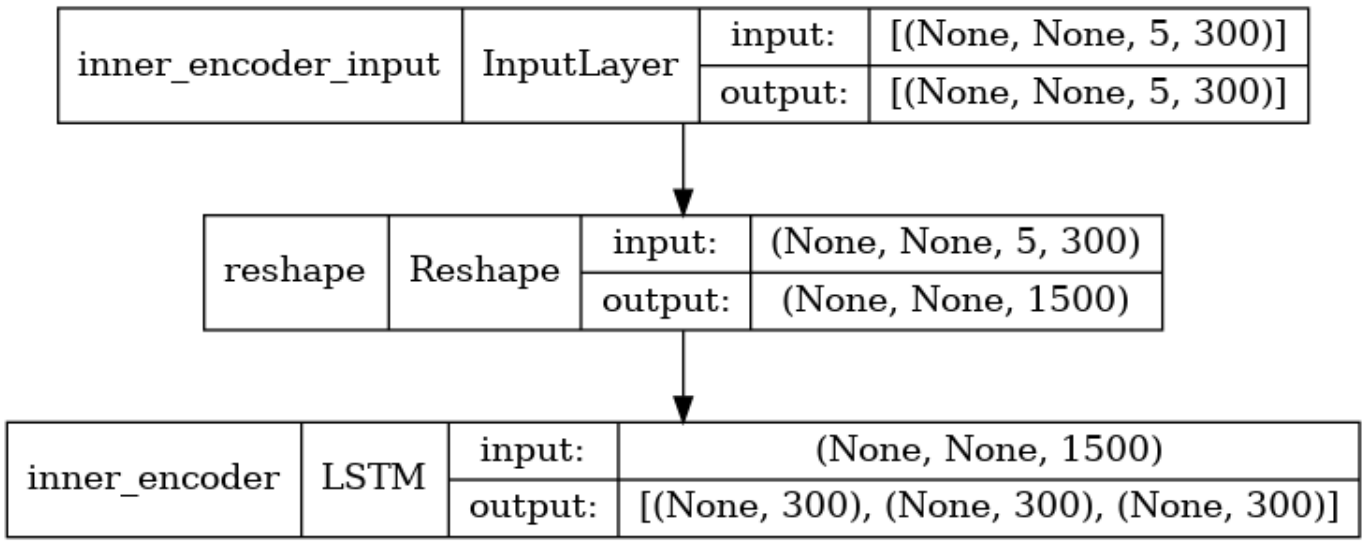

Fig. 9: Inner Encoder for the LSTM/Transformer configuration

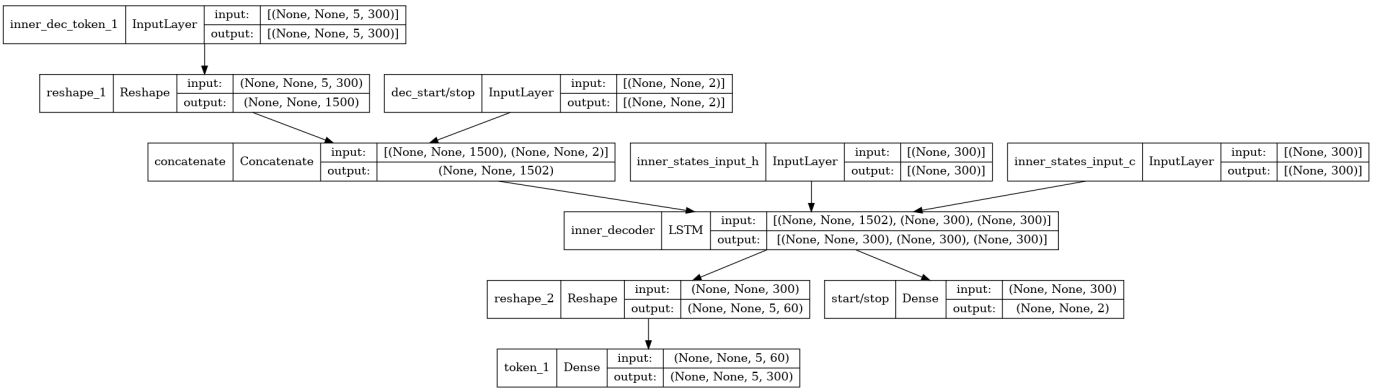

Fig. 10: Inner Decoder for the LSTM/Transformer configuration

D. Transformer/LSTM

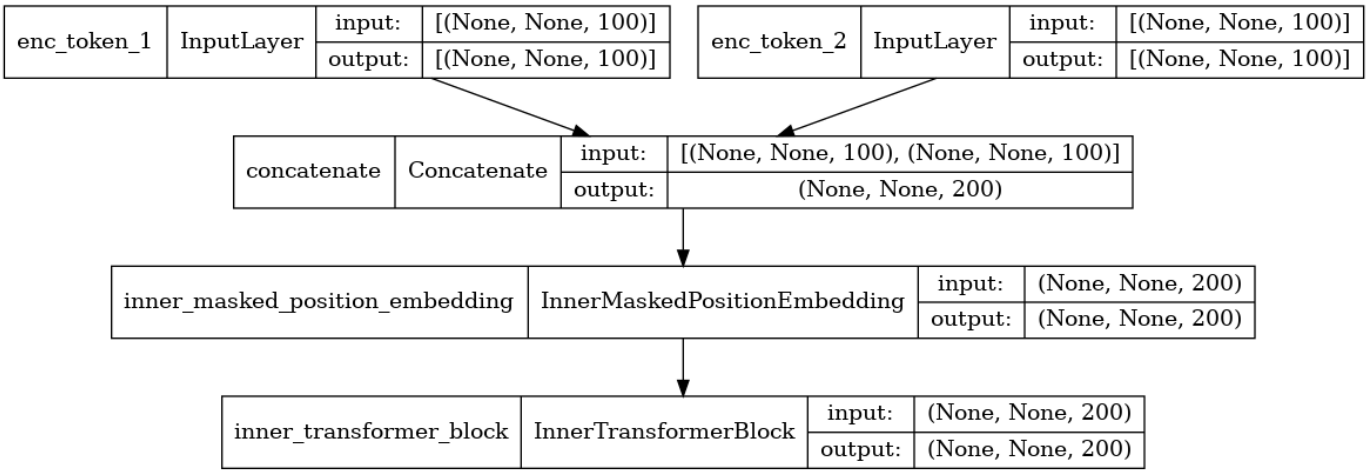

Fig. 11: Inner Encoder for the Transformer/LSTM configuration

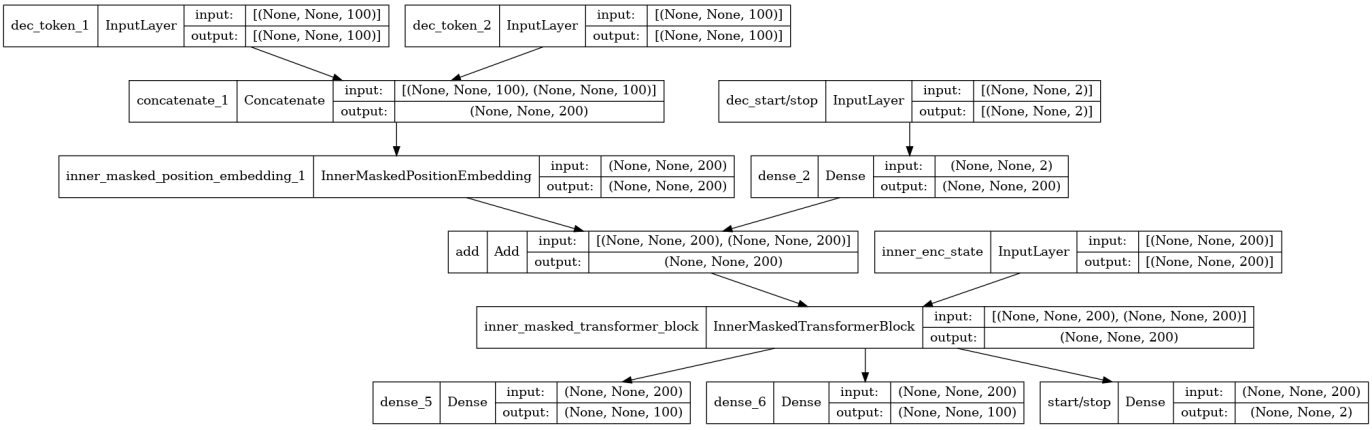

Fig. 12: Inner Decoder for the Transformer/Transformer configuration

REFERENCES

[1] Taylor W. Webb, Ishan Sinha, and Jonathan D. Cohen. “Emergent Symbols through Binding in External Memory”. In: (Mar. 2021). Number: arXiv:2012.14601 arXiv:2012.14601 [cs]. DOI: 10.48550/arXiv.2012.14601. URL: http://arxiv.org/abs/2012.14601.

[2] In: (). URL: https://www.cs.mtsu.edu/~jphillips/papers/MillerNaderiMullinaxPhillips-CogSci-2022-preprint.pdf.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention Is All You Need”. In: arXiv:1706.03762 (Dec. 2017). arXiv:1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762.

[4] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory”. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[5] Trenton Kriete, David C. Noelle, Jonathan D. Cohen, et al. “Indirection and symbol-like processing in the prefrontal cortex and basal ganglia”. en. In: *Proceedings of the National Academy of Sciences* 110.41 (Oct. 2013), pp. 16390–16395. DOI: 10.1073/pnas.1303547110. URL: https://pnas.org/doi/full/10.1073/pnas.1303547110.

[6] Michael P. Jovanovich. “Biologically Inspired Task Abstraction and Generalization Models of Working Memory”. en. PhD thesis. Middle Tennessee State University, Oct. 2017. URL: http://jewlscholar.mtsu.edu/xmlui/handle/mtsu/5561.