Understanding the significance and architecture of AlphaFold

David Li

February 2022

Abstract

This paper presents an analysis of parts of AlphaFold, an attention-based neural network which is capable of predicting protein structures to high accuracy. A list of general purpose features in the AlphaFold network, as well as plausible interpretations of these features, will also be presented. In conjunction with analysing architectural elements of AlphaFold, there will be discussion regarding AlphaFold's ability to learn and reason, and how this will impact machine learning and biological computation in the future.

Contents

1 The Protein Folding Problem

Proteins are large and complex molecules that form the basic building blocks of life. They are responsible for essential tasks such as forming antibodies and transporting and storing nutrients. Since a protein's three dimensional (3D) structure has a tangible impact on its function, an important problem in biology is the Protein Folding Problem: to predict the 3D shape of any protein given its contiguous chain of amino acids.

Each unique protein, which has a sequence of amino acids as its primary structure, gives rise to a corresponding three dimensional (3D) structure that the protein curls and folds up into, which consists of the protein's secondary, tertiary and quarternary structure. This process unlocks its biological functionality, and happens spontaneously in nature. The difficulty of solving the protein folding problem follows from the vast number, type and strength of various intramolecular forces that can have a significant effect on the protein structure.

Prior to computational efforts to solve this problem, scientists would have to undertake arduous, expensive and time-consuming methods, such as X-ray crystallography, to uncover a protein's 3D shape. However, AlphaFold is now able to produce highly accurate predictions proteins, many of which do not share any part of its primary structure with other known proteins. Thus, although AlphaFold cannot achieve 100 percent accurate predictions, some experts consider the protein folding problem to be solved sufficiently for experimental purpose.

2 Significance of AlphaFold

It is remarkable that there are very few explicit instructions in the AlphaFold network. Many human concepts in protein folding, such as alpha helices and inter-molecular forces, may be implicitly learnt and understood by AlphaFold, yet do not have a dedicated contribution to the loss function. This suggests that the architectural elements in AlphaFold can be repurposed to solve other problems. The network itself requires an amino acid chain input, and outputs the 3D atomic coordinates of the folded protein. By modifying databases and loss functions, a similar network could be expected to perform tasks that require prediction of 3D shapes from a contiguous input.

Moreover, there is evidence that AlphaFold possesses the ability to reason, which is defined as the process of comprehending and inferring in an orderly and rational way. There is a minimal number of handcrafted features and loss functions in the network that prescribe what to learn from the data and how to use its knowledge.[1] Instead, architectural features of the network allow it to understand general rules and patterns that are applicable in different situations, and then adapt and change its understanding to solve the task at hand with high accuracy.[2]

The lack of prescribed features also induces creativity in the network. Since there are no pre-specified concepts to learn, AlphaFold is not constrained by concepts deemed important to solve the protein folding problem *a priori*. Instead, there is full autonomy afforded to the network to

construct concepts that it itself deems sufficient to solve the problem.

Hence, AlphaFold's success stems from its ability to understand concepts, reason from them and create original knowledge. This builds upon the previous success achieved by DeepMind in narrow domains with man-made rules, such as Chess and Go. However, in solving the Protein Folding Problem, AlphaFold has allowed AI to make the jump to solving fundamental biological problems that could tangibly impact humanity in the years to come.

Therefore, AlphaFold's success is like a stepping-stone in the AI development journey, bridging the gap between narrow domains that demand strategy, and gaining brain-like capabilities to not only solve problems in nature, but identify them. This potential demonstrates that achieving intelligence more sophisticated than what we thought possible is now within arm's reach.

3 AlphaFold Architecture Overview

To preface this overview, this paper will not analyse an exhaustive list of features in AlphaFold. This overview is thus intended to give adequate context for the discussion of main features of the network. For more details on the AlphaFold architecture, please see [1] and [3]. Moreover, the notation used mirrors that in [3].

The AlphaFold network is composed of two towers. Upon input of the primary structure, multiple protein databases are queried to find similar evolutionary sequences from other species.

The first tower of the Evoformer refines the Multiple Sequence Alignment (MSA) representation. Due to the high computational costs of $N_{seq}^2 \times N_{res}$, AlphaFold utilises "clustering" to reduce the number of sequences embedded into the MSA without significant losses in accuracy. Clustering essentially groups all similar sequences by similarity into different clusters, with a representative chosen from each cluster to form the MSA.[3] These representatives are embedded with extra features that describe its cluster, allowing all sequences to influence the final prediction.[3]

The database search also embeds a pair representation, in which each entry a_{ij} represents the distance between residues i and j. It is thus a distogram, and refined in the second tower.

The first trunk of the network is the Evoformer, which consists of 48 blocks mostly made up of attention-based layers that alternate between refining the MSA and pair representations. Attention is a mechanism that allows the network to recognise which parts of the input is more important in solving the problem. Details are presented in [4] and are omitted here.

Within the Evoformer, the MSA and pair representation towers constantly **communicate** with each other to construct new sturctural hypotheses. This plays a critical role in the network in ensuring consistency and equal importance being placed on both evolutionary and geometric information.

In the tower that deals with the pair representation, the pair representations are transformed into graphs, and iteratively refined in the form of **triangular** updates.

Within both towers, the network utilises **gating** in almost every iterative layer to regulate memory flow throughout the network.

The second trunk of the network is the Stucture Module, in which the refined MSA and pair representations are mapped to 3D coordinates. To increase prediction accuracy, AlphaFold utilises **Adaptive Computation Time** (ACT) to extend the depth of the network for hard-to-predict proteins.

These four features: communication, triangles, gating and ACT consist of the most important parts of the AlphaFold network. Notably, they are also general purpose features in the sense that they are not specific to AlphaFold, and can be applied to other networks as well. This implies that their implementation in other networks should also increase performance. This ability to generalise across networks makes them the focus of discussion for the remainder of the paper.

4 Communication

There are two avenues of communication in the Evoformer: pair representations communicated to the MSA via biases, and MSA representations communicated to the pair representation via Outer Product Mean. Each of these two forms of communication occur once in each block, implying a total of $2 \times 48 = 96$ communications per cycle. This is in stark contrast to previous, similar architectures which facilitates communication only once in the network, upon completion of training.[1]

This continuous conversation between the MSA and pair representations allows for the iterative refinement of an initial structural hypothesis. For example, similar genetic sequences may reveal amino acids A and B have closely grouped residues. This information is then passed to the pair representation through the Outer Product Mean layer. Upon update, the pair representation may present a hypotheseis that, since A and B are close, C and D must also be close. This is passed back to the MSA through pair bias, where similar genetic sequences are consulted as to the validity of this hypothesis.

4.1 Pair Bias in Row Attention

Attention begins in line 5 of algorithm 1, where the pair bias is added to the standard argument $(\frac{\mathbf{q}_{si}^h \cdot \mathbf{k}_{sj}^h}{\sqrt{c}} + b_{ij}^h)$ of the softmax function, where the query and key vectors come from two different residues in the same sequence.The query-key dot product expresses that the relationship between residues i and j are close to the extent that the dot product is large and positive; the bias term conveys the prior hypothesis regarding the distance between residues i and j.[4]

def MSARowAttentionWithPairBias($\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, c = 32, N_{\text{head}} = 8$) :

 # Input projections

1: $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

2: $\mathbf{q}^h_{si}, \mathbf{k}^h_{si}, \mathbf{v}^h_{si} = \text{LinearNoBias}(\mathbf{m}_{si})$ $\quad \mathbf{q}^h_{si}, \mathbf{k}^h_{si}, \mathbf{v}^h_{si} \in \mathbb{R}^c,\ h \in \{1, \ldots, N_{\text{head}}\}$

3: $b^h_{ij} = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{z}_{ij}))$

4: $\mathbf{g}^h_{si} = \text{sigmoid}\left(\text{Linear}(\mathbf{m}_{si})\right)$ $\quad \mathbf{g}^h_{si} \in \mathbb{R}^c$

 # Attention

5: $a^h_{sij} = \text{softmax}_j\left(\frac{1}{\sqrt{c}}\, \mathbf{q}^{h\top}_{si} \mathbf{k}^h_{sj} + b^h_{ij}\right)$

6: $\mathbf{o}^h_{si} = \mathbf{g}^h_{si} \odot \sum_j a^h_{sij} \mathbf{v}^h_{sj}$

 # Output projection

7: $\tilde{\mathbf{m}}_{si} = \text{Linear}\left(\text{concat}_h(\mathbf{o}^h_{si})\right)$ $\quad \tilde{\mathbf{m}}_{si} \in \mathbb{R}^{c_m}$

8: **return** $\{\tilde{\mathbf{m}}_{si}\}$

Figure 1: Row Attention Pseudocode [3]

Moreover, additive bias conveys a sense of Boolean logic. Boolean addition can be interpreted as the logical expression "OR", which implies that attention here would either deal with the MSA belief relating residues i and j, or pair bias hypothesis relating i and j.[5] However, the explanation is somewhat lacking in rigour.

Hence, it is more insightful to speculate why the authors did not use multiplication to relate bias. Boolean multiplication has a much stronger sense of conjunction than addition with disjunction.[5] This reveals that the authors did not wish attention to focus on what both the MSA and pair representations had in common, since that information must already be learnt in previous blocks. Hence, additive bias provides new, but not necessarily fruitful, suggestions as to what structures to try next. This introduction of new hypothesis will then be tested throughout the rest of the block.

The right hand expression of the Hadamard product in line 6 ($\sum_j a^h_{sij} \mathbf{v}^h_{sj}$) concludes the standard attention architecture.

4.2 Outer Product Mean

The outer product layer combines all the information from the MSA and transmits it to the pair representations. In this layer, plausible structural hypotheses inferred from evolutionary history are communicated to the pair representations for further refinement. This is done via algorithm 2.

The use of the outer product bears semblance in interpretation to Tensor Product Representations (TPR) used in Natural Language Processing (NLP).[6, 7] Smolensky defined TPR as a representation of value bindings.[8] In NLP, the TPR is used to bind a word's role in a sentence to the word itself, termed a filler.[7] Although there is no direct translation of the NLP usage into AlphaFold, the Outer Product plays a similar role in binding the genetic information of any pair of residues.

In algorithm 2, the intermediate tensor formed by $\mathbf{a}_{si} \otimes \mathbf{b}_{sj}$ represents the binding of genetic information. $\mathbf{a}_{si}$ represents the genetic information

def OuterProductMean($\{\mathbf{m}_{si}\}, c = 32$) :

1: $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$
2: $\mathbf{a}_{si}, \mathbf{b}_{si} = \text{Linear}(\mathbf{m}_{si})$ $\quad \mathbf{a}_{si}, \mathbf{b}_{si} \in \mathbb{R}^{c}$
3: $\mathbf{o}_{ij} = \text{flatten}\left(\text{mean}_s(\mathbf{a}_{si} \otimes \mathbf{b}_{sj})\right)$ $\quad \mathbf{o}_{ij} \in \mathbb{R}^{c \cdot c}$
4: $\mathbf{z}_{ij} = \text{Linear}(\mathbf{o}_{ij})$ $\quad \mathbf{z}_{ij} \in \mathbb{R}^{c_z}$
5: **return** $\{\mathbf{z}_{ij}\}$

Figure 2: Outer Product Mean [3]

from sequence s regarding residue i; $\mathbf{b}_{sj}$ represents the genetic information from sequence s regarding residue j. From the formation of these intermediate tensors, all available genetic information is combined through this binding process via the outer product.

Next, computing the mean aggregates the bound representations, on which an affine transformation is then performed to extract hypotheses regarding the relative positions of residues i and j. This is the information that is passed onto the pair representation, which will then test the plausibility of these hypotheses with regard to other genetic information passed on regarding other pairs of residues, and geometric and physical constraints.

5 Triangles

When refining the pair representations, the Evoformer is designed to interpret the primary structure in terms of graph theory, with each residue as an entity and each entry in the pair representation as an edge. Each edge is then updated using a triangular multiplicative updates and triangular attention.

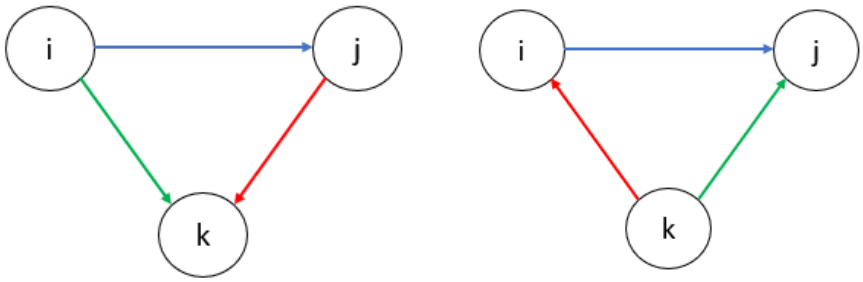
5.1 Multiplicative Updates

Triangular multiplicative updates are more symmetric and cheaper than triangular attention, but the combined use increases accuracy.[1] Whilst attention allows the Evoformer to distinguish between which residues have the most impact, the physical and geometric restraints, such as the triangular inequality on distance, mean that triangular multiplicative updates are imperative in ensuring no excess attention is paid to a particular residue subsequence. For example, whilst an alpha helix may be generated by residues in a close cluster, the Evoformer must not discard the importance of residues distant from the helix to avoid an infinitely repeating helix / atoms curling into each other etc.

Hence, it would be fruitful to conduct ablation studies to determine the impact on omitting these triangular multiplicative updates to determine its contribution to prediction accuracy.

5.2 Triangular Attention

Holistically, the attention around starting and ending nodes allow further geometric and chemical constraints to be learnt by the algorithm. For example, a group of residues at the beginning of the sequence may have partial positive charge, and further along the sequence there is another group of residues with partial negative charge. The incorporation of the third connecting edge informs the position of the residue with respect to these "global" attractions between partially charged residues or groups of residues.



(a) Starting Node

(b) Ending Node

Figure 3: Triangular Attention [3]

In figures 3a and 3b, the blue edge, ij, is being updated, with entity i being the starting node and entity j being the ending node. As such, ij produces the query vector through a linear transformation. The green edge produces the key and value vectors through a linear transformation; the red edge produces the bias term through a linear transformation.

Although it is not feasible to deduce the exact function and interpretation of the different edges in this triangular attention, one plausible explanation may be as follows.

Without the bias term, this attention mechanism builds a relationship between entities i and j using the most important interactions between entities i and k as identified by attention.[4] This iterates for all entities k, and computes a final updates for the edge ij.

However, this relationship is insufficient as chemical interactions is not limited to i and k, but i, j and k, and perhaps extends to more entities and in more complex ways. Hence, the bias edge at least allows for attention to consider the tri-entity relationship explicitly, although implicitly attention may be learning far more complex multi-entity relationships. Moreover, the additive bias here has similar implications to what was discussed prior regarding Boolean logic.

6 Gating

Gating is a concept that was developed to remedy Short-Term memory in RNN's. Over long input sequences, RNN's may fail to carry forward

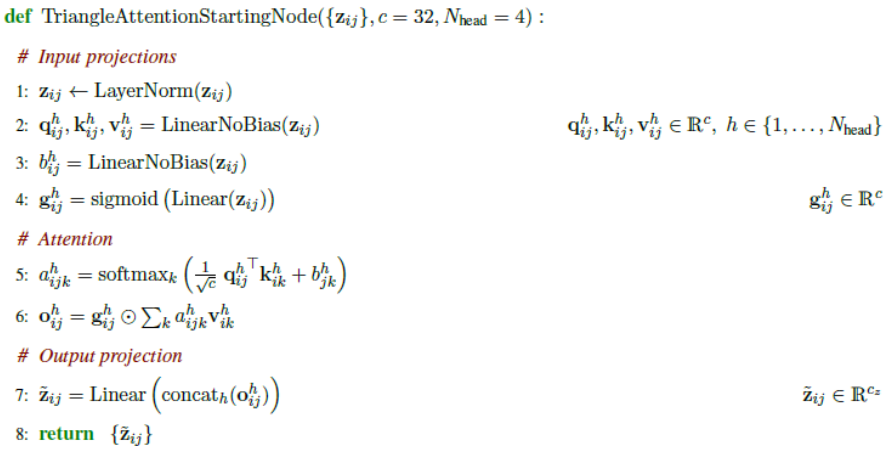


Figure 4: Triangular Attention [3]

information for the full length of input. RNN's are also susceptible to vanishing gradient problems in the early layers, thus not contributing to learning.

Long Short Term Memory (LSTM) is an architecture that remedies this problem.[9] It does this by introducing gates within cells of a vanilla RNN. These gates learn what information is relevant and thus should be remembered, and what should be discarded. This is achieved through sigmoid functions, which squeezes values between 0 and 1. Through elementwise multiplication, a gate with values near to 0 imply the corresponding information in the cell should be forgotten, and values near 1 should be remembered.

AlphaFold utilises gating in most layers of the Evoformer architecture. Although there is not an exact correspondence between gates in LSTM and in AlphaFold, there are similarities in interpretation, and thus terminology from LSTM's has been preserved.

6.1 Output Gates during Attention

Algorithms 1 and 4 are the archetype for two types of attention (MSA and triangular) in the Transformer. Both layers implement output gates upon completion of attention.

The element-wise multiplication by $\mathbf{g}_{si}^h$ in line 6 of both algorithms is strongly reminiscent of an output gate from LSTM's. This choice of architecture raises two points. Firstly, LSTM's usually operate with three gates: a forget gate, input gate and output gate.[9] The lack of forget gate implies that the authors do not necessarily want to sacrifice any prior information, especially given the prior discussion relating to clustering of the MSA. The input gate is also left out as that would imply abandoning information passed by the pair bias in algorithm 2, which would reduce information flow between MSA and pair representations.

The presence of the output gate thus implies some information should

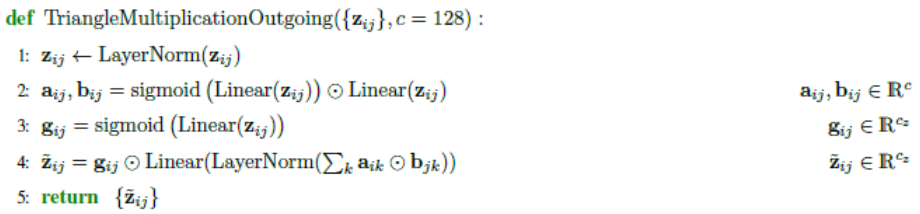


Figure 5: Outgoing Triangular Multiplicative Update [3]

be forgotten at this stage in the algorithm. This is done through the sigmoid function acting on the linear transform of the MSA in line 4, which compresses values between 0 and 1. This is then used to regulate what information should be passed on to the next layer for further refinement. Since attention has already been performed at this stage, some hypotheses communicated by the pair representation may already have been disproven, and the output gate regulates this such that it does not waste valuable computation and memory in future layers.

6.2 Gates in Multiplicative Updates

Algorithm 5 showcases the Triangular Multiplicative Update layer using outgoing edges. This layer utilises two gates: one in line 2 that strongly resembles a forget gate, and the other constructed in line 3 is identical to the output gates considered in the previous section.

As is standard, the forget gate consists of applying the sigmoid function to a linear transformation on the previous state. This has an explanation in the biochemistry of the Protein Folding Problem - there are certain residues with the potential to form stronger intramolecular bonds than others, and so they are more important in determining the protein structure. Hence, the interpretation of "forgetting" certain aspects is lost here. Instead, the gates here a more representative of [10]. The hadamard product in line 2 allows for the forget gate and the linear transform on the input to imposes dynamic gates upon each other. This may extract information more efficiently due to the gradient advantages, although one difference between here and [10] is additional parameters.

The output gate performs a similar role to the attention output gate in regulating plausible structural predictions to ensure computation is not wasted.

7 Adaptive Computation Time

Adaptive Computation Time (ACT), first introduced in [11], is a mechanism that allows Recurrent Neural Networks (RNN) to dynamically learn how many repetitions to "ponder" its input before outputting the next state. This idea is predicated on the idea that deeper networks have been shown to perform better, yet both computational efficiency and ease of learning mean that there is a cost-benefit decision in extending the depth

of networks. Hence, it would be optimal for the network to learn how deep it should extend itself through ACT.

Although the number of cycles in AlphaFold is predetermined as opposed to dynamically learnt, there is evidence that ACT is achieved through Invariant Point Attention (IPA) and recycling.

7.1 Invariant Point Attention

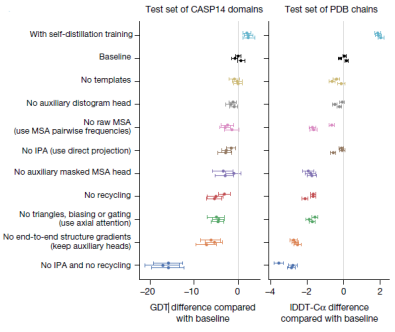
IPA is a new type of attention for dealing with 3D structures used in the structure module. It does so through acting on a set of frames which are parameterised in the structure module as Euclidean transforms.[3] This allows maximal extraction of information in each cycle of training as this type of attention is invariant under global rotations and translations. This reduces the number of models the network has to assess since IPA ensures the network understands rotating the global frame is inconsequential to prediction.

AlphaFold achieves IPA through the L2-Norm of a vector, which represents the Euclidean distance from the origin of a vector space to the vector co-ordinate and is itself invariant to translations and rotations.[3]

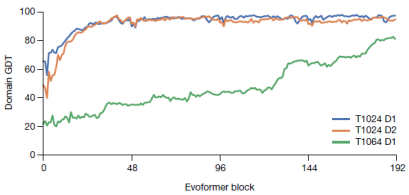
7.2 Recycling

Recycling is achieved through using the previous cycle's output as the new cycle's input.[3] This allows the network to further refine structures which is it less confident about, which is determined by an "intrinsic model accuracy estimate" named pLDDT. In a way, this is like increasing the computation time the network spends on difficult proteins without explicit instruction without suffering additional computational and memory cost throughout each cycle.

7.3 Empirical evidence of IPA and Recycling's effect on ACT



(a) Ablation Studies



(b) GDT over training

Figure 6: Interpreting Recycling [1]

Figure 6a shows the ablation studies conducted by DeepMind. Although neither recycling nor removing IPA independently affects accuracy greatly, the removal of both has a significant effect. This non-linearity suggests that these features in the architecture have interactive roles.

From the previous analysis of IPA and recycling, it is reasonable to think that the removal of both has a significant increase in the number of "wrong" models AF has to compute. Lack of recycling not only reduces the number of blocks when training, but also reduces the number of self-distilled examples; removal of IPA causes global transformations to generate numerous wrong hypotheses that take up valuable computational resources.

Figure 6b supports the hypothesis that recycling during training is a variant of ACT that directs additional training resources to low-confidence segments of the structure. The blue and orange lines represent proteins from the T1024 domain, which is significantly easier to fold as opposed to the T1064 domain, which is represented by the green line. Noting that 48 blocks consist of one layer, the T1024 domains reach its final level of accuracy within half a cycle, whereas the more complex T1064 domain require almost two and a half layers to achieve a significant increase in accuracy.

8 Acknowledgements

I would like to thank my supervisor Daniel Murfet for being so kind, patient and helpful in aiding me with research and editing. This paper would not have been written without your encouragement, and I have never learnt so much so quickly. I hope this contributes a brick to the proverbial brick wall of knowledge.

9 Conclusion

This paper discussed the importance of AlphaFold to both biology and machine learning communities, and its potential as a stepping stone to more sophisticated AI. Four main general purpose features were analysed, and plausible interpretations were presented. The fact that these features are detachable from the AlphaFold network and applicable to others (indeed three out of four features were introduced in other networks) further emphasise the point that AlphaFold's success is due to its potential to reason.

Hopefully AlphaFold's emergence and success will catalyse the invention of more advanced neural networks that can accomplish even more remarkable things.

References

[1] Jumper, J., Evans, R., Pritzel, A., et al., 2021, 'Highly accurate protein structure prediction with AlphaFold', *Nature*, vol. 596, pp. 583-589

[2] AlQuraishi, M., 2019, 'End-to-End differentiable Learning of Protein Structure', *Cell Systems*, vol. 8, no. 4, pp. 292-301.

[3] Supplementary Materials to 'Highly accurate protein structure prediction with AlphaFold'.

[4] Vaswani, A., et a., 2017, 'Attention is all you need', *Advances in Neural Information Processing Systems*, pp. 5998-6008.

[5] Boole, G., 1854, *An Investigation of the Laws of Thought*, Dover Publications, New York.

[6] Huang, Q., Smolensky, P., et al., 2018, Tensor Product Generation Networks for Deep NLP Modelling, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 1263-1273.

[7] Huang, Q., Deng, L. et al., 2019, Attentive Tensor Product Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp.1344-1351.

[8] Smolensky, P., 1990, 'Tensor product variable binding and the representation of symbolic structures in connectionist systems', *Artificial Intelligence*, vol. 46, pp. 159-216.

[9] Hochreiter, S., Schmidhuber, J. 1997, 'Long Short-term Memory', *Neural Computation*, vol. 9, no. 8, pp. 1735-1780.

[10] Wu, Y., Zhang, S., Zhang, Y., et al., 2016, 'On Multiplicative Integration with Recurrent Neural Networks', *30th International Conference on Neural Information Processing Systems*, no. 9, pp. 2864-2872.

[11] Graves, A., 2016, 'Adaptive Computation Time for Recurrent Neural Networks', *CoRR*, 1603.08983.