Incorporating Attention Mechanisms in RNN-based Encoder-Decoder Models

Patrick Hiemsch
pathi619@student.liu.se

Abstract

In this study, we investigate the performance and behavior of the Attention mechanism proposed by Bahdanau et al. (2016) in a machine translation task from German to English. We implement an RNN-based Encoder-Decoder network with Attention and visualize the attention weights generated during translation for selected input sequences. Our analysis reveals that the introduced Attention mechanism is able to learn the alignment of source and target words. It provides valuable insights into the model's behavior and the translation process, by highlighting the parts of the input that are most relevant for generating the corresponding output, contributing to the explainability of the model and it's results.

1 Introduction

Machine translation, which refers to the automated translation of textual input from one language to another, is an important field of research in natural language processing (NLP). Especially the shift in recent years from statistical machine translation (SMT) to neural machine translation (NMT), i.e. the utilization of deep learning techniques, revolutionized field of study. One of the key innovations has been the introduction of attention mechanisms, which allow the model to focus on certain parts of the input sequence when generating the output sequence. The attention mechanism has been shown to significantly improve the performance of machine translation models. Especially the transformer architecture innovated this field when Vaswani et al. (2017) published their paper on attention.

In this research project, we will investigate one of the earlier approaches to attention proposed by Bahdanau et al. (2016), which also contributed to the development of the transformer architecture. We implement an Encoder-Decoder neural network with attention for machine translation from German to English, based on the seminal work. We investigate and visualize the attention weights generated by the model for different input sentences to gain a better understanding of how the attention mechanism works in practice.

2 Theory

In the field of NMT, especially before the rise of the Transformer architecture, a widely used method was the training of an Encoder-Decoder architecture as described in Sutskever et al. (2014). In this class of models, both, the encoder and decoder networks are based on Recurrent Neural Networks (RNN), using Gated Recurrent Units (GRU) or Long Short-Term Memory (LSTM) specifically.

The encoder network is unrolled over the source sentence, computing a single summary hidden state, which is then fed to the decoder. The decoder takes this vector representation of the source sentence together with the start of sentence `<SOS>` token, to produce the next word in the target sequence. This process terminates when the end of sentence token `<EOS>` is produced (or when the length exceeds a certain maximum threshold).

As Bahdanau et al. (2016) describe in their paper, a problem that can arise in the context of these encoder-decoder architectures is that the whole source sentence needs to be compressed into a single vector. It is known, that unidirectional RNNs suffer from a recency bias. This refers to the fact that more recently processed tokens in the sequence will be represented stronger in the final hidden state, compared to the ones closer to the beginning of the sequence. This is especially true for longer sequences, which according to Cho et al. (2014) is the reason that these approaches have difficulties translating long sentences. This effect can be mitigated by using a bi-directional architecture and concatenating the last hidden states of both directions, but it still remains a challenge.

Bahdanau et al. (2016) propose a method that frees the encoder network from the need to compress the whole source sentence into one single vector. They introduce the concept of attention. This proposed part of the network takes all hidden states produced in the encoding phase, after unrolling the network over the source sentence, and computes a weighted fixed-length summary vector c_i. The weights of this attention layer are trainable parameters, which can be learned as usual by backpropagation.

To compute c_i as shown in Equation 1, all hidden states from the encoder step from each of the source sentence's tokens are summed together according to the computed attention weights α_{ij}, which are normalized to sum up to one. So c_i is essentially a weighted mixture of every hidden state of the source sentence. To get the attention weights α_{ij}, each hidden state h_j of the corresponding source token at position j is fed to an alignment model a, together with the latest hidden state of the decoder network s_{i-1}. See Equation 2 for details. This hidden state is yielded by the decoder after producing the latest prediction, so s_{i-1} is the hidden state that followed decoding word $i-1$ in the generated target sentence. In the first step this will be the last hidden state of the encoder network. The alignment model computes the energy e_{ij}, which scores how important the token at position j is for predicting the next word in the sequence. To get the attention weights from e_{ij}, each of them will be normalized through a Softmax-Layer, as showcased in Equation 3. This ensures that the α_{ij} values form a probability distribution over all steps in the source sentence. Important to note is also that the energy scores for padding tokens are manually set to a large negative number (in our implementation `-1e10`) which will lead to a (almost) zero attention score after applying Softmax. The alignment model can be modeled as a feedforward neural network itself.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{1}$$

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \tag{2}$$

$$e_{ij} = a\left(s_{i-1}, h_j\right) \tag{3}$$

Some more details of the exact Encoder-Decoder architecture used in the experiments will be mentioned later when describing the method. For further details on the theoretical aspects of RNNs and attention, have a look at Bahdanau et al. (2016) and Sutskever et al. (2014).

3 Data

In their paper, Bahdanau et al. (2016) use a bilingual text corpus from ACL WMT '14[1], which contains English-French sentence pairs. The total number of words included in their combined corpus is 348M.

For this project, we work with a dataset that originates from the same source, ACL WMT '14, but is targeted towards the translation from German to English and vice versa. It includes the proceedings of the European Parliament from 1996 to 2011. The dataset can be downloaded for free from the ACL WMT website[2].

Without preprocessing each corpus contains roughly 44.6M words in 1,920,209 sentences. Since the scope of the project is more on exploring the network architecture and investigating the effect of attention, the preprocessing is kept simple: Both text sets are combined into one big dataset. Then all sentence pairs are removed which contain none, more than 30 or less than 4 words. This leads to a full corpus of 1,084,301 sentences.

In a final step, the full corpus is randomly split into training, validation and test set with ratio 8:1:1.

4 Method

As described in the theory part, the network architecture used for the experiments in this project is closely related to Bahdanau et al. (2016). The implemented encoder network is a one-layer bidirectional GRU based architecture. Its forward pass returns the hidden state for each token in the source sentence - a stacked vector of both directions - and a summary vector of the source sentence, used as the initial hidden state for the decoder. This summary represents the output of a linear layer that maps the stacked vector of the last hidden state in each direction to a vector that matches the decoder's hidden dimension. Additionally, it also returns a padding mask. This mask has the same dimensions as the computed energy by the attention layer, so `(batch_size, padded_seq_len)`. The padded sequence length refers to the length of each sequence in a data mini-batch, which needs

[1] `https://www.statmt.org/wmt14/translation-task.html`

[2] `https://www.statmt.org/wmt13/training-parallel-europarl-v7.tgz`

to be the same for one batch in order for the computational graph to be built and executed. Some of the sequences will be padded with <pad> tokens in the end of it to match this length. It is crucial that padding is done in the end, since we pack[3] the sequences when we feed them to the different layers, which expects back-padded representations. The mentioned mask will then be used to set the energy scores e_{ij} manually to a large negative number to eliminate their contribution to c_i. The attention layer takes the mask, s_{i-1} and all encoder hidden states h_j as arguments and yields the hidden state c_i after applying attention. The alignment model used is a feedforward network with one hidden layer of equal size as the decoder hidden dimension. The decoder is similar in structure to the encoder, a one-layer GRU but with only one direction. It yields the next word logits and its current hidden state s_i.

For training, we use an adapted version of teacher forcing. In general, teacher forcing, as originally proposed by Williams and Zipser (1989), refers to consistently providing the decoder with the correct preceding token y_{i-1}, instead of the predicted counterpart $\hat{y}_{i-1}$. In our method, we adopt a probabilistic variation of teacher forcing. For each subsequent step in the target sequence, a probabilistic decision is made on whether to use the actual label, y_{i-1}, or the model's previous prediction, $\hat{y}_{i-1}$. This is a simplified version of the method described in Bengio et al. (2015), which lets the teacher forcing probability decay over time as the networks learning process progresses.

Regarding translation, Bahdanau et al. (2016) incorporate beam search in their experiments which is a heuristic approach to maximize the joint probability of the target sentence token. This is done by considering potential future steps - depending on the beam size - and comparing the different joint probabilities

$$p(\mathbf{y}) = \prod_{t=1}^{T} p\left(y_t \mid \left\{y_1, \cdots, y_{t-1}\right\}, c\right).$$

For simplicity, in our implementation the system will just act greedily, selecting the token with the highest probability at each step in the sequence.

In the experimental phase, we initially carry out a hyperparameter search, primarily focusing on critical parameters like the learning rate, batch size, and the ratio of teacher forcing. During each experiment, we monitor the validation loss to detect any signs of overfitting, preserving the optimal model weights based on the performance on the validation data. As the time horizon is limited and the focus of the project is to understand, implement and Experiment with the attention layer as part of a RNN-based Encoder-Decoder architecture, the hyperparameter search is kept limited.

In order to assess whether the integration of the attention mechanism has truly enhanced the performance, we utilize the optimal model settings derived from the hyperparameter search to train an alternate model with fixed uniform attention weights. This way, the attention layer is no longer learnable and each input token is weighted equally, allowing for an isolated asessment of the attention mechanism's effects on performance. To obtain an unbiased estimate of both models' performance, we subsequently evaluate them on the unseen test set.

To achieve a comprehensive understanding of the translation capabilities of both systems, we also measure the BLEU score based on the test set. It was first proposed by Papineni et al. (2002) and compares the generated text with one or more reference texts, which helps to ensure that the generated text is semantically and syntactically correct and meaningful.

To build the vocabularies for the source, German, and the target, English, we use the `torchtext` library. It is important to note in this context that especially the objects `Field` and `BucketIterator` are used, which are not longer part of the newest package version. We use version `0.6.0`. For both vocabularies, the vocabulary size is set to 8,000. This choice seems reasonable when looking at the results from Gowda and May (2020). They argue that for medium data sizes under 1.3 million sentences, a vocabulary size of 8,000 is reasonable.

The influence of the padding tokens on gradient computations and attention weights is eliminated in our implementation. This is achieved by using packed sequences to pass to the GRU-cell in the encoder model, by specifying to ignore the padding index in the target batch when computing the Cross-Entropy loss and by implicitly setting attention weights to 0 for all padding tokens.

[3] `https://pytorch.org/docs/stable/generated/torch.nn.utils.rnn.pack_padded_sequence.html`

5 Results

For our experiments we use the following shared hyperparameters: The vocabulary size is fixed to 8,000, the embedding dimension is set to 256 for both languages, both encoder and decoder have only one GRU layer, no dropout is used and the hidden dimensions are fixed to 512.

For Experiment 1 the learning rate is set to 1e-4, batch size is 128 and teacher forcing ratio equals 0.8.

For Experiment 2 the learning rate is set to 5e-4, batch size is 256 and teacher forcing ratio equals 0.8.

For Experiment 3 the learning rate is set to 5e-4, batch size is 128 and teacher forcing ratio equals 0.5.

For Experiment 4 the learning rate is set to 1e-4, batch size is 128 and teacher forcing ratio equals 0.5.

We evaluate each of the models based on their training and validation Cross-Entropy loss. The evaluation procedure is performed after reaching 25% epoch size, so four measurements per epoch.

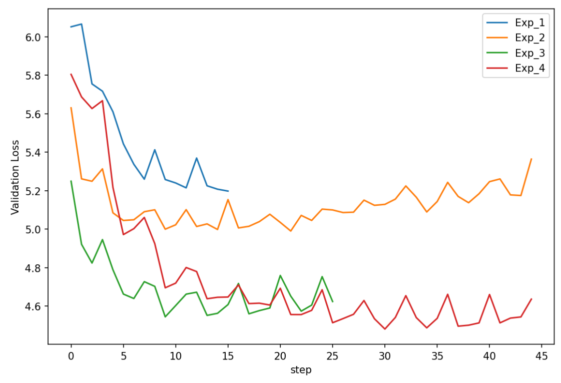

Figure 1: Validation loss curves for hyperparameter search

Figure 1 illustrates the validation loss curves for the conducted hyperparameter experiments. From the plots we can see that all models seem rather unstable with periodical changes in the loss. In general we can see that the models using a teacher forcing ratio of 0.5 show superior results on the validation data. These models, due to less reliance on teacher forcing, are trained in a manner that more closely mirrors the conditions they will face during evaluation, where teacher forcing is set to 0. This approach enhances learning. However, as previously noted, there is significant potential for improvement, possibly through a teacher forcing schedule that gradually diminishes over time. For models trained for more epochs, there is a tendency to overfit the training data after about six epochs, indicated by a steady increase in validation loss. This behavior is especially pronounced in Experiment 2.

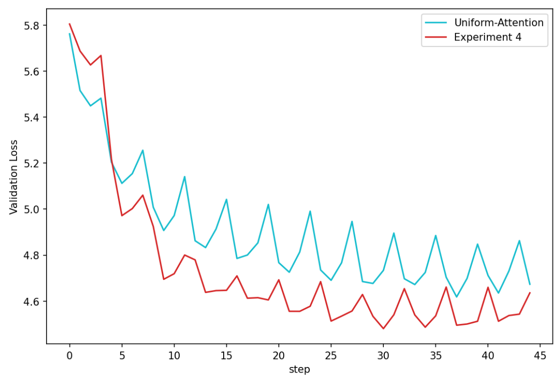

Figure 2: Validation loss for Uniform-Attention and Experiment 4

In summary, Experiment 4 delivered the most promising results based on the validation loss. Following the outlined methodology, these model settings are adopted to train the uniform attention model, that will be used for comparison.

The corresponding validation plots, along with the curve for Experiment 4, are showcased in Figure 2. It becomes apparent that the model utilizing uniform attention converges towards a higher loss than the one from Experiment 4, where the attention layer was learned during training.

However, since the validation set was used for hyperparameter search, there might be some bias in this assessment. To account for this, we also measure the Cross-Entropy loss on the unseen test set. The model with uniform attention registered a test loss of 4.61, whereas Experiment 4 yielded a slightly lower loss of 4.48. This suggests that learning attention weights, rather than assigning equal weights to all steps in the input sequence, can indeed bolster model performance.

This is also reflected in the BLEU scores based on the test set: the model with uniform attention achieves a BLEU score of 12.09, whereas the model from Experiment 4 shows a considerably higher score of 17.75. However, it is crucial to note that both scores suggest a rather limited capability for accurate translations, as scores below 20 on the BLEU scale are generally considered low and can be classified as "Hard to get the gist" according to

Google Cloud Translation documentation[4].

Example translations

In the following a few exemplary translations produced by the model from Experiment 4 are showcased. These instances are selected from the test dataset and represent examples that demonstrated particularly high-quality results based on their sentence BLEU score.

Ger: "Dies sind alles wichtige Bereiche, die wir in unserem eigenen Interesse weiterentwickeln müssen."
Gold: "These are all important areas which we need to develop in our own interests."
Model4: "These are all areas that we need to develop in our own interests."

Ger: "Natürlich sind wir mit der gegenwärtigen Situation in China nicht zufrieden."
Gold: "Of course we are not satisfied with the current situation in China."
Model4: "Of course, we are not satisfied with the current situation in China."

Ger: "Wir können die gegenwärtige Situation in Afghanistan nicht hinnehmen."
Gold: "We cannot accept the current situation in Afghanistan."
Model4: "We can not accept the current situation in afghanistan."

Attention visualization

Figure 3 displays the attention weights and alignment for the second example sentence discussed above. The sentence structure in German differs slightly from the English translation, a nuance that is captured by the attention weights. For example, the English verb "satisfied" is a direct translation of the German word "zufrieden", but it appears earlier in the sentence. It is evident from the plot that the highest weight assigned when translating to "satisfied" was placed on this exact German word, which seems appropriate given the context.

This pattern is similarly observed in Figure 4, which visualizes the attention scores for the third example previously discussed. Even though, again, the word order differs between the two languages, the model has learned to place the most substantial attention on the German word "hinnehmen" when translating to "accept".

[4]https://cloud.google.com/translate/automl/docs/evaluate

It is important to note that, in general, both examples still somewhat reflect a uniform attention structure - a diagonal pattern. However, this is reasonable considering the word order in both sentences remains largely similar, with the exception of the verb placement. As we have observed, the model has learned to adjust its attention to the appropriate word in the source language for these particular instances.

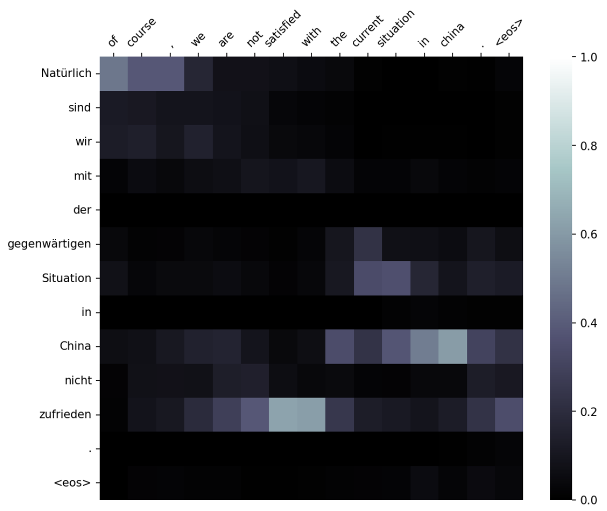

Figure 3: Attention weights for the second example

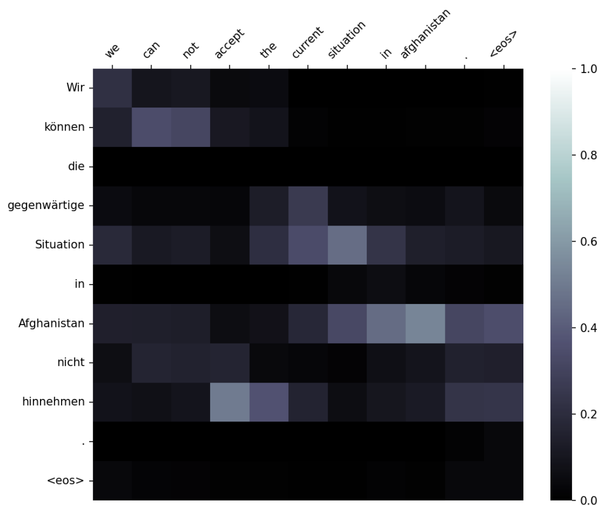

Figure 4: Attention weights for the third example

6 Discussion

As we can see, for the showcased examples from the test set, the model learned good translations accompanied by reasonable attention weights. . However, the comprehensive translation proficiency of the model, as indicated by the BLEU score, still falls short when assessed across the complete test dataset. In the following, we want to discuss some of the many factors that could influence the results we have seen:

Since the time horizon was limited and a lot of effort was spent to implement the pipelines, there was not enough time to explore the vast hyperparameter space in a sufficient way. More experiments with a more sophisticated hyperparameter tuning pipeline could have been beneficial. The corpus of the dataset with over one million sentences is fairly big given the resources of our project. Even though GPU training was employed, the training times and memory requirements of the models are high. In their paper Bahdanau et al. (2016) trained all of their models for over 100 hours. Their best performing model RNNsearch-50 was even trained for over 10 subsequent days (252 hours) on a bigger GPU. As mentioned above, the use of Beam Search could also improve the quality of the translations by heuristically maximizing the joint probability of the translation output words. This would be interesting to include in a further stage of this project.

The hidden state size also holds potential for further investigation. The models in our experiments were mostly initialized with a hidden state size of 512, which is already fairly big. However, the training and validation loss show a lot of periodical spikes and seem rather unstable. This could be due to the fact that the training loader yields batches of sentences of similar lengths to decrease padding and increase computational efficiency. This could be one of the reasons for the seen behaviour of the losses. For example, the model adjusts parameters on shorter sentences and then tries to adapt them based on the longer batches afterwards. However, if the model is not big enough to be able to capture the long term relationships, which is especially influenced by the hidden state dimensions, this could lead to undesirable gradient changes, which are then again changed by the smaller sentences, then by the longer ones again, etc. Therefore, a more detailed analysis of the hidden dimension sizes, especially larger ones, could be beneficial for the dataset at hand with a high average sentence length of roughly 23.5.

To mitigate the mentioned instabilities, another option could be to thoroughly monitor gradients. For instance the euclidean norm of the gradients could be observed over time to help identify and eliminate potential gradient instabilities such as exploding or vanishing gradients.

Additionally, experimenting with a learning rate schedule that progressively reduces the learning rate over time could be beneficial. This could allow the model to fine-tune weight adjustments more precisely, particularly towards the end of the training process.

Furthermore, Bahdanau et al. (2016) use a Maxout layer, as proposed by Goodfellow et al. (2013), before the computation of the Softmax target token probabilities. In the model used for this work the target output layer is just a linear layer without hidden layer which maps the stacked vector of decoder hidden state s_{i-1}, the attention encoder hidden state c_i, and the most recent embedded target token (depending on teacher forcing y_{i-1} or $\hat{y}_{i-1}$) to the target vocabulary size. This is also a factor to keep in mind for an extension of this project.

The same is true for dropout regularization (Srivastava et al., 2014), which could decrease possible overfitting and stabilize training and generalization performance. This is especially true for large models with high dimensional embedding and hidden dimensions.

The embedding layer for our model was kept at 256 for all runs. However, Bahdanau et al. (2016) map each word to a 620 dimensional vector, which could also help to learn better representations of words. This could also represent an opportunity for improvement of the architecture.

The vocabulary size on the other hand was kept fixed at 8,000 to reduce model parameters. The paper uses 30,000. Even though we presented a reason for the choice of vocabulary size, this is a parameter that should be investigated further.

All the mentioned factors above refer to model architecture and the increase of parameter size. To connect this to our main research interest for the project, the attention mechanism, all these changes present an opportunity for the model to learn more efficiently and more complex long-term relationships in the different vector spaces. This is of course in particular true for the attention layer which relies on good representations of word contexts in the hidden dimensions to be able to align words in a meaningful way.

Something that could be investigated as well is using pretrained embeddings to initialize the source and target embedding layers and either freeze them for training or fine-tune them based on the task at hand. This offers another opportunity for more stable training right from the start.

One more bias we want to point out is the fact that the used dataset contains transcripts of the european parlament, which is biased towards certain topics and a certain style of language. For the training of a more versatile model, and particularly for improving word alignment through attention, it would be advantageous to use a corpus with greater diversity. During the evaluation process, it was observed that certain phrases like "the vote will take place on" are overrepresented in the dataset. Excluding these overly repetitive sentences could potentially stimulate a more diverse learning process, thereby enhancing the model's generalization capabilities.

7 Conclusion

As the experiments have shown, the attention mechanism by Bahdanau et al. (2016) can be an interesting extension to the usual Encoder-Decoder RNN-based architecture. It can help the machine translation network yield better performance since it alleviates the burden of the encoder network to encode the whole source sentence in one single vector representation. Although the performance enhancements were modest, the comparison with the model using uniform attention validated the benefits of this approach. By learning to translate and to align at the same time, it also boosts explainability of the translation model since attention weights are a clear indicator for what source sentence tokens influence the networks prediction of the particular output word the most. All in all it is a promising approach but as discussed, there is still great potential for improvement.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural Machine Translation by Jointly Learning to Align and Translate.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout Networks.

Thamme Gowda and Jonathan May. 2020. Finding the Optimal Vocabulary Size for Neural Machine Translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ronald J. Williams and David Zipser. 1989. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280.