Attention in Neural Networks - 1. Introduction to attention mechanism

01 Jan 2020 | Attention mechanism Deep learning Pytorch

> Updated 11/15/2020: Visual Transformer

Attention Mechanism in Neural Networks - 1. Introduction

Attention is arguably one of the most powerful concepts in the deep learning field nowadays. It is based on a common-sensical intuition that we **"attend to"** a certain part when processing a large amount of information.

[Photo by Romain Vignes on Unsplash]

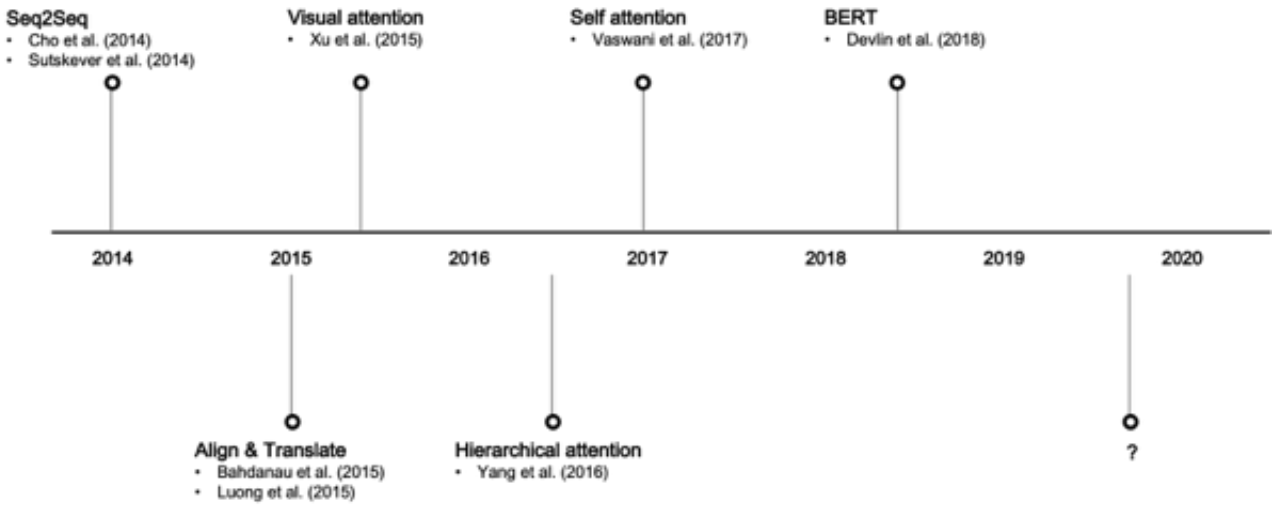
This simple yet powerful concept has revolutionized the field, bringing out many breakthroughs

in not only natural language processing (NLP) tasks, but also recommendation, healthcare analytics, image processing, speech recognition, etc.

Therefore, in this posting series, I will illustrate the development of the attention mechanism in neural networks *with emphasis on applications and real-world deployment*. I will try to implement as many attention networks as possible with Pytorch from scratch - from data import and processing to model evaluation and interpretations.

Final disclaimer is that I am **not** an expert or authority on attention. The primary purpose of this posting series is for my own education and organization. However, I am sharing my learning process here to help anyone who is eager to learn new things, just like myself. Please do not heistate leave a comment if you detect any mistakes or errors that I make, or you have any other (great) ideas and suggestions. Thank you for reading my article.

Key developments in attention



Attention mechanism was first proposed in the NLP field and still actively researched in the field. Above is the key designs and seminal papers that led to major developments. Here, I will briefly review them one by one.

- Seq2Seq, or RNN Encoder-Decoder (Cho et al. (2014), Sutskever et al. (2014))
- Alignment models (Bahdanau et al. (2015), Luong et al. (2015))
- Visual attention (Xu et al. (2015))
- Hierarchical attention (Yang et al. (2016))

- Transformer (Vaswani et al. (2017))

Sequence to sequence (Seq2Seq) architecture for machine translation

Many text information is in a sequence format, e.g., words, sentences, and documents. Seq2Seq is a two-part deep learning architecture to map sequence inputs into sequence outputs. It was initially proposed for the machine translation task, but can be applied for other sequence-to-sequence mapping tasks such as captioning and question retrieval.

Cho et al. (2014) and Sutskever et al. (2014) indepedently proposed similar deep learning architectures comprising two recurrent neural networks (RNN), namely encoder and decoder.

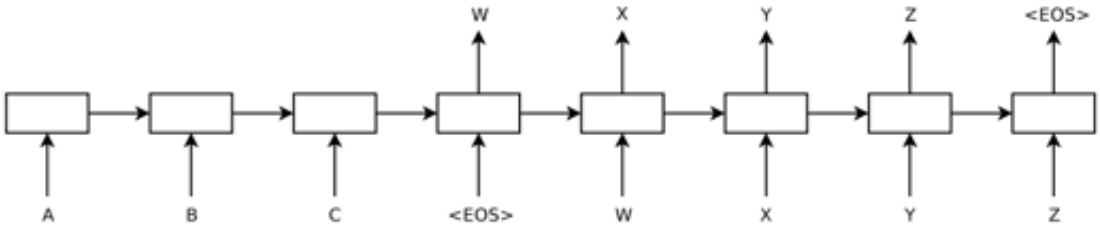


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

[Image source: Sutskever et al. (2014)]

The encoder reads a sequence input with variable lengths, e.g., English words, and the decoder produces a sequence output, e.g., corresponding French words, considering the hidden state from the encoder. The hidden state sends source information from the encoder to the decoder, linking the two. Both the encoder and decoder consist of RNN cells or its variants such as LSTM and GRU.

Align & Translate

A potential problem of the vanilla Seq2Seq architecture is that some information might not be captured by a fixed-length vector, i.e., the final hidden state from the encoder ($h_t h_t$). This can be especially problematic when processing long sentences where RNN is unable to send

adequate information to the end of the sentences due to gradient exploding, etc.

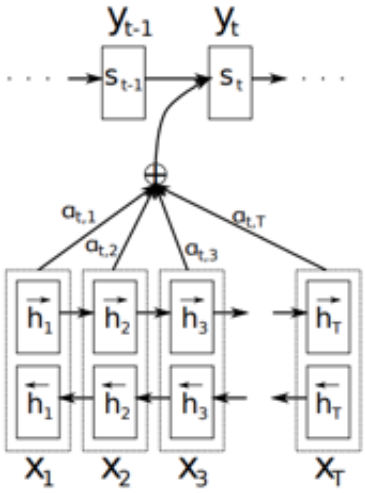


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

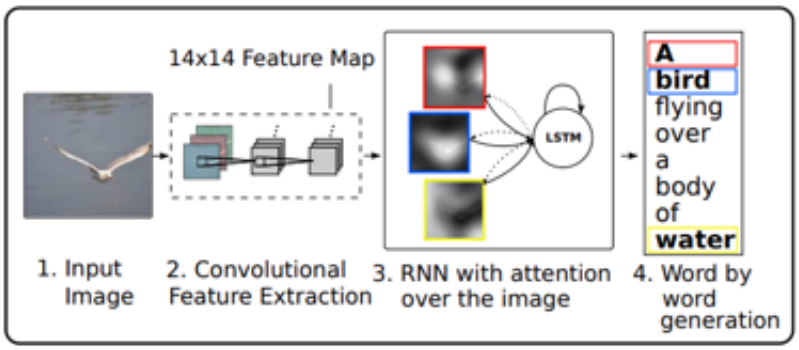
[Image source: Bahdanau et al. (2015)]

Therefore, Bahdanau et al. (2015) proposed utilizing a context vector to align the source and target inputs. The context vector preserves information from all hidden states from encoder cells and aligns them with the current target output. By doing so, the model is able to *"attend to"* a certain part of the source inputs and learn the complex relationship between the source and target better. Luong et al. (2015) outlines various types of attention models to align the source and target.

Visual attention

Xu et al. (2015) proposed an attention framework that extends beyond the conventional Seq2Seq architecture. Their framework attempts to align the input image and output word, tackling the image captioning problem.

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in Sections 3.1 & 5.4



[Image source: Xu et al. (2015)]

Accordingly, they utilized a convolutional layer to extract features from the image and align such features using RNN with attention. The generated words (captions) are aligned with specific parts of the image, highlighting the relevant objects as below. Their framework is one of the earlier attempts to apply attention to other problems than neural machine translation.

Figure 4. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

[Image source: Xu et al. (2015)]

Hierarchical attention

Yang et al. (2016) demonstrated with their hierarchical attention network (HAN) that attention can be effectively used on various levels. Also, they showed that attention mechanism applicable to the classification problem, not just sequence generation.

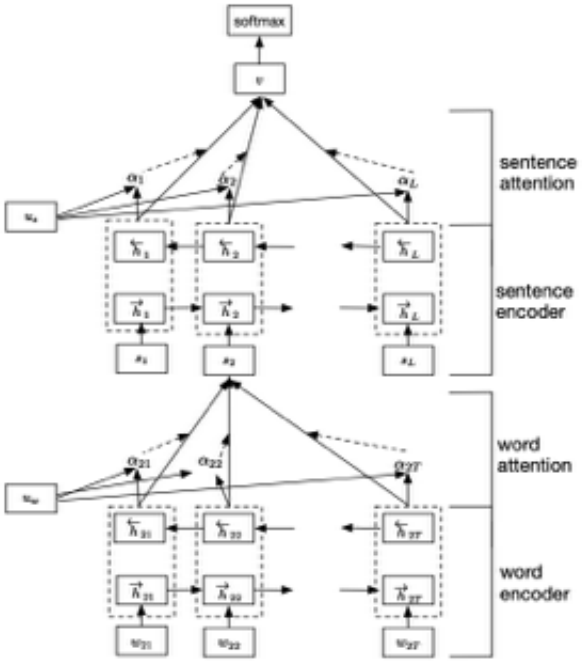


Figure 2: Hierarchical Attention Network.

[Image source: Yang et al. (2016)]

HAN comprises two encoder networks - i.e., word and sentence encoders. The word encoder processes each word and aligns them a sentence of interest. Then, the sentence encoder aligns each sentence with the final output. HAN enables hierarchical interpretation of results as below. The user can understand (1) which sentence is crucial in classifying the document and (2) which part of the sentence, i.e., which words, are salient in that sentence.

pork belly = delicious . || scallops? || I don't even like scallops, and these were a-m-a-z-i-n-g . || fun and tasty cocktails. || next time I in Phoenix, I will go back here. || Highly recommend.

Figure 1: A simple example review from Yelp 2013 that consists of five sentences, delimited by period, question mark. The first and third sentence delivers stronger meaning and inside, the word *delicious*, *a-m-a-z-i-n-g* contributes the most in defining sentiment of the two sentences.

[Image source: Yang et al. (2016)]

Transformer and BERT

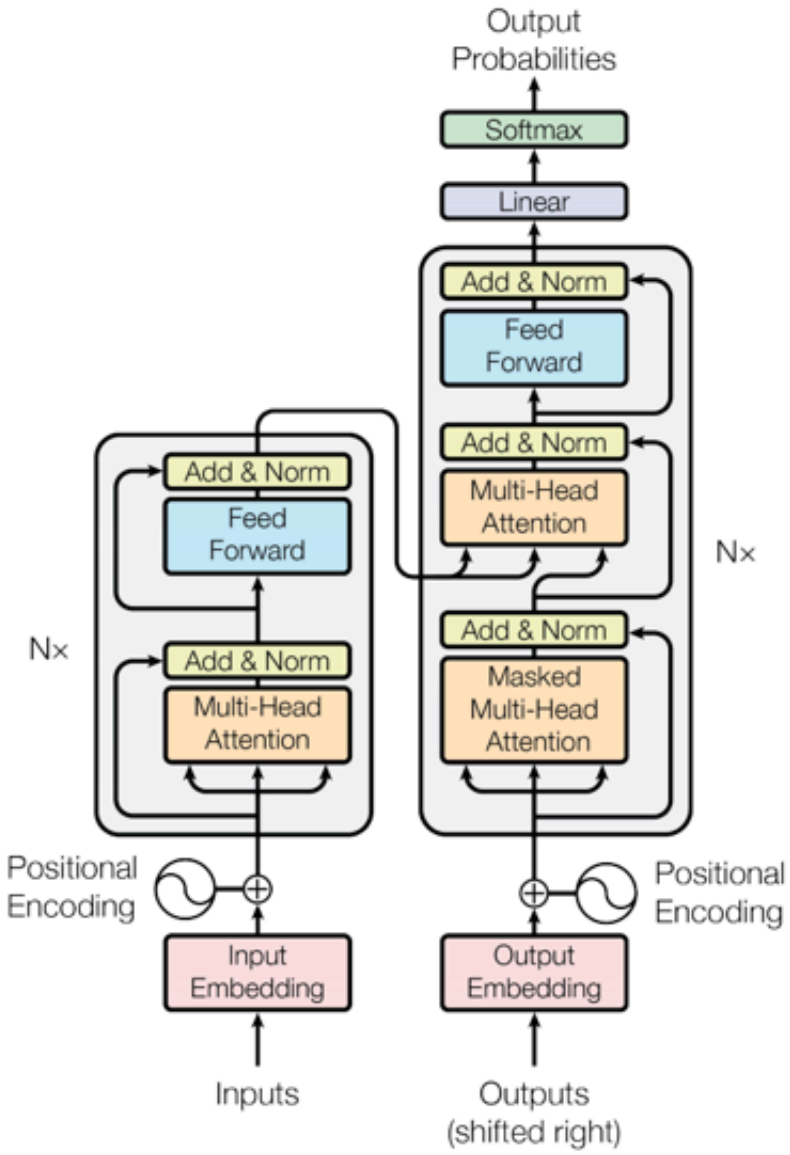


Figure 1: The Transformer - model architecture.

[Image source: Vaswani et al. (2017)]

The Transformer neural network architecture proposed by Vaswani et al. (2017) marked one of the major breakthroughs of the decade in the NLP field. The multi-head self-attention layer in Transformer aligns words in a sequence with other words in the sequence, thereby calculating a representation of the sequence. It is not only more effective in representation, but also more computationally efficient compared to convolution and recursive operations.

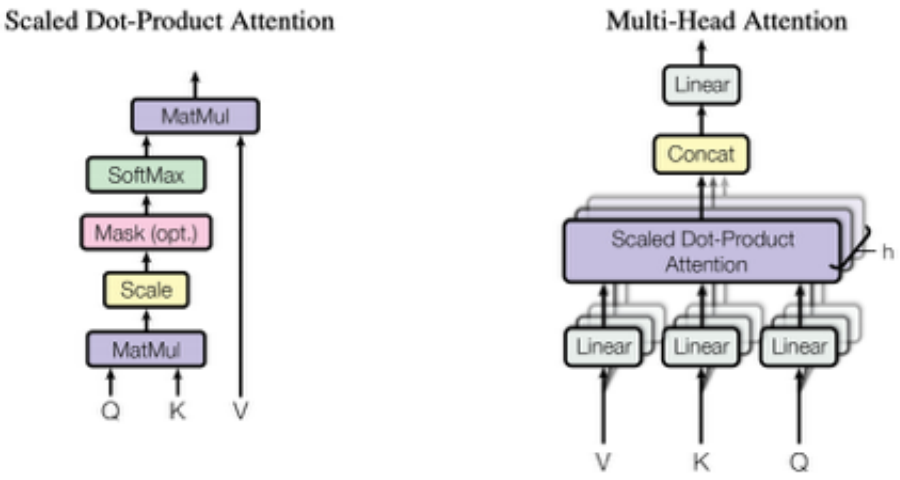


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

[Image source: Vaswani et al. (2017)]

Thus, the Transformer architecture discards the convolution and recursive operations and replaces them with multi-head attention. The multi-head attention is essentially multiple attention layers jointly learning different representations from different positions.

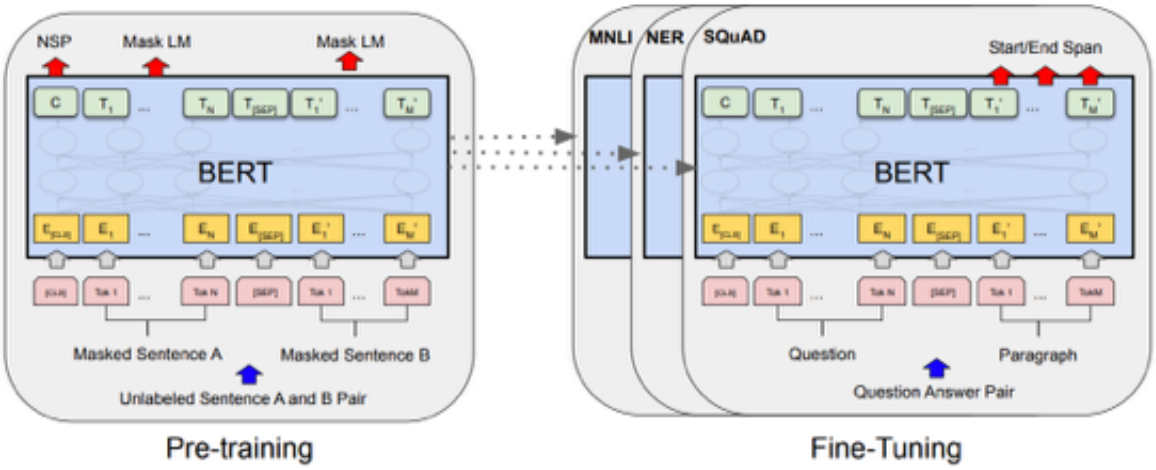


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

[Image source: Devlin et al. (2018)]

The intuition behind Transformer inspired a number of researchers, leading to the development of self-attention-based models such as Bidirectional Encoder Representations from Transformers (BERT) by Devlin et al. (2019). BERT pretrains bidirectional representations with the improved Transformer architecture. BERT shows state-of-the-art performance in various NLP tasks as of 2019. And there have a number of transformer-based language models that showed breakthrough results such as XLNet, RoBERTa, GPT-2, and ALBERT.

Vision Transformer

In the last few years, Transformer definitely revolutionalized the NLP field. Transformer-inspired models such as GPT and BERT showed record-breaking results on numerous NLP tasks. With that said, Dosovitskiy et al. (2020) demonstrated claimed that Transformer can be used for computer vision tasks, which is another *AI-complete problem*. This might sound a bit outdated since attention has been used for image-related tasks fairly extensively, e.g., Xu et al. (2015).

However, Dosovitskiy et al. (2020)'s claim is revolutionary since in their proposed model architecture, Transformer virtually replaces convolutional layers rather than complementing them. Furthermore, the *Vision Transformer* outperforms state-of-the-art, large-scale CNN models when trained with sufficient data. This might mean that CNN's golden age, which lasted for years, can come to end similar to that of RNN by Transformer.

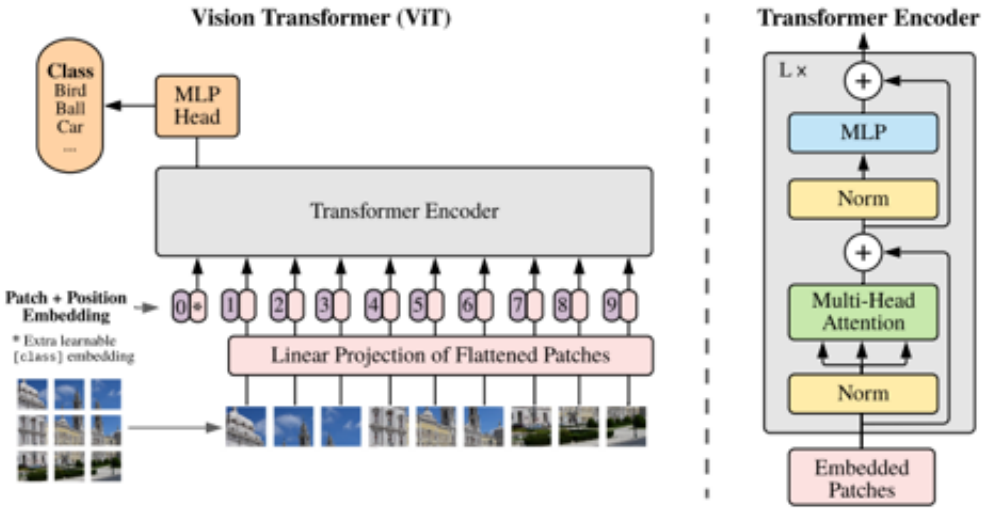


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

[Image source: Dosovitskiy et al. (2020)]

Other applications

I have outlined major developments in attention with emphasis on NLP in this posting. However, attention mechanism is now widely used in a number of applications as mentioned. Below are some examples of successful applications of attention in other domains. However, attention mechanism is very actively researched nowadays and it is expected that there will be (is) more and more domains welcoming the application of attentional models.

- Heathcare: Choi et al. (2016)
- Speech recognition: Chorowski et al. (2015)
- Graph attention networks: Velickovic´ et al. (2018)
- Recommender systems: Seo et al. (2017) Tay et al. (2018)
- Self-driving cars: Kim and Canny (2017)

In this posting, the concept of attention mechanism was gently introduced and major developments so far were outlined. From the next posting, we will look into details of key designs of seminal models. Let's start out with the Seq2Seq model that motivated the

development of alignment models.

References

- Cho et al. (2014)
- Sutskever et al. (2014)
- Bahdanau et al. (2015)
- Luong et al. (2015)
- Xu et al. (2015)
- Yang et al. 2016
- Vaswani et al. (2017)
- Devlin et al. (2019)
- Dosovitskiy et al. (2020)

Videos for more intuitive and in-depth explanations on attention...

- Attention and Memory in Deep Learning (DeepMind)
- Neural Machine Translation and Models with Attention (Chris Manning)
- Attention Model (Andrew Ng)

Links to this posting series

- Seq2Seq (1) - Introduction to Seq2Seq
- Seq2Seq (2) - Preparing data for machine translation Seq2Seq / Colab Notebook
- Seq2Seq (3) - Implementation of Seq2Seq / Colab Notebook
- Seq2Seq (4) - Training and evaluating Seq2Seq / Colab Notebook
- Seq2Seq (5) - Variant of Seq2Seq / Colab Notebook
- Seq2Seq (6) - Mini-batch Seq2Seq / Colab Notebook

Related Posts

Matrix Factorization with fast.ai - Collaborative filtering with Python 16 27 Nov 2020

Deep Recommender Systems - Collaborative filtering with Python 15 15 Nov 2020

Implementing Matrix Factorization models in Python - Collaborative filtering with Python 14 22 Oct 2020

ALSO ON **BUOMSOOKIM**

Deep learning state of the art 2020 ...

8 months ago • 1 comment

This is one of talks in MIT deep learning series by Lex Fridman on state of the ...

다층 퍼셉트론 1 (Regression with ...

3 years ago • 4 comments

_____...
Layer (type) Output Shape Param # ...

4. Sequence-to-Sequence ...

10 months ago • 8 comments

Attention Mechanism in Neural Networks - 4. Sequence-to-Sequence ...

Impo Googl

2 years

Note: T how to time fr

Thank you very much for this amazing post! I'm a newbie and I can say I understood everything you explained in this blog.

Reply • Share ›

Buomsoo Kim Mod ↗ mitra mirshafiee • 4 months ago

So glad to hear that! Thank you very much for your kind words!!

Reply • Share ›

Karthik Srinivasan • 10 months ago

Fantastic post!

Reply • Share ›

Buomsoo Kim Mod ↗ Karthik Srinivasan • 10 months ago

Thank you very much for reading!

Reply • Share ›