Attention and Transformer Networks

N.M. Militsis, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 4.1

Artificial Intelligence & Information Analysis Lab

Attention and Transformer Networks

- **Motivation**
- Transformer architecture
- Input embeddings
- Positional encoding
- Scaled dot-product attention
- Layer normalization
- Residual connection
- Training
- Transformers efficiency

Motivation

- Numerous real-world problems require to process sequences of variable length L.

Sequence is an ordered collection of data points, dependent to one another, indexed in time or space. Each data point is a vector denoted as:

$$\mathbf{x}_l \in \mathbb{R}^d, \qquad l = 1, \ldots, L.$$

Motivation

RNN limitations

- ***Slow convergence*** during training and ***high inference time.*** Model architecture prevents parallelization.

- ***Exploding and vanishing gradients***. When unfolded through time, the model depth is proportional to the input length L.

- ***Long-range dependencies*** are bottlenecked by a fixed size memory.

Motivation

Dealing with RNNs limitations

- ***Gradient clipping*** [PAS2012] prevents ***exploding gradients***.

- ***Gating mechanisms*** applied in Gated Reccurent Units (GRUs) [CHO2014] and Long Short-Term Memory networks (LSTMs) [HOC1997] deal with ***vanishing gradients***.

Motivation

CNNs limitations

- ***Local interactions*** are considered in each layer.

- High ***computational complexity***. The number of operations required to capture ***long-term dependencies***, grows linearly [GEH2017] or logarithmically [KAL2016] with the distance between sequence samples.

Motivation

Transformers

Transformer networks [VAS2017] deal with the limitations of RNNs and CNNs, leveraging ***attention mechanisms*** not as a supplement to standard convolutional or recurrent operations but as a standalone module.

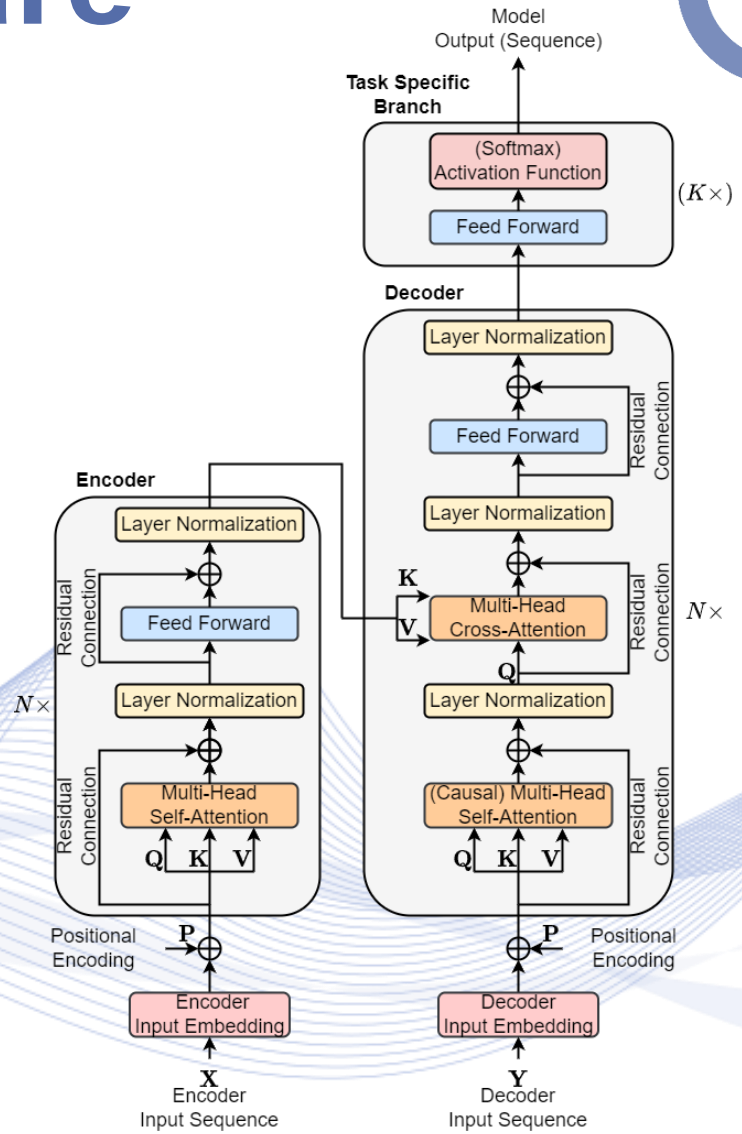
Attention and Transformer Networks

- Motivation
- **Transformer architecture**
- Input embeddings
- Positional encoding
- Scaled dot-product attention
- Layer normalization
- Residual connection
- Training
- Transformers efficiency

Transformer architecture

Transformer was originally designed for neural sequence transduction.

It has an ***encoder-decoder*** structure followed by one or more ***task specific branches***.



General Transformer architecture.

Transformer architecture

Encoder

- The encoder consists of a cascade of N_B identical blocks.

- Each block has two sub-layers:
  - A ***multi-head self-attention module***.
  - A ***position-wise*** fully connected ***feed-forward*** network.

- ***Residual connection*** [HE2016] is employed around each sub-layer followed by ***layer normalization*** [BA2016].

Transformer architecture

Decoder

- The decoder also consists of a cascade of N_B identical blocks.

- Each block has three sub-layers:
  - A (***causal***) ***multi-head self-attention module***. Optionally a mask is employed to prevent current data point from attending subsequent ones.
  - A ***multi-head cross-attention module*** between encoder and decoder sequences.
  - A ***position-wise*** fully connected ***feed-forward*** network.

- Again, residual connection and layer normalization are

Transformer architecture

- The input sequences to the encoder and the decoder having lengths $L \neq L'$ form matrices $\mathbf{X} \in \mathbb{R}^{L \times d}$ and $\mathbf{Y} \in \mathbb{R}^{L' \times d}$ respectively.
- In the ***original*** version, the encoder maps an input sequence $\mathbf{X}$ to a latent sequence $\mathbf{Z} \in \mathbb{R}^{L \times d_m}$.
- Given $\mathbf{Z}$, the decoder generates one data sample at a time of an output sequence $\boldsymbol{z}'_l \in \mathbb{R}^{d'}, l = 1, \ldots, L$, forming matrix $\boldsymbol{Z}' \in \mathbb{R}^{L' \times d'}$.

Transformer architecture

- At each step, the ***original*** recursive decoder (autoregressive) employs the previously generated output sequence samples when generating the current one.

- ***Typical application: machine translation.***
- After the decoder, one task-specific layer consists of a linear layer followed by SoftMax activation function.

- This way, in each step, the ***original model*** approaches Neural Machine Translation (NMT) as a classification task.

Transformer architecture

Originally*, a transformer network was trained in *supervised fashion and tested on the English-to-German and English-to-French newstest2014 tests outperforming (using BLUE score) previous state-of-the-art models.

Currently, it is also trained on vast amounts of data in an ***unsupervised*** manner [DEV2018, RAD2018].

- Then it is fine-tuned on downstream tasks.

Attention and Transformer Networks

- Motivation
- Transformer architecture
- **Input embeddings**
- Positional encoding
- Scaled dot-product attention
- Layer normalization
- Residual connection
- Training
- Transformers efficiency

Input embeddings

- A ***linear projection*** is used to convert the encoder or decoder input sequence samples to ***embeddings*** of dimension d_m:

$$\mathbf{x}_{el} = \mathbf{W}_e \mathbf{x}_l.$$

- To facilitate the following residual connections, all sub-layers in the model produce outputs of the same dimension d_m.

- In the ***original model***, the embedding layers of the encoder and the decoder, ***arbitrarily*** share the same weight matrix $\mathbf{W}_e$ multiplied by $\sqrt{d_m}$.

Input embeddings

- Input embeddings are computed as follows:

$$\mathbf{X}_e = \sqrt{d_m}\mathbf{X}_i\mathbf{W}_e$$

$$\mathbf{Y}_e = \sqrt{d_m}\mathbf{Y}_i\mathbf{W}_e.$$

where $\mathbf{W}_e \in \mathbb{R}^{d \times d_m}$.

Attention and Transformer Networks

- Motivation
- Transformer architecture
- Input embeddings
- **Positional encoding**
- Scaled dot-product attention
- Layer normalization
- Residual connection
- Training
- Transformers efficiency

Positional encoding

Challenge

- Transformers contain attention matrices formed by (transformed) vector dot products.
- Scaled dot-product attention is ***permutation equivariant***. That is, shifted versions of the input sequence samples lead to the same (shifted) output sequence.
- Therefore, Transformers are sequence order ***agnostic***.

- However, ***sequence semantics*** highly ***depend on the input sequence order***.

Positional encoding

- In the ***original model***, positional information is provided through ***additive*** vectors of the same dimension d_m as the input embeddings.

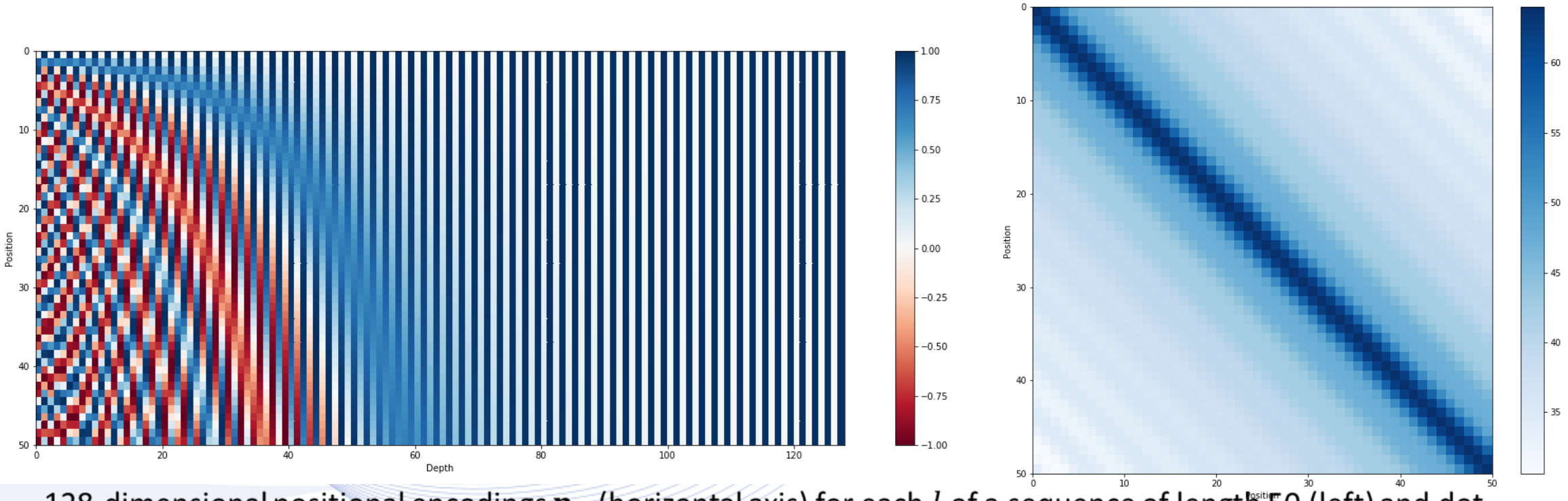
$$\mathbf{x}'_{e_l} = \mathbf{x}_{e_l} + \mathbf{p}_l.$$

Each dimension of a positional encoding is a ***sinusoidal function***:

$$p_{l,2d} \triangleq \sin\left(l/1000^{2d/d_m}\right),$$
$$p_{l,2d+1} \triangleq \cos\left(l/1000^{2d/d_m}\right), d = 1, \ldots, d_m, \qquad l = 1, \ldots, L.$$

Positional encoding

- Input embeddings with ***similar relative position*** in the sequence have ***similar positional encodings***.



128-dimensional positional encodings $\mathbf{p}_l$ (horizontal axis) for each l of a sequence of length 50 (left) and dot-products of $\mathbf{p}_l$ and $\mathbf{p}_k$ (right) for $l, k = 1, \ldots, 50$ [KAZ2019].

Attention and Transformer Networks

- Motivation
- Transformer architecture
- Input embeddings
- Positional encoding
- **Scaled dot-product attention**
- Layer normalization
- Residual connection
- Training
- Transformers efficiency

Scaled dot-product attention

Three new sequences $\mathbf{Q} \in \mathbb{R}^{L \times d_k}$ (***queries***), $\mathbf{K} \in \mathbb{R}^{L' \times d_k}$(***keys***), $\mathbf{V} \in \mathbb{R}^{L' \times d_v}$ (***values***) are generated:

$$\mathbf{Q} = \mathbf{X}'_e \mathbf{W}_Q \left(+\mathbf{1}_{L \times 1} \mathbf{b}_Q\right), \quad \mathbf{W}_Q \in \mathbb{R}^{d_m \times d_k}, \mathbf{b}_Q \in \mathbb{R}^{d_k},$$
$$\mathbf{K} = \mathbf{Y}'_e \mathbf{W}_K (+\mathbf{1}_{L' \times 1} \mathbf{b}_K), \quad \mathbf{W}_K \in \mathbb{R}^{d_m \times d_k}, \mathbf{b}_K \in \mathbb{R}^{d_k},$$
$$\mathbf{V} = \mathbf{Y}'_e \mathbf{W}_V (+\mathbf{1}_{L' \times 1} \mathbf{b}_V), \quad \mathbf{W}_V \in \mathbb{R}^{d_m \times d_v}, \mathbf{b}_V \in \mathbb{R}^{d_v},$$

by linearly transforming two sequences $\mathbf{X}'_e \in \mathbb{R}^{L \times d}$ and $\mathbf{Y}'_e \in \mathbb{R}^{L' \times d}$, where $L \neq L'$,

- Learnable weight matrices $\mathbf{W}_Q, \mathbf{W}_K$ $\mathbf{W}_V$.
- In the ***original model***, it is arbitrarily chosen that $d_k = d_v$.

Scaled dot-product attention

- Using the terminology in [GRAV2014], attention is an averaging of ***values***, associated to ***keys*** matching to specific ***queries***.

In ***cross-attention*** each data point of sequence $\mathbf{X}'_e$ attends to all data points of sequence $\mathbf{Y}'_e$ in order to compute a new representation of sequence $\mathbf{X}'_e$:

$$\mathbf{X}''_e = Softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}.$$

The ***row-wise Softmax operator*** renders a probability distribution, representing the normalized correlation scores of each query to all the keys.

Scaled dot-product attention

- Each row of the new representation $\mathbf{X}''_e$ is a weighted average of the rows of $\mathbf{V}$, using the normalized correlation scores generated by the Softmax operator.

The normalization by $\sqrt{d_k}$ is ***arbitrarily*** chosen in the ***original model*** as it leads to more stable gradient values during training, thus avoiding the exploding gradients problem.

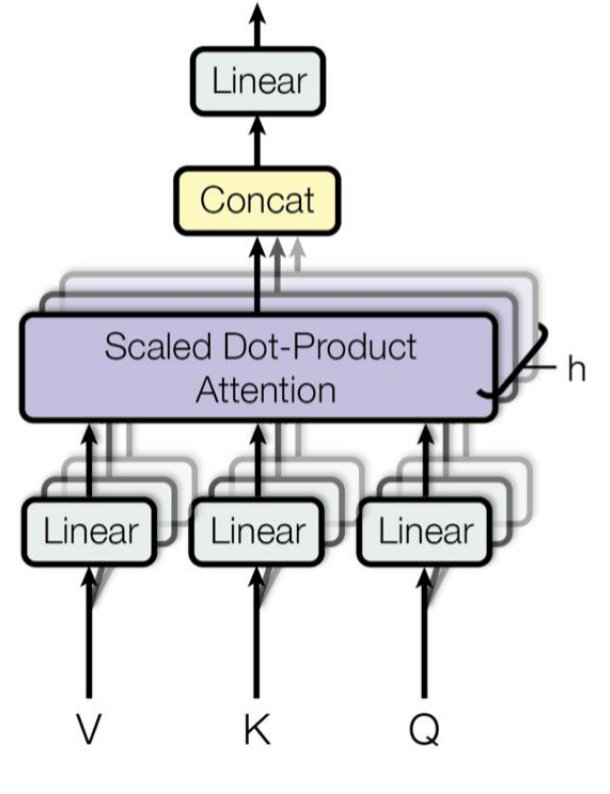
When $\mathbf{X}'_e \equiv \mathbf{Y}'_e$, the attention mechanism is called ***self-attention*** or ***intra-attention***.

Scaled dot-product attention

Multi-headed attention

- The scaled dot-product attention is ***independently*** computed H times ***in parallel*** forming the so-called ***attention heads***.

- Multiple ***independent attention heads*** are intended to improve the expressivity of the model, including disambiguating different semantic uses of the same input embedding.



Multi-headed attention [VAS2017].

Scaled dot-product attention

Multi-headed attention

- Multi-headed attention facilitates adding ***more parameters*** into the network, ***exploiting*** the parallel computing capabilities of ***GPUs***.

- It is an efficient way of ***increasing*** the ***width*** instead of the depth of the network ***without*** adding significant ***computational complexity***.

Scaled dot-product attention

Multi-headed attention

A maximal number of independent attention heads is chosen:

$$H = d_m / d_k.$$

The sequences $\mathbf{X}''_{e_i} \in \mathbb{R}^{L \times d_v}, i = 1, \ldots, H$ produced by each head are horizontally concatenated and is linearly projected using a learnable matrix $\mathbf{W}_o \in \mathbb{R}^{Hd_v \times d_m}$:

$$\mathbf{X}''_{emb} = \left[\mathbf{X}''_{e_1} || \ldots \mathbf{X}''_{e_i} \ldots || \mathbf{X}''_{e_H}\right] \mathbf{W}_o.$$

Attention and Transformer Networks

- Motivation
- Transformer architecture
- Input embeddings
- Positional encoding
- Scaled dot-product attention
- **Layer normalization**
- Residual connection
- Training
- Transformers efficiency

Layer normalization

Layer normalization is applied point-wise for each sequence sample $\mathbf{x}'_{e_l} \in \mathbb{R}^{d_m}, l = 1, \ldots, L$:

$$\mathbf{x}''_{e_l} = \left(\frac{\mathbf{x}'_{e_l} - \boldsymbol{\mu}_l}{\sigma_l} \right) \odot \boldsymbol{\gamma} + \boldsymbol{\beta},$$

- $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{d_m}$ are learnable parameter vectors.
- $\odot$ denotes elementwise product.
- Vector $\boldsymbol{\mu}_l$ entries μ_l and σ_l are computed using the elements of $\mathbf{x}'_{e_l}$:

$$\mu_l = \frac{1}{\cdot} x_d \qquad \sigma_l = \sqrt{\sum (x_d - \mu_l)^2}.$$

Attention and Transformer Networks

- Motivation
- Transformer architecture
- Input embeddings
- Positional encoding
- Scaled dot-product attention
- Layer normalization
- **Residual connection**
- Training
- Transformers efficiency

Residual connection

Given a specific input embedding $\mathbf{x}'_{e_l}$, a ***residual connection*** is applied as follows:

$$\mathbf{x}''_{e_l} = F(\mathbf{x}'_{e_l}) + \mathbf{x}'_{e_l}.$$

- F: operator representing a feed-forward network or multi-head attention.
- Residual connections are empirically shown to facilitate training convergence.
- The addition $F(\mathbf{x}'_{e_l}) + \mathbf{x}'_{e_l}$ in forward pass results in duplicating the corresponding gradient in backward pass to deal with the vanishing gradients problem.

Attention and Transformer Networks

- Motivation
- Transformer architecture
- Input embeddings
- Positional encoding
- Scaled dot-product attention
- Layer normalization
- Residual connection
- **Training**
- Transformers efficiency

Transformer Training

Classification

- A sufficient large training sample set $\mathcal{D}$ is required for ***Supervised Learning*** (regression, classification):

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N\}.$$

- $\mathbf{x}_i \in \mathbb{R}^d$: d –dimensional input (feature) vector of the i-th training sample.
- $\mathbf{y}_i$: its target label (output), ***not to be confused with decoder input*** having the same notation $\mathbf{y}_l$.
- In classification tasks, $\mathbf{y}_i \in [0,1]^m$.

- ***Training***: Given N pairs of training samples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots N\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{y}_i \in [0,1]^m$, estimate $\boldsymbol{\theta}$ by minimizing a loss function: $\min J(\mathbf{y}, \hat{\mathbf{y}})$

Transformer Training

- Depending on the task to be solved, an appropriate loss function $J(\boldsymbol{\theta})$ is formed and then optimized through a ***gradient descent-based criterion***.

- Empirically, it has been shown that optimizing the loss function with ***Adam*** [KIN2015] ***outperforms*** the ***Stochastic Gradient Descent*** (SGD) counterpart.

- The main reason for the poor performance of SGD is that stochastic gradients are accompanied by a heavy-tailed noise distribution.

Transformer Training

- Depending on the task to be solved, an appropriate loss function $J(\boldsymbol{\theta})$ is formed and then optimized through a ***gradient descent-based criterion***.

- Empirically, it has been shown that optimizing the loss function with ***Adam*** [KIN2015] ***outperforms*** the ***Stochastic Gradient Descent*** (SGD) counterpart.

- The main reason for the poor performance of SGD is that stochastic gradients are accompanied by a heavy-tailed noise distribution.

Transformer Training

When using Adam optimizer, in every iteration t of the training, the network parameters $\boldsymbol{\theta}_t$ are updated as follows through back propagation:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - a\left(\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon}\right).$$

- a: learning rate
- ε: a constant hyperparameter.

Transformer Training

- The vectors $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ are computed as follows:

$$\hat{\mathbf{m}}_t = \mathbf{m}_t/(1-\beta_1^t), \qquad \mathbf{m}_t = \beta_1\mathbf{m}_{t-1} + (1-\beta_1)\nabla J$$
$$\hat{\mathbf{v}}_t = \mathbf{v}_t/(1-\beta_2^t), \qquad \mathbf{v}_t = \beta_2\mathbf{v}_{t-1} + (1-\beta_2)(\nabla J)^2.$$

- β_1, β_2: discount factors (hyperparameters).
- ∇J: the gradient of J with respect to $\boldsymbol{\theta}_{t-1}$.

- The terms $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ facilitate bias correction, preventing the too high gradient values in the early iterations.

Transformer Training

- The $\mathbf{m}_t$ term is an exponentially weighted average of the gradients ∇J (***fisrt moment***).
- The discount factor β_1 (***momentum***) accounts for the noise imposed by SGD.

- The $\mathbf{v}_t$ term is an exponentially weighted average of the second-order gradients $(\nabla J)^2$ (***second moment***).
- It hinders too large or small steps towards the steepest descent when the loss function J is too inclined or flat, respectively.

Transformer Training

Usually, the loss function J contains an L_2 regularization term $\|\boldsymbol{\theta}_{t-1}\|^2$.

- The ***first moment*** in Adam is computed as follows:

$$(\nabla J)' = \nabla J + 2d\boldsymbol{\theta}_{t-1}.$$

- $2d\boldsymbol{\theta}_{t-1}$ is the first order gradient of the L_2 regularization term $d\|\boldsymbol{\theta}_{t-1}\|^2$.
- d is hyperparameter.

In the variant ***AdamW*** [LOS2019], the L_2 regularization is applied directly into the update step:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha\left(\frac{\hat{\mathbf{m}}_t}{\quad} + d\boldsymbol{\theta}_{t-1}\right)$$

Transformer Training

- AdamW is the most widely used optimization algorithm used in Transformer training.
- It tends to have better convergence behavior compared to Adam, especially when the L_2 regularization improves generalization performance.

- AdamW has gained popularity in Natural Language Processing (NLP) tasks, particularly in transformer-based models like BERT (Bidirectional Encoder Representations from Transformers).

Attention and Transformer Networks

- Motivation
- Transformer architecture
- Input embeddings
- Positional encoding
- Scaled dot-product attention
- Layer normalization
- Residual connection
- Training
- **Transformers efficiency**

Transformers efficiency

- ***Parallelizable*** operations fully exploiting modern hardware (GPUs) and reducing FLOPs.
- ***Long-range semantic dependencies*** are efficiently captured.
- Model ***performance scales*** very well with the number of ***parameters***.
- Transformer can be ***pre-trained*** in ***unsupervised*** fashion and then ***fine-tuned*** in ***supervised*** way on downstream tasks.
- ***State-of-the-art performance*** in various modalities (text, image, audio, etc.)

Bibliography

[RUM1986] D. Rumelhard, G. Hinton, R. Williams, "Learning representations by back-propagating errors", Nature, issue 323, pp. 533-536, 1986.

[WAI1989] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K.J. Lang, "Phoneme recognition using time-delay neural networks", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, issue 3, pp. 328-339, 1989.

[BAI2018] S. Bai, J.Z. Kolter, V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling", arxiv preprint arXiv:1803.01271, 2018.

[PAS2012] R. Pascanu, T. Mikolov, Y. Bengio "On the difficulty of training Recurrent Neural Networks", arxiv preprint arXiv:1211.5063, 2012.

[CHO2014] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation", Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[HOC1997] S. Hochreiter, J. Schmidhuber "Long Short-Term Memory", Neural Computation, vol. 9, issue 8, pp. 1735-1780, 1997

Bibliography

[BAH2015] D. Bahdanau, K. Cho, Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", International Conference on Learning Representations (ICLR), 2015.

[GEH2017] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y.N. Dauphin "Convolutional Sequence to Sequence Learning", arxiv preprint arXiv:1705.03122, 2017.

[KAL2016] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, K. Kavukcuoglou, "Neural Machine Translation in Linear Time", arxiv preprint arXiv:1610.10099, 2016.

[VAS2017] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, "Attention Is All You Need", Advances in Neural Information Processing Systems (NIPS), 2017.

[DEV2018] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arxiv preprint arXiv:1810.04805, 2018.

[RAD2018] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, "Improving Language Understanding by Generative Pre-Training", OpenAI Blog, 2018.

[HE2016] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778, 2016.

Bibliography

[BA2016] J.L BA, J.R. Kiros, G. E. Hinton, "Layer Normalization", arxiv preprint arXiv:1607.06450, 2016.

[KAZ2019] A. Kazemnejad, "Transformer Architecture: The Positional Encoding", https://kazemnejad.com/blog/transformer_architecture_positional_encoding/, 2019

[GRAV2014] A. Graves, G. Wayne, I. Danihelka, Neural Turing Machines, arxiv preprint arXiv:1410.5401, 2014

[KIN2015] D.P. Kingma, J. Ba, "Adam: A method for stochastic optimization", International Conference on Learning Representations (ICLR), 2015.

[LOS2019] I. Loshchilov, F. Hutter, "Decoupled weight decay regularization", International Conference on Learning Representations (ICLR), 2019.

Q & A

Thank you very much for your attention!

Contact: Prof. I. Pitas
pitas@csd.auth.gr