BAYESUVIUS

A VISUAL DICTIONARY OF BAYESIAN NETWORKS AND CAUSAL INFERENCE

ROBERT R. TUCCI

Bayesuvius, a visual dictionary of Bayesian Networks and Causal Inference

Robert R. Tucci
www.ar-tiste.xyz

February 21, 2024
This book is constantly being expanded and improved. To download the latest version, go to `https://github.com/rrtucci/Bayesuvius`

Bayesuvius
by Robert R. Tucci
Copyright ©2020-2023, Robert R. Tucci.

This work is licensed under the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States License. To view a copy of this license, visit the link `https://creativecommons.org/licenses/by-nc-nd/3.0/` or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042.

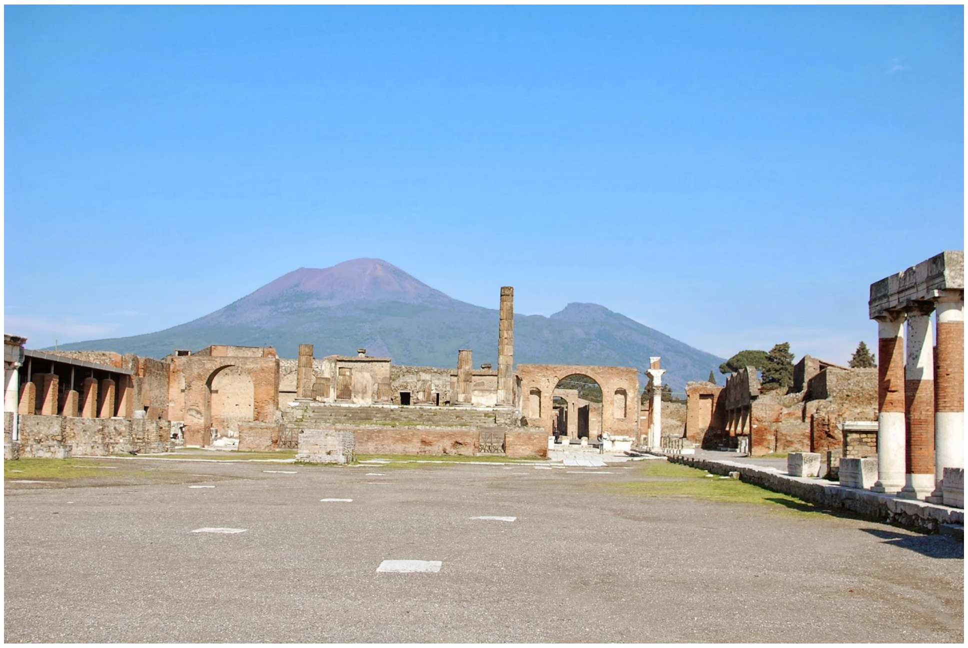


Figure 1: View of Mount Vesuvius from Pompeii

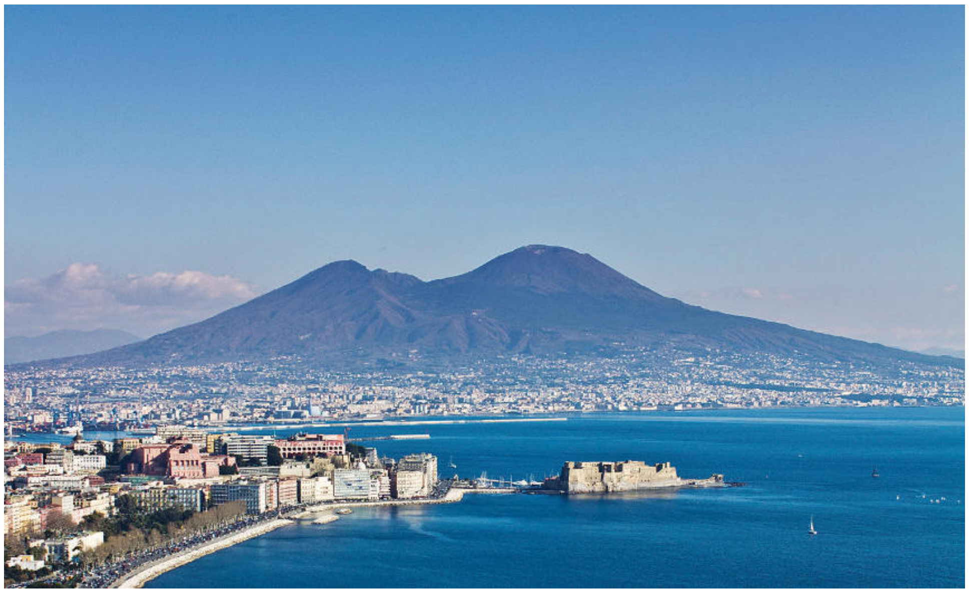


Figure 2: Mount Vesuvius and Bay of Naples

Contents

Appendices

Chapter 95

Transformer Networks

The primary reference for this chapter is Ref.[81]. Ref.[81] is the highly influential 2017 paper entitled “Attention is all you need" that introduced **Transformer Networks** (tranets) and Attention into the AI vernacular. Besides Ref.[81], I also read blog posts such as Ref.[29] and the Wikipedia article on tranet (Ref. [162]). For a complete list of the large number of excellent blog post that I read to learn about this subject, see my open source software texnn (Ref.[80]).[1]

Transformer Networks (tranets) have been taking the fields of Natural Language Processing (NLP) and Large Language Models (LLM) by storm in recent years. They were introduced in 2017 and already are the basis of numerous LLMs. Two famous examples are, BERT (Bidirectional Encoder Representations from Transformers) and ChatGPT (Generative Pre-trained Transformer). Both of these have been trained with huge databases, of which all of the English Wikipedia ($\sim 10^9$ words) is but a small part.

How well ChatGPT works was a huge surprise to most people, including experts in AI/ML. My conjecture is that this surprising LLM performance is due to causality. Let me explain. I believe tranets and the LLM that use them, are just curve-fitters (so are Least Squares, vanilla NNs, Convolutional NNs, etc.). But, we lucked out, because tranets are very good at fitting causal data, and the space of all human generated text, including math equations and computer code, is causally connected (i.e., has a **causally connected topology**.).

Normally, tranets are drawn as box diagrams that are somewhat cryptic and ambiguous, at least to me. In this chapter, instead of drawing them as box diagrams, I represent them as causal DAGs (bnets). This makes their causal nature more explicit than the box diagrams, and, in my opinion, also makes them less ambiguous and more understandable than the box diagrams.

Recurrent Neural Nets (RNNs) are discussed in Chapter 75. tranets are quickly displacing RNNs, an older method, in NLP. tranets are better than RNNs for doing

[1]texnn is Python software that I wrote specifically for drawing the bnets of this chapter, but later I generalized it to a stand-alone app that can draw any bnet (including SCMs, NNs and tranets), not just a tranet bnet.

NLP in several important ways. Whereas RNNs analyze the tokens (words) of a sentence sequentially (like a Kalman Filter), tranets analyze them in parallel, and thus are more amenable to parallel computing. Also, because RNNs analyze the words of a sentence sequentially, they tend to give more importance to the end of a sentence than to its beginning. That's because RNNs start forgetting the beginning of a sentence by the time they reach its end, like a patient with Alzheimer's. tranets do not suffer from this malady.

Dynamical bnets are discussed in Chapter 25. In Chapter 75, we showed that RNNs are dynamical bnets. In this chapter we will show that tranets are dynamical bnets too.

In this chapter, we will use the Numpy-like tensor notation discussed in Section C.49. In particular, note that $[n] = [0 : n] = \{0, 1, \ldots, n-1\}$ and that $T^{[n],[m]}$ is an $n \times m$ matrix.

95.1 Tensor Notation

Our tensor notation is discussed in Section C.49. Here is a quick review of some of the more salient facts in that section on tensors. Below, we will often accompany an equation in tensor component notation with the equivalent matrix equation, in parenthesis.

We use Greek letters for tensor indices.

Let $\alpha \in [a]$, $\beta \in [b]$, $\gamma \in [c]$, $\delta \in [d]$, $\nu \in [n]$, $\Delta \in [D]$.

- **reshaping**

$$T^{\nu,\delta} \to T^{\Delta} \quad \left(T^{[n_{\underline{h}}],[d]} \to T^{[D]}\right) \tag{95.1}$$

$$T^{\Delta} \to T^{\nu,\delta} \quad \left(T^{[D]} \to T^{[n_{\underline{h}}],[d]}\right) \tag{95.2}$$

- **concatenation**

$$T^{[n]} = (T^0, T^1, \ldots, T^{n-1}) = (T^\nu)_{\nu \in [n]} \tag{95.3}$$

- **Hadamard product (element-wise, entry-wise multiplication)**

$$T^{[n]} * S^{[n]} = (T^\nu S^\nu)_{\nu \in [n]} \tag{95.4}$$

- **Matrix multiplication**

  $T^{[n]} = T^{[n],[1]}$ is a column vector.

$$(T^{[n]})^T S^{[n]} = \text{scalar} \tag{95.5}$$

$$T^{[a],[b]}S^{[b],[c]} = \left[\sum_{\beta\in[b]} T^{\alpha,\beta}S^{\beta,\gamma}\right]_{\alpha\in[a],\gamma\in[c]} \tag{95.6}$$

Most treatments of tranets, including the "Attention is all you need" paper, order the operations chronologically from left to right (L2R). So if A occurs before B, they write AB. This is contrary to what is done in Linear Algebra, where one orders the operations chronologically from right to left (R2L), and one writes BA. In this chapter, will adhere to the Linear Algebra convention, since it is so prevalent and is the overwhelming precedent.

95.2 Recurrent Neural Net with Attention

95.2.1 Single Head Attention

Let

ℓ be the maximum number of words allowed in a sentence. Some words might be blanks (padding).

d be the so called **hidden or embedding dimension**.

$e^t_\alpha \in \mathbb{R}^d$ be a d-dimensional column vector for word $\alpha \in [\ell]$ at time t.

$W^t_{\underline{q}}, W^t_{\underline{k}}, W^t_{\underline{v}} \in \mathbb{R}^{d\times d}$ be the weight matrices for time slice t. The letters Q, K, V stand for Query, Key and Value, respectively. These matrices are learned by training the net. They transform e^t_α as follows

$$v^t_\alpha = W^t_{\underline{v}} e^t_\alpha \tag{95.7}$$

$$q^t_\alpha = W^t_{\underline{q}} e^t_\alpha \tag{95.8}$$

$$k^t_\alpha = W^t_{\underline{k}} e^t_\alpha \tag{95.9}$$

Fig.95.1 represents a tranet of a 3-word sentence as a dynamical bnet. The TPMs (Transition Probability Matrices), printed in blue, for bnet Fig.95.1, are as follows:

$$P(v^t_\alpha | e^t_\alpha) = \mathbb{1}(\quad v^t_\alpha = W^t_{\underline{v}} e^t_\alpha \quad) \tag{95.10}$$

$$P(q^t_\alpha | e^t_\alpha) = \mathbb{1}(\quad q^t_\alpha = W^t_{\underline{q}} e^t_\alpha \quad) \tag{95.11}$$

$$P(k^t_\alpha | e^t_\alpha) = \mathbb{1}(\quad k^t_\alpha = W^t_{\underline{k}} e^t_\alpha \quad) \tag{95.12}$$

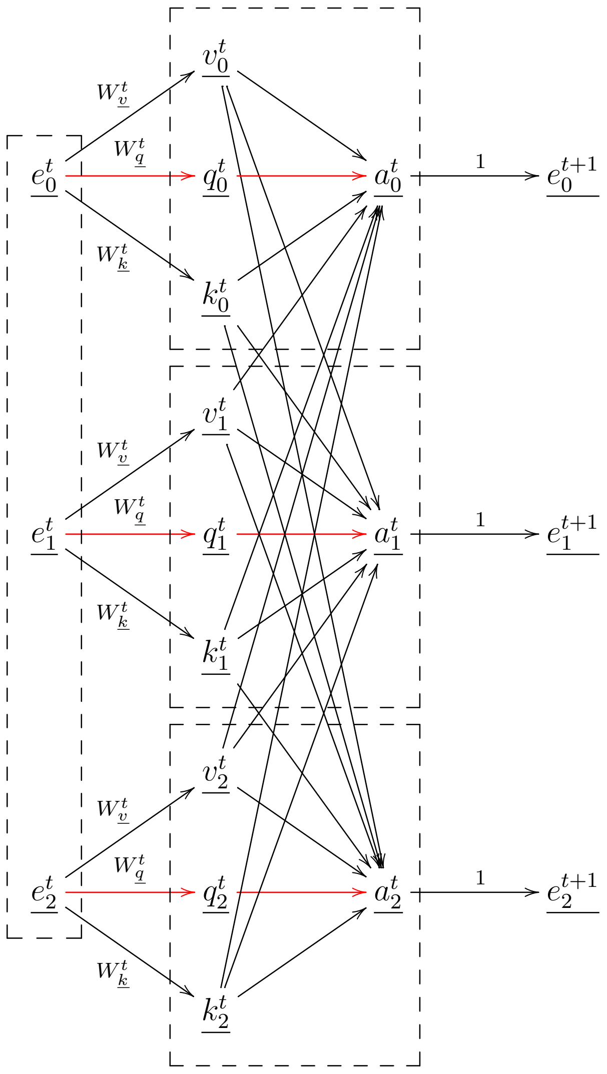


Figure 95.1: Dynamical bnet with single-head Attention for 3 words. Time-slice t. Note that k_α^t for all α points to $\underline{a}_{\alpha'}^t$ for all α'. Likewise, $\underline{v}_\alpha^t$ for all α points to $\underline{a}_{\alpha'}^t$ for all α'. However, $\underline{q}_\alpha^t$ points only to $\underline{a}_\alpha^t$.

$$P(e_\alpha^{t+1}|a_\alpha^t) = \mathbb{1}(\quad e_\alpha^{t+1} = a_\alpha^t \quad) \tag{95.13}$$

$$P(a_\alpha^{t+1}|v_\cdot^t, q_\alpha^t, k_\cdot^t) = \mathbb{1}(\quad a_\alpha^{t+1} = \sum_{\alpha' \in [\ell]} v_{\alpha'}^t P(\alpha'|\alpha) \quad) \tag{95.14}$$

where the conditional probability $P(\alpha'|\alpha)$ is defined as[2]

[2]The reason sums over $\delta \in [d]$ are divided by $\sqrt{d}$ is to prevent the argument of the exponential

$$P(\alpha'|\alpha) = \text{softmax}\left[\frac{1}{\sqrt{d}}\sum_{\delta\in[d]}(k^t)^{\delta,[\ell]}(q^t)^{\delta,\alpha}\right](\alpha'|\alpha) \tag{95.15}$$

$$= \frac{\exp\left(\frac{1}{\sqrt{d}}(k^t_{\alpha'})^T q^t_\alpha\right)}{\sum_{\alpha''\in[\ell]}\exp\left(\frac{1}{\sqrt{d}}(k^t_{\alpha''})^T q^t_\alpha\right)} \tag{95.16}$$

The right hand side of Eq.(95.14) constitutes an average over all the word vectors $\{\underline{v}^t_\alpha : \alpha \in [\ell]\}$ in a sentence. This average is called the **Attention** (for a single head).[3]

$$\boxed{\text{Attention}^{\delta,\alpha}\left((v^t)^{[d],[\ell]}, (k^t)^{[d],[\ell]}, (q^t)^{[d],[\ell]}\right) = \sum_{\alpha'\in[\ell]}(v^t)^{\delta,\alpha'}P(\alpha'|\alpha)} \tag{95.17}$$

On first encounter, the structure of an Attention bnet seems a bit mysterious. Then one realizes that this is an old friend. If the dashed boxes in Fig.95.1 are each "shrunk" to single nodes, then it becomes a TAN Bayes Net. Each of the 3 subgraphs $\underline{e}^t, (\underline{v}^t, \underline{q}^t, \underline{k}^t), \underline{a}^t$ also constitutes a TAN Bayes net. [4].[5] In broad terms, Fig.95.1 can be described by saying that each word undergoes a special kind of 3-class (q,k,v) Naive Bayes classification, and the results of that classification are sent to the new version of every word (except the q class which only sends info to one word, not all of them).

It's also useful to think of Attention as a filter with input signal $(e^t)^{[d],[\ell]}$ and output signal $(e^{t+1})^{[d],[\ell]}$.

Fig.95.1 can be "folded" (i.e., the 3 words can be represented by as single node). When folded, Fig.95.1 becomes Fig.95.2. Note that in Fig.95.2, we have started indicating the shapes of tensors by a superscript, using the tensor notation explained in Section C.49. We will continue doing this henceforth in this chapter.

The structural equations for Fig.95.2, printed in blue, are as follows.

$$(a^t)^{[d],[\ell]} = \text{Attention}((v^t)^{[d],[\ell]}, (k^t)^{[d],[\ell]}, (q^t)^{[d],[\ell]}) \tag{95.18a}$$

from getting too large.

[3]Variations of this definition of Attention have been proposed. This particular one is the original one from the "Attention is all you need paper". Some people call it the "scaled dot product Attention".

[4]Tree Augmented Naive (TAN) Bayes nets were introduced in Chapter 9.

[5]A **reverse or upside down tree** is obtained by reversing the directions of all the arrows of a tree directed graph. A TAN Bayes net is normally defined as in Chapter9, as a Naive Bayes net augmented with a tree. In an Attention bnet, the Naive Bayes Net is augmented with a reverse tree (RT) instead of a tree (T), so technically Attention bnets contain RTAN Bayes nets, not TAN Bayes nets.

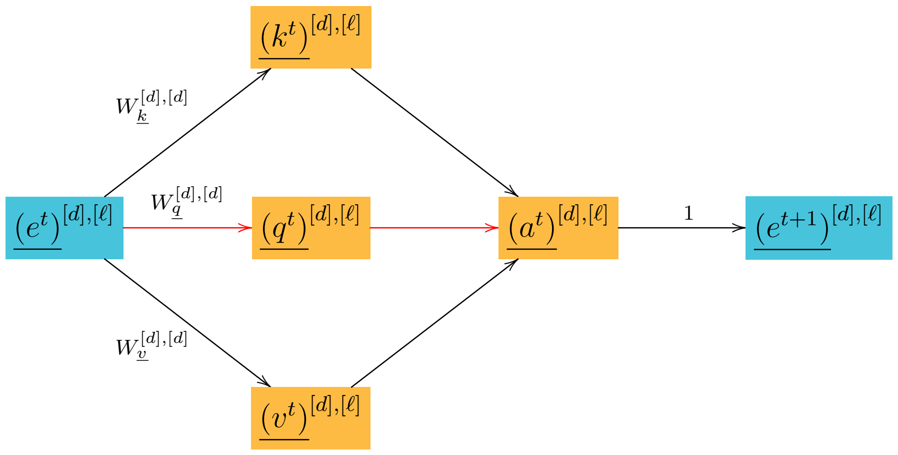

$\underline{(k^t)}^{[d],[\ell]}$
$W_{\underline{k}}^{[d],[d]}$
$W_{\underline{q}}^{[d],[d]}$
$\underline{(e^t)}^{[d],[\ell]}$
$\underline{(q^t)}^{[d],[\ell]}$
$\underline{(a^t)}^{[d],[\ell]}$
1
$\underline{(e^{t+1})}^{[d],[\ell]}$
$W_{\underline{v}}^{[d],[d]}$
$\underline{(v^t)}^{[d],[\ell]}$

Figure 95.2: Folded version of Fig.95.1 when $\ell = 3$. Note that all orange nodes have the same tensor shape.

$$(e^t)^{[d],[\ell]} = \text{prior} \tag{95.18b}$$

$$(e^{t+1})^{[d],[\ell]} = (a^t)^{[d],[\ell]} \tag{95.18c}$$

$$(k^t)^{[d],[\ell]} = W_{\underline{k}}^{[d],[d]}(e^t)^{[d],[\ell]} \tag{95.18d}$$

$$(q^t)^{[d],[\ell]} = W_{\underline{q}}^{[d],[d]}(e^t)^{[d],[\ell]} \tag{95.18e}$$

$$(v^t)^{[d],[\ell]} = W_{\underline{v}}^{[d],[d]}(e^t)^{[d],[\ell]} \tag{95.18f}$$

95.2.2 Multi-Head Attention

In this section, we will generalize the single head Attention, as defined in the previous section, to multi-head Attention.

Let
$n_{\underline{h}}$ = number of heads. $\nu \in [n_{\underline{h}}]$.
d = same as before, the hidden, embedding dimension. $\delta \in [d]$

$D = n_{\underline{h}} d$. $\Delta \in [D]$. We will do some tensor reshaping: $T^{[n_{\underline{h}}],[d]} \to T^{[D]}$, or, in component form, $T^{\nu,\delta} \to T^{\Delta}$.

Consider weight matrices $W_{\underline{k}}^{[D],[d]}$, $W_{\underline{q}}^{[D],[d]}$, and $W_{\underline{v}}^{[D],[d]}$ such that

$$(k^t)^{\nu,\delta,\alpha} = \sum_{\delta' \in [d]} W_{\underline{k}}^{\nu,\delta,\delta'} (e^t)^{\delta',\alpha} \tag{95.19}$$

$$(q^t)^{\nu,\delta,\alpha} = \sum_{\delta' \in [d]} W_{\underline{q}}^{\nu,\delta,\delta'} (e^t)^{\delta',\alpha} \tag{95.20}$$

$$(v^t)^{\nu,\delta,\alpha} = \sum_{\delta' \in [d]} W_{\underline{v}}^{\nu,\delta,\delta'} (e^t)^{\delta',\alpha} \tag{95.21}$$

We define the **Multi-head Attention** by

$$\boxed{\text{Attention}^{\nu,\delta,\alpha}\left((v^t)^{[D],[\ell]}, (k^t)^{[D],[\ell]}, (q^t)^{[D],[\ell]}\right) = \sum_{\alpha' \in [\ell]} (v^t)^{\nu,\delta,\alpha'} P(\alpha'|\alpha,\nu)} \tag{95.22}$$

where

$$\begin{aligned} P(\alpha'|\alpha,\nu) &= \text{softmax}\left[\frac{1}{\sqrt{d}} \sum_{\delta \in [d]} (k^t)^{\nu,\delta,[\ell]} (q^t)^{\nu,\delta,\alpha}\right](\alpha'|\alpha,\nu) & (95.23) \\ &= \frac{\exp\left[\frac{1}{\sqrt{d}} \sum_{\delta\in[d]} (k^t)^{\nu,\delta,\alpha'} (q^t)^{\nu,\delta,\alpha}\right]}{\sum_{\alpha''\in[\ell]} \exp\left[\frac{1}{\sqrt{d}} \sum_{\delta\in[d]} (k^t)^{\nu,\delta,\alpha''} (q^t)^{\nu,\delta,\alpha}\right]} & (95.24) \end{aligned}$$

The structural equations, printed in blue, for the bnet Fig.95.3, are as follows. Note that Attention() always has the same tensor shape as its 3 arguments. Note also that the 3 weight matrices $W_{\underline{k}}^{[D],[d]}$, $W_{\underline{q}}^{[D],[d]}$, and $W_{\underline{v}}^{[D],[d]}$ raise the hidden dimension, whereas the weight matrix $W_{\underline{a}}^{[d],[D]}$ lowers it. $W_{\underline{a}}^{[d],[D]} = 1$ in the single head case.

$$(a^t)^{[D],[\ell]} = \text{Attention}((v^t)^{[D],[\ell]}, (k^t)^{[D],[\ell]}, (q^t)^{[D],[\ell]}) \tag{95.25a}$$

$$(e^t)^{[d],[\ell]} = \text{prior} \tag{95.25b}$$

$$(e^{t+1})^{[d],[\ell]} = W_{\underline{a}}^{[d],[D]} (a^t)^{[D],[\ell]} \tag{95.25c}$$

$$(k^t)^{[D],[\ell]} = W_{\underline{k}}^{[D],[d]} (e^t)^{[d],[\ell]} \tag{95.25d}$$

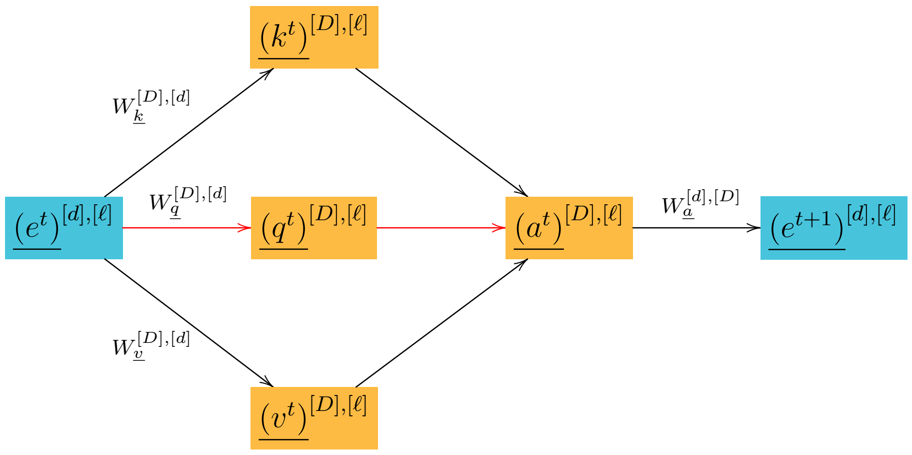

$(\underline{k}^t)^{[D],[\ell]}$
$W_{\underline{k}}^{[D],[d]}$
$(\underline{e}^t)^{[d],[\ell]}$
$W_{\underline{q}}^{[D],[d]}$
$(\underline{q}^t)^{[D],[\ell]}$
$(\underline{a}^t)^{[D],[\ell]}$
$W_{\underline{a}}^{[d],[D]}$
$(\underline{e}^{t+1})^{[d],[\ell]}$
$W_{\underline{v}}^{[D],[d]}$
$(\underline{v}^t)^{[D],[\ell]}$

Figure 95.3: Dynamical bnet with single-head Attention for ℓ words. Time-slice t. This is a generalization of the single head Attention of Fig.95.2. Note that all orange nodes have the same tensor shape.

$$(q^t)^{[D],[\ell]} = W_{\underline{q}}^{[D],[d]}(e^t)^{[d],[\ell]} \tag{95.25e}$$

$$(v^t)^{[D],[\ell]} = W_{\underline{v}}^{[D],[d]}(e^t)^{[d],[\ell]} \tag{95.25f}$$

95.3 Vanilla tranet

In this section, we will discuss the tranet of the "Attention is all you need" paper, Ref.[81]. As is common in the literature, we will refer to that tranet as the "Vanilla" tranet. Ref.[81] describes its tranet graphically with Fig.95.4. Our goal is to find a causal DAG (bnet) version of that figure.

Let

$\ell =$, context window, maximum number of words in a sentence segment. $\alpha \in [\ell]$, $\ell \sim 100$

$L =$ number of words in vocabulary, $\beta \in [L]$, $L >> \ell$

$d = d_{\underline{q}} = d_{\underline{k}} = d_{\underline{v}} = 64$, hidden dimension per head, $\delta \in [d]$.

$n_{\underline{h}} = 8$, number of heads, $\nu \in [n_{\underline{h}}]$

$D = n_{\underline{h}} d = 8(64) = 512$, hidden dimension for all heads, $\Delta \in [D]$

$\Lambda = 6$, number copies, connected in series, of boxed bnet, $\lambda \in [\Lambda]$

Before we present the bnet version of Fig.95.4, we discuss some of the definitions needed to understand and motivate Fig.95.4.

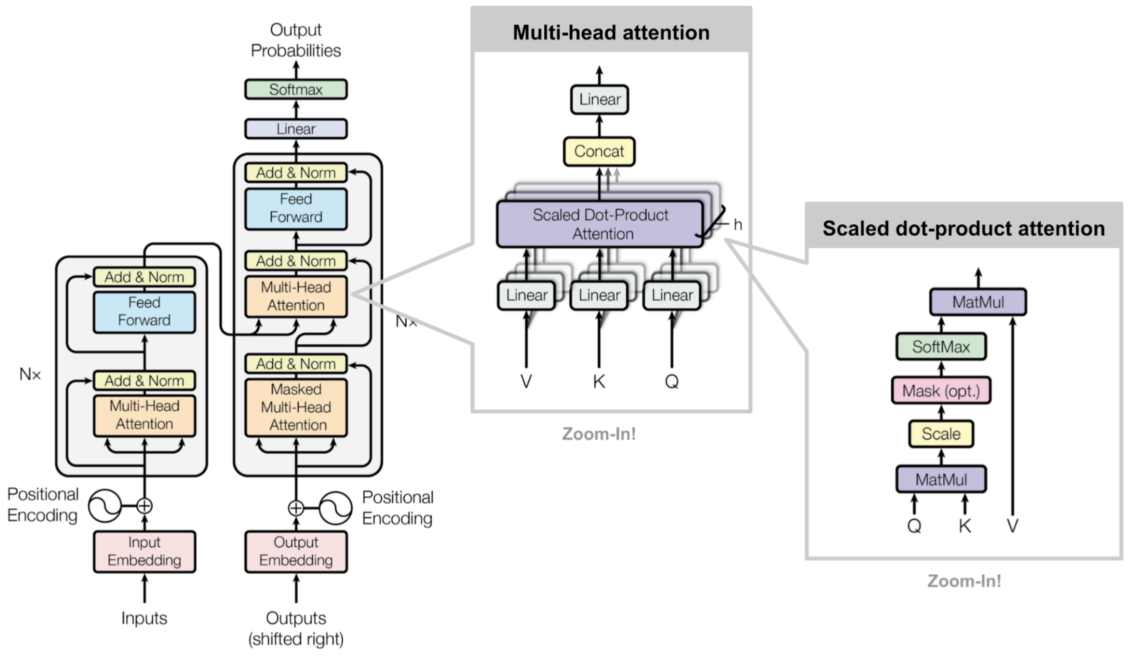


Figure 95.4: Vanilla tranet

- **Encoder Input** $x^{\beta,\alpha}$

$$x^{\beta,\alpha} = \delta(\beta, \beta(\alpha)) \left(x^{[L],[\ell]} \text{ has one hot columns.}\right) \tag{95.26}$$

- **Embedding (a.k.a. encoding) Matrix** $\mathcal{E}^{\delta,\beta}$

$$e^{\delta,\alpha} = \sum_{\beta} \mathcal{E}^{\delta,\beta} x^{\beta,\alpha} \quad \left(e^{[d],[\ell]} = \mathcal{E}^{[d],[L]} x^{[L],[\ell]}\right) \tag{95.27}$$

- **Weight matrices** $W_{\underline{q}}, W_{\underline{k}}, W_{\underline{v}}$

$$Q^{\nu,\delta,\alpha} = \sum_{\delta'} W_{\underline{q}}^{\nu,\delta,\delta'} e^{\delta',\alpha} \quad \left(Q^{[D],[\ell]} = W_{\underline{q}}^{[D],[d]} e^{[d],[\ell]}\right) \tag{95.28}$$

$$K^{\nu,\delta,\alpha} = \sum_{\delta'} W_{\underline{k}}^{\nu,\delta,\delta'} e^{\delta',\alpha} \quad \left(K^{[D],[\ell]} = W_{\underline{k}}^{[D],[d]} e^{[d],[\ell]}\right) \tag{95.29}$$

$$V^{\nu,\delta,\alpha} = \sum_{\delta'} W_{\underline{v}}^{\nu,\delta,\delta'} e^{\delta',\alpha} \quad \left(V^{[D],[\ell]} = W_{\underline{v}}^{[D],[d]} e^{[d],[\ell]}\right) \tag{95.30}$$

- **Multi-head Attention**

$$B^{\nu,\alpha',\alpha} = \frac{1}{\sqrt{d}}\sum_{\delta} K^{\nu,\delta,\alpha'} Q^{\nu,\delta,\alpha} \quad \left(B^{\nu,\alpha',\alpha} = \frac{1}{\sqrt{d}}(K^{\nu,[d],\alpha'})^T Q^{\nu,[d],\alpha}\right) \tag{95.31}$$

$$A^{\nu,\delta,\alpha} \;=\; \sum_{\alpha'} V^{\nu,\delta,\alpha'} \underbrace{\text{softmax}(B^{\nu,[\ell],\alpha})(\alpha'|\alpha,\nu)}_{P(\alpha'|\alpha,\nu)} \tag{95.32}$$

$$\sum_{\alpha'\in[\ell]} P(\alpha'|\alpha,\nu) = 1 \tag{95.33}$$

$$A^{\nu,\delta,\alpha} \to A^{\Delta,\alpha} \left(A^{[n_{\underline{h}}],[d],[\ell]} \to A^{[D],[\ell]}\right) \tag{95.34}$$

Important: Note that the softmax() makes the α' component a probability, not the α one!

For example, suppose $\nu = 1$ (one head), $\ell = 2$ (a 2 word segment), and $d = 3$ (hidden dimension is 3). The $Q^{[3],[2]}, K^{[3],[2]}, V^{[3],[2]}$ are 3×2 matrices (i.e., two 3-dim column vectors). One uses the $Q^{[3],[2]}$ and $K^{[3],[2]}$ to arrive at a 2×2 matrix $P(\alpha'|\alpha)$ of probabilities. Then one uses that matrix of probabilities to replace

$$\left[V^{[3],0}, V^{[3],1}\right] \to \left[V^{[3],0}P(0|0) + V^{[3],1}P(1|0), V^{[3],0}P(0|1) + V^{[3],1}P(1|1)\right] \tag{95.35}$$

- **Positional Embedding Matrix $\mathcal{E}^{\delta,\beta}_{pos}$**

$$\mathcal{E}^{\delta,\beta}_{pos} = \begin{cases} \sin\left(2\pi\frac{\beta}{(2\pi)10^{4\delta/d}}\right) = \sin(2\pi\frac{\beta}{\lambda(\delta)}) & \text{if } \delta \text{ is even} \\ \cos\left(2\pi\frac{\beta}{(2\pi)10^{4(\delta-1)/d}}\right) = \cos(2\pi\frac{\beta}{\lambda(\delta)}) & \text{if } \delta \text{ is odd} \end{cases} \tag{95.36}$$

$\mathcal{E}^{\delta,\beta}_{pos}$ changes in phase by $\pi/2$ every time δ changes by 1. Its wavelength λ is independent of β, but increases rapidly with δ, from $\lambda(\delta = 0) = 2\pi * 1$ to $\lambda(\delta = d) = 2\pi * 10^4$.

Total Embedding equals initial embedding plus positional embedding:

$$\mathcal{E}^{\delta,\beta} = \mathcal{E}^{\delta,\beta}_0 + \mathcal{E}^{\delta,\beta}_{pos} \tag{95.37}$$

The purpose of positional embedding is to take $e^{\beta,\alpha}$ to $e^{\delta,\alpha} = \sum_\beta \mathcal{E}^{\delta,\beta}_{pos} e^{\beta,\alpha}$ where $e^{\delta,\alpha}$ changes quickly as δ (i.e., position) changes.

- **ReLU**

  For a tensor T of arbitrary shape,

$$ReLU(T) = (T)_+ = max(0, T) \tag{95.38}$$

  max element-wise.

- **Feed Forward Neural Net**

$$F(e^{\delta,\alpha}) = \sum_{\Delta\in[n_{ff}]} W_2^{\delta,\Delta} ReLU\left(\sum_{\delta'\in[d]} W_1^{\Delta,\delta'} e^{\delta',\alpha} + b_1^{\Delta,\alpha}\right) + b_2^{\delta,\alpha} \tag{95.39}$$

  n_{ff} is called the `intermediate_size` in BERT.

- **Softmax**

  softmax() takes a vector and returns a vector of probabilities of the same length

$$e^{[n]} \to P^{[n]} \tag{95.40}$$

  where

$$P^\alpha = \frac{\exp(e^\alpha)}{\sum_{\alpha\in[n]} \exp(e^\alpha)} \quad \left(P^{[n]} = \frac{\exp(e^{[n]})}{\| \exp(e^{[n]}) \|_0}\right) \tag{95.41}$$

  For example,

$$(1, 0, 0) \to (e, 1, 1)/norm \tag{95.42}$$

$$(10, 0, 0) \to (e^{10}, 1, 1)/norm \approx (1, 0, 0) \tag{95.43}$$

  For any $a \in \mathbb{R}$,

$$(a, a, a) \to \frac{1}{3}(1, 1, 1) \tag{95.44}$$

- **Skip Connection (Add & Normalize)**

  A **skip connection** is when you split the input to a **filter** into two streams, one stream goes through the filter, the other doesn't. The one that doesn't is then merged with the output of the filter via a **add & normalize** node. The reason for making skip connections is that the signal exiting a filter is usually full of jumps and kinks. By merging that filter output with some of the filter input, one smooths out the filter output to some degree. This makes back-propagation differentiation better behaved.

  The filter might be a Multi-Head Attention or a Feed Forward NN.

Add & Normalize just means $(A+B)/norm$ where A and B are the two input signals and "norm" is some norm of $A+B$ (for instance, $\| A+B \|_2$).

Normalization keeps the signal from growing too big and saturating the signal that will enter components upstream. Normalization can also involve subtracting the mean $\langle X\rangle$ of the signal X so as to get a signal $X - \langle X\rangle$ that has zero mean.

- **Redundancy**

  For better results, the Encoder and Decoder both contain Λ copies, connected in series, of the boxed bnet.

  Redundancy (see Chapter 79) has been used to avoid catastrophic failure at least as early as the dawn of the age of rocketry, when it was used to avoid the all too common occurrence of exploding rockets. There are 2 basic types of redundancy: in series connection (as in the repeated identical layers in a feedforward NN or a recurrent NN), and in parallel connection (as in tranet heads, and the plates in a bnet (see Chapter 71)).

- **Right Shifted Outputs**

  "Outputs (Shifted Right)" in Fig.95.4 refers to what is called **forced teaching** in the RNN (recurrent neural net) literature. We explain forced teaching in Fig.95.5.

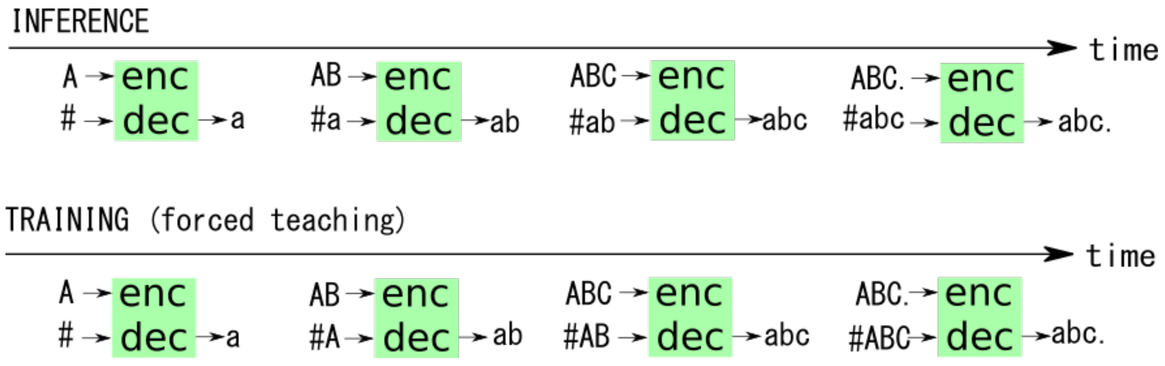


Figure 95.5: Training and Inference for vanilla transformer. "enc" and "dec" denote the encoder and decoder, respectively. A hash character represents the SOS (start of sentence) token, and a period represents the EOS (end of sentence) token. Capital letters represent ground truth tokens, and lower case ones represent predictions.

- **Masked Attention**

$$P(\alpha'|\alpha,\nu) = 0 \quad \text{if } \alpha' < \alpha \tag{95.45}$$

α, and α' are word positions in a sentence, and α' is in the future (downstream) compared to α. So as to not violate causality, this condition enforces the constraint that no attention is paid to word positions in the future of α.

95.3.1 Single Head Attention

Fig.95.6 gives a bnet representation of the "Single Head Attention" portion of Fig.95.4. The structural equations for that bnet, printed in blue, are as follows.

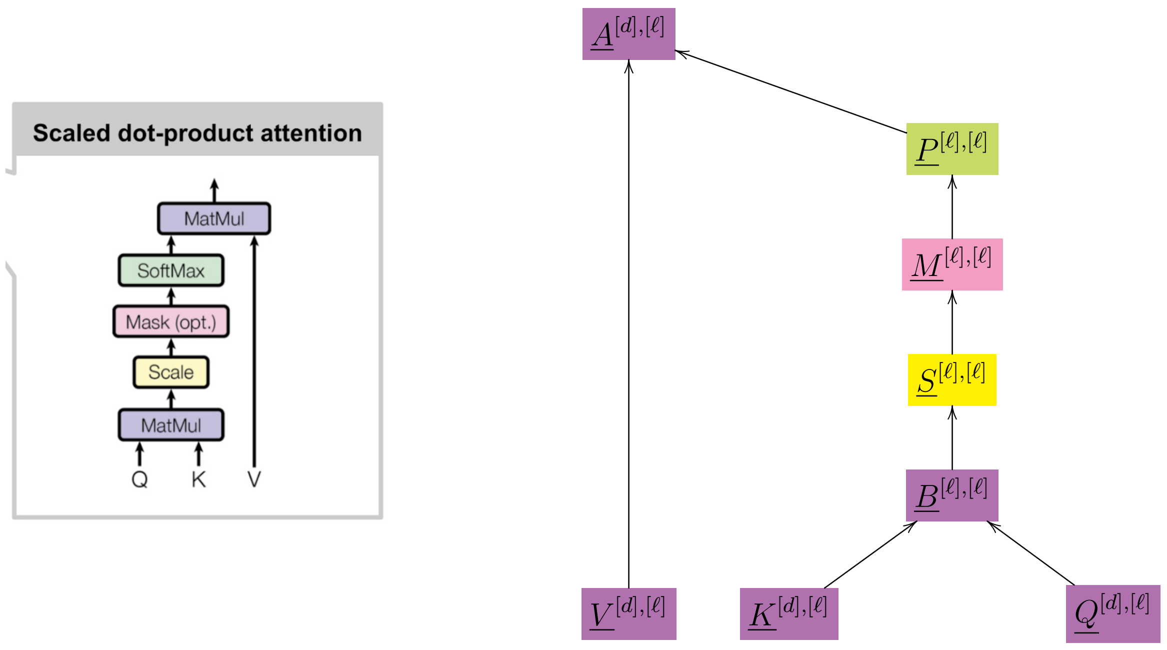

Scaled dot-product attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
$\underline{A}^{[d],[\ell]}$
$\underline{P}^{[\ell],[\ell]}$
$\underline{M}^{[\ell],[\ell]}$
$\underline{S}^{[\ell],[\ell]}$
$\underline{B}^{[\ell],[\ell]}$
$\underline{V}^{[d],[\ell]}$
$\underline{K}^{[d],[\ell]}$
$\underline{Q}^{[d],[\ell]}$

Figure 95.6: Single Head Attention. (Scaled Dot Product)

$$A^{[d],[\ell]} = V^{[d],[\ell]} P^{[\ell],[\ell]} \quad \left(\text{Note that } \sum_{\alpha \in [\ell]} P^{\alpha,[\ell]} = 1 \right) \tag{95.46a}$$

$$B^{[\ell],[\ell]} = (K^{[d],[\ell]})^T Q^{[d],[\ell]} \tag{95.46b}$$

$$K^{[d],[\ell]} = \text{prior} \tag{95.46c}$$

$$M^{[\ell],[\ell]} = \text{mask}(S^{[\ell],[\ell]}) \tag{95.46d}$$

$$P^{[\ell],[\ell]} = \text{softmax}(M^{[\ell],[\ell]}) \quad \left(\text{Note that } \sum_{\alpha\in[\ell]} P^{\alpha,[\ell]} = 1\right) \tag{95.46e}$$

$$Q^{[d],[\ell]} = \text{prior} \tag{95.46f}$$

$$S^{[\ell],[\ell]} = \frac{B^{[\ell],[\ell]}}{\sqrt{d}} \tag{95.46g}$$

$$V^{[d],[\ell]} = \text{prior} \tag{95.46h}$$

95.3.2 Multi-Head Attention

Fig.95.7 gives a bnet representation of the "Multi-Head Attention" portion of Fig.95.4. The structural equations for that bnet, printed in blue, are as follows.

$$A^{[D],[\ell]} = [A_0^{[d],[\ell]} | A_1^{[d],[\ell]}] \tag{95.47a}$$

$$A_0^{[d],[\ell]} = \text{Attention}(V_0^{[d],[\ell]}, K_0^{[d],[\ell]}, Q_0^{[d],[\ell]}) \tag{95.47b}$$

$$A_1^{[d],[\ell]} = \text{Attention}(V_1^{[d],[\ell]}, K_1^{[d],[\ell]}, Q_1^{[d],[\ell]}) \tag{95.47c}$$

$$K^{[D],[\ell]} = W_{\underline{k}}^{[D],[d]} e^{[d],[\ell]} \tag{95.47d}$$

$$K_0^{[d],[\ell]} = \text{linear}(K^{[D],[\ell]}) \ \text{(split, then project a component)} \tag{95.47e}$$

$$K_1^{[d],[\ell]} = \text{linear}(K^{[D],[\ell]}) \ \text{(split, then project a component)} \tag{95.47f}$$

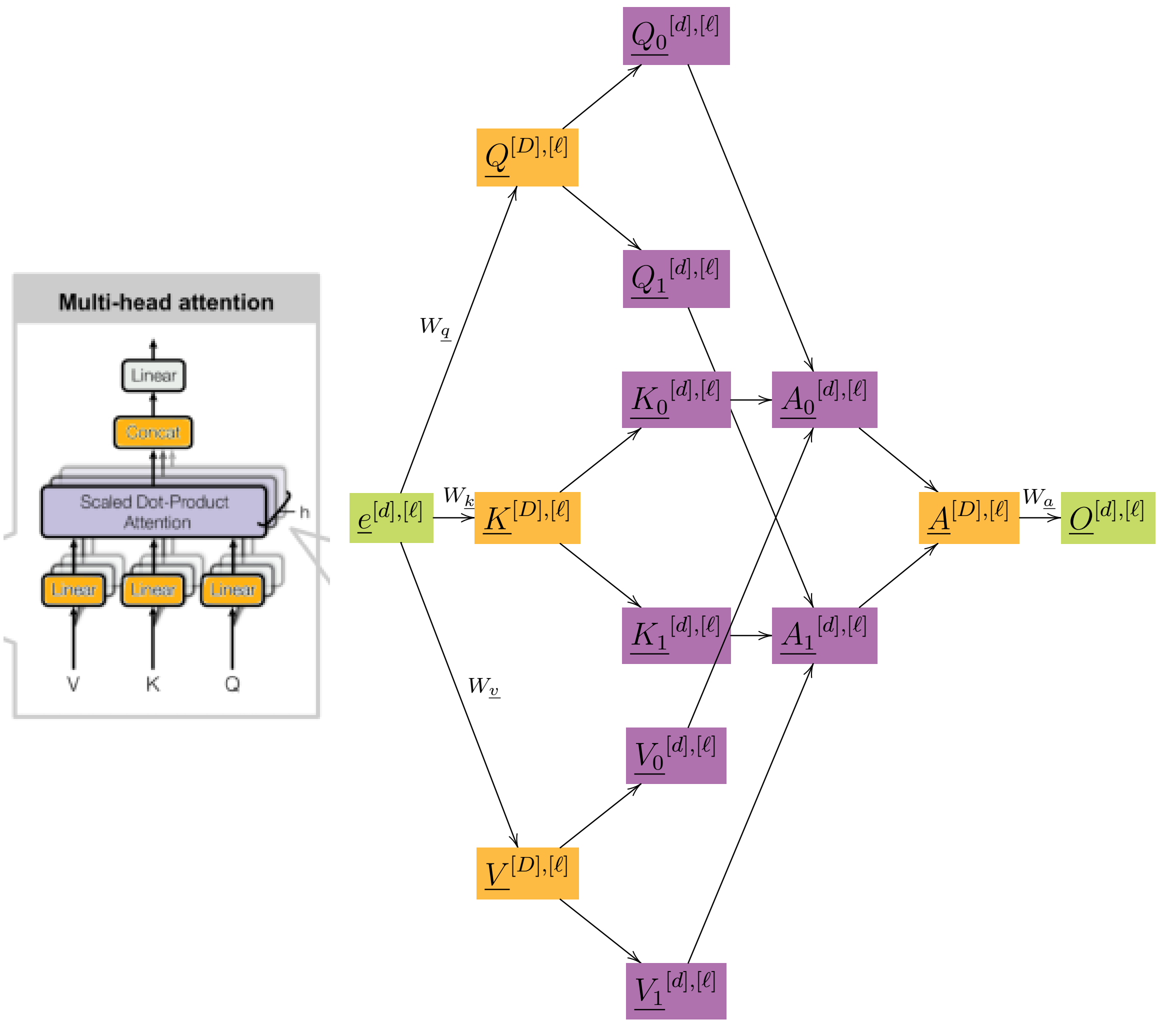

Multi-head attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q
$\underline{e}^{[d],[\ell]}$
$W_{\underline{q}}$
$W_{\underline{k}}$
$W_{\underline{v}}$
$\underline{Q}^{[D],[\ell]}$
$\underline{K}^{[D],[\ell]}$
$\underline{V}^{[D],[\ell]}$
$\underline{Q_0}^{[d],[\ell]}$
$\underline{Q_1}^{[d],[\ell]}$
$\underline{K_0}^{[d],[\ell]}$
$\underline{K_1}^{[d],[\ell]}$
$\underline{V_0}^{[d],[\ell]}$
$\underline{V_1}^{[d],[\ell]}$
$\underline{A_0}^{[d],[\ell]}$
$\underline{A_1}^{[d],[\ell]}$
$\underline{A}^{[D],[\ell]}$
$W_{\underline{a}}$
$\underline{O}^{[d],[\ell]}$

Figure 95.7: Multi-head Attention with 2 heads. Note that the orange nodes all have the same tensor shape.

$$O^{[d],[\ell]} = W_{\underline{a}}^{[d],[D]} A^{[D],[\ell]} \tag{95.47g}$$

$$Q^{[D],[\ell]} = W_{\underline{q}}^{[D],[d]} e^{[d],[\ell]} \tag{95.47h}$$

$$Q_0^{[d],[\ell]} = \text{linear}(Q^{[D],[\ell]}) \;\; \text{(split, then project a component)} \tag{95.47i}$$

$$Q_1^{[d],[\ell]} = \text{linear}(Q^{[D],[\ell]}) \ \text{(split, then project a component)} \tag{95.47j}$$

$$V^{[D],[\ell]} = W_{\underline{v}}^{[D],[d]} e^{[d],[\ell]} \tag{95.47k}$$

$$V_0^{[d],[\ell]} = \text{linear}(V^{[D],[\ell]}) \ \text{(split, then project a component)} \tag{95.47l}$$

$$V_1^{[d],[\ell]} = \text{linear}(V^{[D],[\ell]}) \ \text{(split, then project a component)} \tag{95.47m}$$

$$e^{[d],[\ell]} = \ \text{prior} \tag{95.47n}$$

95.3.3 Encoder

Fig.95.8 gives a bnet representation of the “Encoder" portion of Fig.95.4. The structural equations for that bnet, printed in blue, are as follows.

$$A^{[D],[\ell]} = \text{Attention}(Q^{[D],[\ell]}, K^{[D],[\ell]}, V^{[D],[\ell]}) \tag{95.48a}$$

$$e^{[d],[\ell]} = \mathcal{E}^{[d],[L]} x^{[L],[\ell]} \tag{95.48b}$$

$$F^{[d],[\ell]} = \text{feed\_forward\_nn}(N^{[d],[\ell]}) \tag{95.48c}$$

$$K^{[D],[\ell]} = W_{\underline{k}}^{[D],[d]} e^{[d],[\ell]} \tag{95.48d}$$

$$n^{[d],[\ell]} = \text{normalize}(N^{[d],[\ell]} + F^{[d],[\ell]}) \tag{95.48e}$$

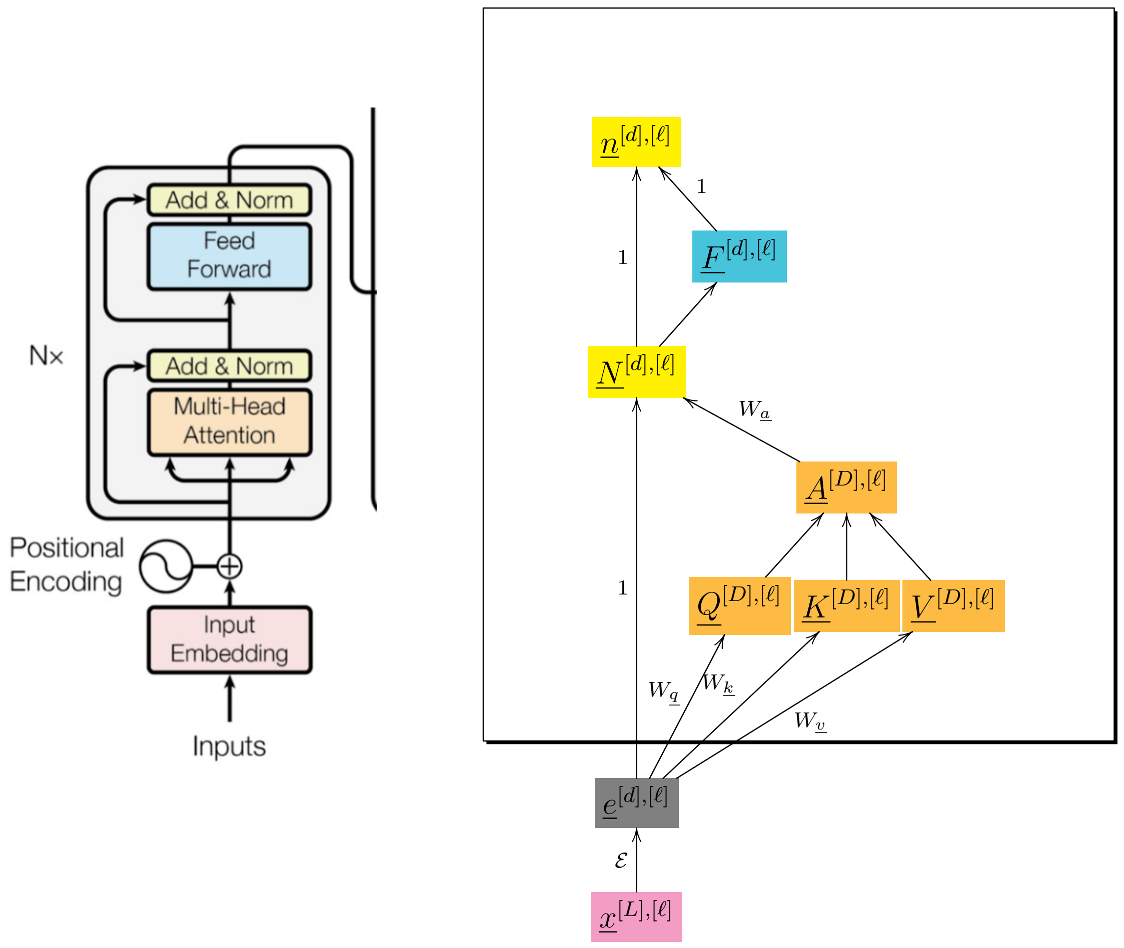

Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
$\underline{n}^{[d],[\ell]}$
1
1
$\underline{F}^{[d],[\ell]}$
$\underline{N}^{[d],[\ell]}$
$W_{\underline{a}}$
$\underline{A}^{[D],[\ell]}$
1
$\underline{Q}^{[D],[\ell]}$
$\underline{K}^{[D],[\ell]}$
$\underline{V}^{[D],[\ell]}$
$W_{\underline{q}}$
$W_{\underline{k}}$
$W_{\underline{v}}$
$\underline{e}^{[d],[\ell]}$
$\mathcal{E}$
$\underline{x}^{[L],[\ell]}$

Figure 95.8: Encoder of Vanilla Transformer Net. Λ copies of the boxed part are connected in series.

$$N^{[d],[\ell]} = \text{normalize}(e^{[d],[\ell]} + W_{\underline{a}}^{[d],[D]} A^{[D],[\ell]}) \tag{95.48f}$$

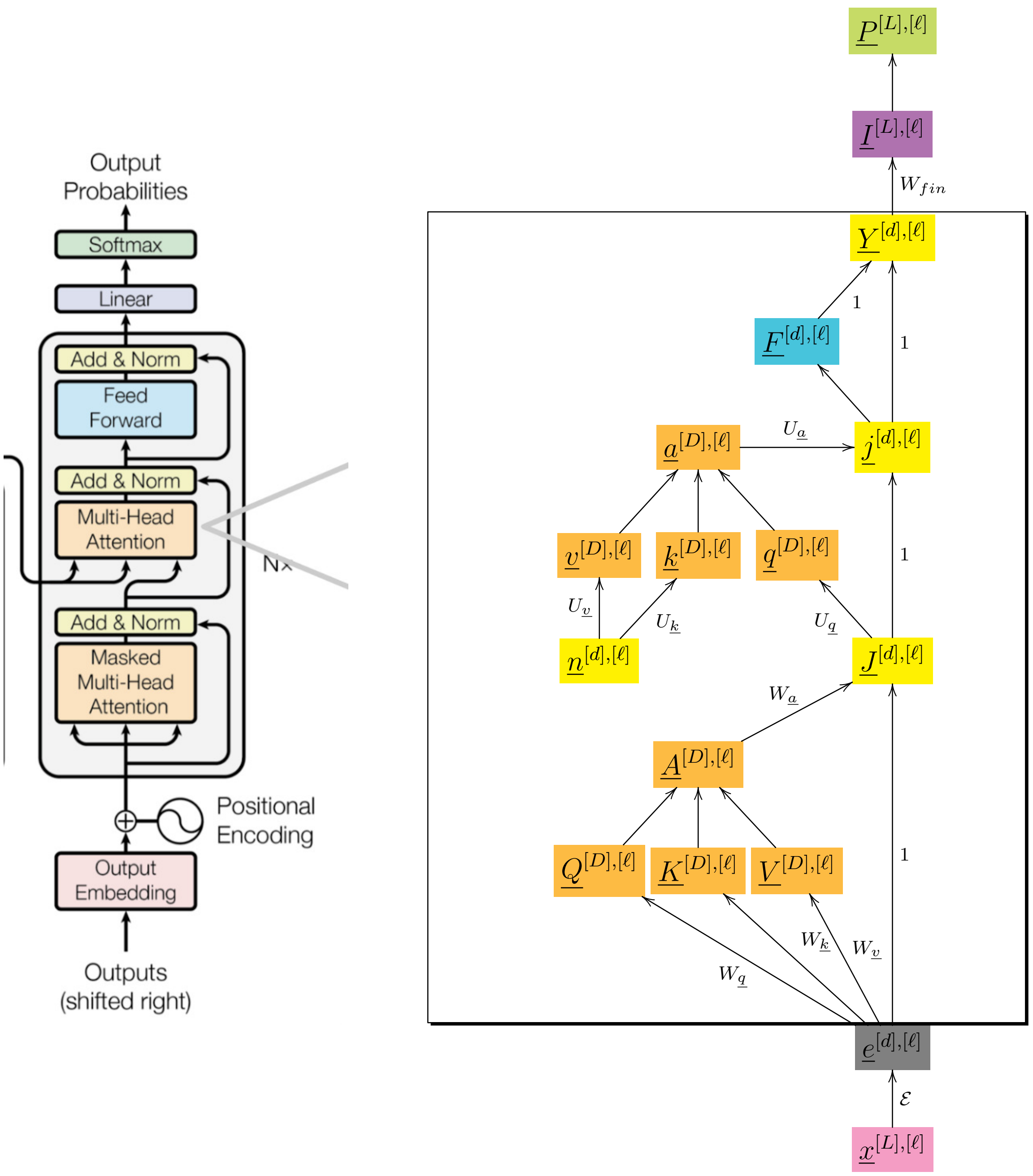
$$Q^{[D],[\ell]} = W_{\underline{q}}^{[D],[d]} e^{[d],[\ell]} \tag{95.48g}$$

$$V^{[D],[\ell]} = W_{\underline{v}}^{[D],[d]} e^{[d],[\ell]} \tag{95.48h}$$

$$x^{[L],[\ell]} = \quad \text{prior} \tag{95.48i}$$

95.3.4 Decoder

Fig.95.9 gives a bnet representation of the “Decoder" portion of Fig.95.4. The structural equations for that bnet, printed in blue, are as follows.



Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
N×
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs (shifted right)
$\underline{P}^{[L],[\ell]}$
$\underline{I}^{[L],[\ell]}$
W_{fin}
$\underline{Y}^{[d],[\ell]}$
1
$\underline{F}^{[d],[\ell]}$
1
$\underline{a}^{[D],[\ell]}$
$U_{\underline{a}}$
$\underline{j}^{[d],[\ell]}$
$\underline{v}^{[D],[\ell]}$
$\underline{k}^{[D],[\ell]}$
$\underline{q}^{[D],[\ell]}$
1
$U_{\underline{v}}$
$U_{\underline{k}}$
$U_{\underline{q}}$
$\underline{n}^{[d],[\ell]}$
$\underline{J}^{[d],[\ell]}$
$W_{\underline{a}}$
$\underline{A}^{[D],[\ell]}$
$\underline{Q}^{[D],[\ell]}$
$\underline{K}^{[D],[\ell]}$
$\underline{V}^{[D],[\ell]}$
1
$W_{\underline{k}}$
$W_{\underline{v}}$
$W_{\underline{q}}$
$\underline{e}^{[d],[\ell]}$
$\mathcal{E}$
$\underline{x}^{[L],[\ell]}$

Figure 95.9: Decoder of Vanilla Transformer Net. Λ copies of the boxed part are connected in series.

$$a^{[D],[\ell]} = \text{Attention}(v^{[D],[\ell]}, k^{[D],[\ell]}, q^{[D],[\ell]}) \tag{95.49a}$$

$$A^{[D],[\ell]} = \text{Attention}(Q^{[D],[\ell]}, K^{[D],[\ell]}, V^{[D],[\ell]}) \tag{95.49b}$$

$$e^{[d],[\ell]} = \mathcal{E}^{[d],[L]} x^{[L],[\ell]} \tag{95.49c}$$

$$F^{[d],[\ell]} = \text{feed\_forward\_nn}(j^{[d],[\ell]}) \tag{95.49d}$$

$$I^{[L],[\ell]} = W_{fin}^{[L],[d]} Y^{[d],[\ell]} \tag{95.49e}$$

$$j^{[d],[\ell]} = \text{normalize}(U_{\underline{a}}^{[d],[D]} a^{[D],[\ell]} + J^{[d],[\ell]}) \tag{95.49f}$$

$$J^{[d],[\ell]} = \text{normalize}(W_{\underline{a}}^{[d],[D]} A^{[D],[\ell]} + e^{[d],[\ell]}) \tag{95.49g}$$

$$K^{[D],[\ell]} = W_{\underline{k}}^{[D],[d]} e^{[d],[\ell]} \tag{95.49h}$$

$$k^{[D],[\ell]} = U_{\underline{k}}^{[D],[d]} n^{[d],[\ell]} \tag{95.49i}$$

$$n^{[d],[\ell]} = \quad \text{Prior coming from Encoder.} \tag{95.49j}$$

$$P^{[L],[\ell]} = \text{softmax}(I^{[L],[\ell]}) \;\; (\sum_{\alpha \in [\ell]} P^{[L],\alpha} = 1) \tag{95.49k}$$

$$q^{[D],[\ell]} = U_{\underline{q}}^{[D],[d]} J^{[d],[\ell]} \tag{95.49l}$$

$$Q^{[D],[\ell]} = W_{\underline{q}}^{[D],[d]} e^{[d],[\ell]} \tag{95.49m}$$

$$V^{[D],[\ell]} = W_{\underline{v}}^{[D],[d]} e^{[d],[\ell]} \tag{95.49n}$$

$$v^{[D],[\ell]} = U_{\underline{v}}^{[D],[d]} n^{[d],[\ell]} \tag{95.49o}$$

$$x^{[L],[\ell]} = \text{ prior, right shifted output} \tag{95.49p}$$

$$Y^{[d],[\ell]} = \text{normalize}(F^{[d],[\ell]} + J^{[d],[\ell]}) \tag{95.49q}$$

95.4 BERT

I used the the Wikipedia article on BERT, Ref[94] to write this section.

BERT (Bidirectional Encoder Representations from Transformer) is a realization of the Encoder half of the Vanilla tranet. One can either add a smaller NN to the output of BERT (this is called **fine-tuning**), or one can add a de-embedding layer to its output so that the total device takes word lists to word lists.

In the language of Bayesian Networks, fine-tuning is the same as using BERT as a prior probability. See Chapter 84 on sentence splitting for an example of BERT fine-tuning.

95.4.1 BERT parameter values

BERT comes in two sizes, base and large. See Table 95.1 for a listing of some BERT parameter values.

	BERT base	BERT large
ℓ, context window	512	512
L, `vocab_size`	30,522	30,522
d, `hidden_size`	768	1024
$n_{\underline{h}}$, `num_attention_heads`	12	16
Λ, `num_hidden_layers`	12	24
D', `intermediate_size`	3,072	3,072
number of parameters	110M	340M

Table 95.1: Some hyper-parameter values for BERT base and BERT large

95.4.2 BERT Embedding

So far, we have described the embedding step as a single step from tokenization into words, to 1 hot vectors, to embedding vectors. There are other additional steps in the embedding process that we haven't described so far (namely, tokenization into subwords, adding special tokens, and padding). We would like to describe those additional steps now, in the context of the BERT model. Here is an example.

Let's consider a short sentence: "The cat is on the mat."

1. **Tokenization into words:** Tokenize the sentence into individual words:

   "The", "cat", "is", "on", "the", "mat", "."

2. **Tokenization into subwords:** Further tokenize words into subword units using WordPiece tokenization or a similar method. For example:

$$\begin{aligned}
&The \to The\\
&cat \to ca, t\\
&is \to is\\
&on \to on\\
&the \to the\\
&mat \to mat\\
&. \to .
\end{aligned}$$

   Any subword not appearing in BERTs vocabulary is replaced by [UNK] for "unknown".

3. **Adding Special Tokens:** Add special tokens, such as [CLS] (classification) at the beginning and [SEP] (separator) at the end:

   "[CLS]", "The", "ca", "t", "is", "on", "the", "mat", ".", "[SEP]"

4. **Padding:** If necessary, pad or truncate the sequence to a fixed length. Add padding tokens "[PAD]" to reach a specified sequence length.

5. **Embedding Matrix:** Create a tensor with 1-hot columns

$$x^{\beta,\alpha} = \delta(\beta, \beta(\alpha)) \tag{95.50}$$

   where $\alpha \in [\ell]$, $\beta \in [L]$ and where $\beta(\alpha)$ is the location in the BERT vocab corresponding to token α in the the padded string. Now multiply x times the previously discussed embedding matrix $\mathcal{E}$ to get

$$e^{[d],[\ell]} = \mathcal{E}^{[d],[L]} x^{[L],[\ell]} \in \mathbb{R}^{d\times\ell} \tag{95.51}$$

   This gives a vector in $\mathbb{R}^d$ for each token α in the padded string. The matrix $\mathcal{E}$ is pre-trained and captures contextual information and word similarities. It can also include positional embedding, as discussed before.

95.4.3 BERT training

BERT was trained[6] simultaneously on two tasks.[7]

1. **language modeling**: 15% of tokens were selected for prediction. Those tokens selected for prediction were replaced by the [MASK] token 80% of the time, by a random word 10% of the time, and not replaced at all 10% of the time. The training objective was to predict the selected token given its context.

2. **next sentence prediction:** Given two spans of text, the model predicts if these two spans appeared sequentially in the training corpus, outputting either [IsNext] or [NotNext]. For example,

   - Given "[CLS] my dog is cute [SEP] he likes playing", the model should output token [IsNext].
   - Given "[CLS] my dog is cute [SEP] how do magnets work", the model should output token [NotNext]

[6]Sometimes this is called “pre-training" to distinguish it from the “training" of the smaller NN that is attached to the output of BERT when doing fine-tuning.

[7]This section on BERT training quotes Wikipedia Ref.[94] heavily.

Bibliography

[1] Alan Agresti. *An introduction to categorical data analysis*. John Wiley & Sons, 2018.

[2] Data Analytics and IIT Delhi Intelligence Research (DAIR) Group. Openie6. https://github.com/dair-iitd/openie6.

[3] Elias Bareinboim, Jin Tian, and Judea Pearl. Recovering from selection bias in causal and statistical inference. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. https://ftp.cs.ucla.edu/pub/stat_ser/r425.pdf.

[4] Dan Bendel. Metropolis-Hastings: A comprehensive overview and proof. https://similarweb.engineering/mcmc/.

[5] David Benkeser and Antoine Chambaz. A ride in targeted learning territory. https://achambaz.github.io/tlride/tlride-book.pdf.

[6] Alexandra M Carvalho. Scoring functions for learning Bayesian networks. http://www.lx.it.pt/~asmc/pub/talks/09-TA/ta_pres.pdf.

[7] Bo Chang. Copula: a very short introduction, article in Bo's Blog. https://bochang.me/blog/posts/copula/.

[8] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. https://arxiv.org/abs/1603.02754.

[9] Victor Chernozhukov, Carlos Cinelli, Whitney Newey, Amit Sharma, and Vasilis Syrgkanis. Long story short: Omitted variable bias in causal machine learning. https://www.nber.org/system/files/working_papers/w30302/w30302.pdf.

[10] Carlos Cinelli, Andrew Forney, and Judea Pearl. A crash course in good and bad controls. https://ftp.cs.ucla.edu/pub/stat_ser/r493.pdf.

[11] Carlos Cinelli and Chad Hazlett. Making sense of sensitivity: Extending omitted variable bias. https://carloscinelli.com/files/Cinelli%20and%20Hazlett%20(2020)%20-%20Making%20Sense%20of%20Sensitivity.pdf.

[12] Scott Cunningham. *Causal inference: The mixtape.* Yale University Press, 2021. `https://mixtape.scunning.com/index.html`.

[13] Robin J. Evans. Graphical methods for inequality constraints in marginalized DAGs. `https://arxiv.org/abs/1209.2978`.

[14] Matheus Facure Alves. *Causal Inference for The Brave and True*. 2021. `https://matheusfacure.github.io/python-causality-handbook/landing-page.html`.

[15] George Fei. Modeling uplift directly: Uplift decision tree with kl divergence and euclidean distance as splitting criteria. `https://www.aboutwayfair.com/tech-innovation/modeling-uplift-directly-uplift-decision-tree-with-kl-divergence-and-euclidean-distance-as-splitting-criteria`.

[16] Charles Fox, Neil Girdhar, and Kevin Gurney. A causal Bayesian network view of reinforcement learning. `https://www.aaai.org/Papers/FLAIRS/2008/FLAIRS08-030.pdf"`.

[17] Bruno Gonçalves. Model testing and causal search. blog post `https://medium.com/data-for-science/causal-inference-part-vii-model-testing-and-causal-search-536b796f0384`.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, David Warde-Farley Bing Xu, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. `https://arxiv.org/abs/1406.2661`.

[19] Pierre Gutierrez and Jean-Yves Gérardy. Causal inference and uplift modelling: A review of the literature. In *Proceedings of The 3rd International Conference on Predictive Applications and APIs*, pages 1–13, 2017. `http://proceedings.mlr.press/v67/gutierrez17a.html`.

[20] James Douglas Hamilton. *Time series analysis.* Princeton University Press, 2020.

[21] Sebastian Haneuse and Andrea Rotnitzky. Estimation of the effect of interventions that modify the received treatment. *Statistics in medicine*, 32(30):5260–5277, 2013. Main:`https://sci-hub.se/10.1002/sim.5907`, Supplement: `https://onlinelibrary.wiley.com/action/downloadSupplement?doi=10.1002%2Fsim.5907&file=sim5907-sup-0001-Appendix.pdf`.

[22] Christina Heinze-Deml. Causality, spring semester 2019 at ETH Zurich. `https://stat.ethz.ch/lectures/ss19/causality.php#course_materials`.

[23] MA Hernán and J Robins. Causal inference: What if. Boca Raton: Chapman & Hill/CRC, `https://www.hsph.harvard.edu/miguel-hernan/causal-inference-book/`.

[24] Katherine Hoffman. An illustrated guide to modified treatment policies, part 1: Introduction and motivation, article in KHstats blog. https://www.khstats.com/blog/lmtp/lmtp/.

[25] Katherine Hoffman. An illustrated guide to TMLE, article in KHstats blog. https://www.khstats.com/blog/tmle/tutorial/.

[26] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International journal of approximate reasoning*, 15(3):225–263, 1996. http://www.ar-tiste.com/Huang-Darwiche1996.pdf.

[27] Tommi S. Jaakkola and Michael I. Jordan. Variational probabilistic inference and the QMR-DT network. http://arxiv.org/abs/1105.5462.

[28] Michael I. Jordan. course: Stat260-Bayesian modeling and inference, lecture date: February 8-2010, title: The conjugate prior for the Normal distribution. https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/lectures/lecture5.pdf.

[29] Chaitanya K. Joshi. Transformer (machine learning model). https://graphdeeplearning.github.io/post/transformers-are-gnns/.

[30] Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Mausam, and Soumen Chakrabarti. Openie6: Iterative grid labeling and coordination analysis for open information extraction. https://arxiv.org/abs/2010.03147.

[31] Chung-Ming Kuan. Introduction to time series analysis, Fall 2014 lectures given at the Department of Finance, National Taiwan University. https://homepage.ntu.edu.tw/~ckuan/pdf/2014fall/Lec-TimeSeries_slide-Fall2014.pdf.

[32] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988. http://www.eecis.udel.edu/~shatkay/Course/papers/Lauritzen1988.pdf.

[33] Sergey Levine. Course CS 285 at UC Berkeley, Deep reinforcement learning. http://rail.eecs.berkeley.edu/deeprlcourse/.

[34] Ang Li. Ph.D. Thesis, UCLA 2021. https://ftp.cs.ucla.edu/pub/stat_ser/r507.pdf.

[35] Dimitris Margaritis. Learning Bayesian network model structure from data (thesis, 2003, Carnegie Mellon Univ). https://apps.dtic.mil/sti/citations/ADA461103.

[36] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. In *BMC bioinformatics*, volume 7, page S7. Springer, 2006. https://link.springer.com/article/10.1186/1471-2105-7-S1-S7.

[37] Samuele Mazzanti. Black-box models are actually more explainable than a logistic regression. https://towardsdatascience.com/black-box-models-are-actually-more-explainable-than-a-logistic-regression-f263c22795d.

[38] Samuele Mazzanti. SHAP values explained exactly how you wished someone explained to you. https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30.

[39] Robert J. McEliece, David J. C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. http://authors.library.caltech.edu/6938/1/MCEieeejstc98.pdf.

[40] Scott Mueller, Ang Li, and Judea Pearl. Causes of effects: Learning individual responses from population data. *arXiv preprint arXiv:2104.13730*, 2021. https://arxiv.org/abs/2104.13730.

[41] Brady Neal. Introduction to causal inference, Fall 2020, lectures and book. https://www.bradyneal.com/causal-inference-course.

[42] Richard E Neapolitan. *Learning Bayesian networks*. Pearson Prentice Hall, 2004.

[43] Andrew Ng. Lecture at deeplearning.ai on recurrent neural networks. http://www.ar-tiste.com/ng-lec-rnn.pdf.

[44] Gregory Nuel. Tutorial on exact belief propagation in Bayesian networks: from messages to algorithms. https://arxiv.org/abs/1201.4724.

[45] paperspace.com. PyTorch 101, Part 1: Understanding graphs, automatic differentiation and autograd. https://blog.paperspace.com/pytorch-101-understanding-graphs-and-automatic-differentiation/.

[46] Judea Pearl. Linear models: A useful microscope for causal analysis. https://ftp.cs.ucla.edu/pub/stat_ser/r409-corrected-reprint.pdf.

[47] Judea Pearl. Mediating instrumental variables. https://ftp.cs.ucla.edu/pub/stat_ser/r210.pdf.

[48] Judea Pearl. On the testability of causal models with latent and instrumental variables. https://arxiv.org/abs/1302.4976.

[49] Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. `https://www.aaai.org/Papers/AAAI/1982/AAAI82-032.pdf`, 1982.

[50] Judea Pearl. *Probabilistic Inference in Intelligent Systems*. Morgan Kaufmann, 1988.

[51] Judea Pearl. The causal mediation formula—a guide to the assessment of pathways and mechanisms. *Prevention science*, 13(4):426–436, 2012. `https://apps.dtic.mil/sti/pdfs/ADA557663.pdf`.

[52] Judea Pearl. *Causality: Models, Reasoning, and Inference, Second Edition*. Cambridge University Press, 2013.

[53] Judea Pearl. Causal and counterfactual inference. *The Handbook of Rationality*, pages 1–41, 2019. `https://ftp.cs.ucla.edu/pub/stat_ser/r485.pdf`.

[54] Judea Pearl and Elias Bareinboim. Transportability of causal and statistical relations: A formal approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011. `https://ojs.aaai.org/index.php/AAAI/article/view/7861`.

[55] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[56] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[57] Judea Pearl and James M Robins. Probabilistic evaluation of sequential plans from causal models with hidden variables. *arXiv preprint arXiv:1302.4977*, 2013. `https://arxiv.org/abs/1302.4977`.

[58] Ashwin Rao and Tikhon Jelvis. *Foundations of Reinforcement Learning with Applications in Finance*. `https://stanford.edu/~ashlearn/RLForFinanceBook/book.pdf`.

[59] ReliaSoft. System analysis reference. `http://reliawiki.org/index.php/System_Analysis_Reference`.

[60] Piotr Rzepakowski and Szymon Jaroszewicz. Decision trees for uplift modeling with single and multiple treatments. *Knowledge and Information Systems*, 32(2):303–327, 2012. `https://link.springer.com/content/pdf/10.1007/s10115-011-0434-0.pdf`.

[61] Scholarpedia. Granger causality. `http://www.scholarpedia.org/article/Granger_causality`.

[62] Marco Scutari. bnlearn. https://www.bnlearn.com/.

[63] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019. https://arxiv.org/abs/1805.11908.

[64] Nitish Srivastava, G E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[65] StatQuest. XGBoost Parts 1-4 (videos). https://statquest.org/video-index/.

[66] Russell Stuart and Norvig Peter. Artificial intelligence-a modern approach 3rd ed, 2016.

[67] Masayoshi Takahashi. Statistical inference in missing data by MCMC and non-MCMC multiple imputation algorithms: Assessing the effects of between-imputation iterations. *Data Science Journal*, 16, 2017. https://datascience.codata.org/articles/10.5334/dsj-2017-037/.

[68] theinvestorsbook.com. Pert analysis. https://theinvestorsbook.com/pert-analysis.html.

[69] Jin Tian and Judea Pearl. Probabilities of causation: Bounds and identification. *Annals of Mathematics and Artificial Intelligence*, 28(1):287–313, 2000. https://ftp.cs.ucla.edu/pub/stat_ser/r271-A.pdf.

[70] Robert R. Tucci. Bayesian networks (a.k.a. causal models, DAGs) and the passage of time. blog post in blog Quantum Bayesian Networks, https://qbnets.wordpress.com/2021/07/16/bayesian-networks-aka-causal-models-dags-and-the-passage-of-time/.

[71] Robert R. Tucci. Bell's inequalities for Bayesian statisticians. blog post in blog Quantum Bayesian Networks, https://qbnets.wordpress.com/2008/09/19/bells-inequaties-for-bayesian-statistician/.

[72] Robert R. Tucci. Goodness of causal fit. https://github.com/rrtucci/DAG_Lie_Detector.

[73] Robert R. Tucci. JudeasRx. https://github.com/rrtucci/JudeasRx.

[74] Robert R. Tucci. Mappa Mundi. https://github.com/rrtucci/mappa_mundi.

[75] Robert R. Tucci. Quantum d-separation and quantum belief propagation. https://arxiv.org/abs/2012.09635.

[76] Robert R. Tucci. Quantum Fog. https://github.com/artiste-qb-net/quantum-fog.

[77] Robert R. Tucci. SCuMpy. https://github.com/rrtucci/scumpy.

[78] Robert R. Tucci. SentenceAx. https://github.com/rrtucci/SentenceAx.

[79] Robert R. Tucci. Shannon information theory without shedding tears over delta & epsilon proofs or typical sequences. https://arxiv.org/abs/1208.2737.

[80] Robert R. Tucci. texnn. https://github.com/rrtucci/texnn.

[81] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. https://arxiv.org/abs/1706.03762.

[82] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook nureg-0492. https://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/.

[83] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference (book). https://people.eecs.berkeley.edu/~wainwrig/Papers/WaiJor08_FTML.pdf.

[84] Lilian Weng. The multi-armed bandit problem and its solutions. lilianweng.github.io/lil-log, http://lilianweng.github.io/lil-log/2018/01/23/the-multi-armed-bandit-problem-and-its-solutions.html, 2018.

[85] Lilian Weng. What are diffusion models? lilianweng.github.io/lil-log, https://lilianweng.github.io/posts/2021-07-11-diffusion-models/, 2021.

[86] Wikibooks. Control systems. https://en.wikibooks.org/wiki/Control_Systems.

[87] Wikipedia. AdaBoost. https://en.wikipedia.org/wiki/AdaBoost.

[88] Wikipedia. Autoregressive model. https://en.wikipedia.org/wiki/Autoregressive_model.

[89] Wikipedia. Autoregressive moving-average model. https://en.wikipedia.org/wiki/Autoregressive%E2%80%93moving-average_model.

[90] Wikipedia. Baum-Welsh algorithm. https://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm.

[91] Wikipedia. Belief propagation. https://en.wikipedia.org/wiki/Belief_propagation.

[92] Wikipedia. Berkson's paradox. https://en.wikipedia.org/wiki/Berkson%27s_paradox.

[93] Wikipedia. Bernoulli distribution. https://en.wikipedia.org/wiki/Bernoulli_distribution.

[94] Wikipedia. BERT language model. https://en.wikipedia.org/wiki/BERT_(language_model).

[95] Wikipedia. Beta distribution. https://en.wikipedia.org/wiki/Beta_distribution.

[96] Wikipedia. Beta function. https://en.wikipedia.org/wiki/Beta_function.

[97] Wikipedia. Binary decision diagram. https://en.wikipedia.org/wiki/Binary_decision_diagram.

[98] Wikipedia. Boolean algebra. https://en.wikipedia.org/wiki/Boolean_algebra.

[99] Wikipedia. Bootstrap aggregating. https://en.wikipedia.org/wiki/Bootstrap_aggregating.

[100] Wikipedia. Categorical distribution. https://en.wikipedia.org/wiki/Categorical_distribution.

[101] Wikipedia. Chi-square distribution. https://en.wikipedia.org/wiki/Chi-square_distribution.

[102] Wikipedia. Chow-Liu tree. https://en.wikipedia.org/wiki/Chow%E2%80%93Liu_tree.

[103] Wikipedia. Cochran's theorem. https://en.wikipedia.org/wiki/Cochran%27s_theorem.

[104] Wikipedia. Conjugate prior. https://en.wikipedia.org/wiki/Conjugate_prior.

[105] Wikipedia. Copula. https://en.wikipedia.org/wiki/Copula_(probability_theory).

[106] Wikipedia. Cramer-Rao bound. https://en.wikipedia.org/wiki/Cram%C3%A9r%E2%80%93Rao_bound.

[107] Wikipedia. Cross-validation. https://en.wikipedia.org/wiki/Cross-validation_(statistics).

[108] Wikipedia. Data processing inequality. https://en.wikipedia.org/wiki/Data_processing_inequality.

[109] Wikipedia. Dirichlet distribution. https://en.wikipedia.org/wiki/Dirichlet_distribution.

[110] Wikipedia. Errors in variables models. https://en.wikipedia.org/wiki/Errors-in-variables_models.

[111] Wikipedia. Expectation maximization. https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm.

[112] Wikipedia. F-distribution. https://en.wikipedia.org/wiki/F-distribution.

[113] Wikipedia. Frisch-Waugh-Lovell theorem. https://en.wikipedia.org/wiki/Frisch%E2%80%93Waugh%E2%80%93Lovell_theorem.

[114] Wikipedia. Functional derivative. https://en.wikipedia.org/wiki/Functional_derivative.

[115] Wikipedia. Gamma distribution. https://en.wikipedia.org/wiki/Gamma_distribution.

[116] Wikipedia. Gamma function. https://en.wikipedia.org/wiki/Gamma_function.

[117] Wikipedia. Gated recurrent unit. https://en.wikipedia.org/wiki/Gated_recurrent_unit.

[118] Wikipedia. Gibbs sampling. https://en.wikipedia.org/wiki/Gibbs_sampling.

[119] Wikipedia. Granger causality. https://en.wikipedia.org/wiki/Granger_causality.

[120] Wikipedia. Hidden Markov model. https://en.wikipedia.org/wiki/Hidden_Markov_model.

[121] Wikipedia. Hoeffding's inequality. https://en.wikipedia.org/wiki/Hoeffding%27s_inequality.

[122] Wikipedia. Importance sampling. https://en.wikipedia.org/wiki/Importance_sampling.

[123] Wikipedia. Instrumental variables estimation. https://en.wikipedia.org/wiki/Instrumental_variables_estimation.

[124] Wikipedia. Inverse transform sampling. https://en.wikipedia.org/wiki/Inverse_transform_sampling.

[125] Wikipedia. Jackknife resampling. https://en.wikipedia.org/wiki/Cross-validation_(statistics).

[126] Wikipedia. Junction tree algorithm. https://en.wikipedia.org/wiki/Junction_tree_algorithm.

[127] Wikipedia. k-means clustering. https://en.wikipedia.org/wiki/K-means_clustering.

[128] Wikipedia. Kalman filter. https://en.wikipedia.org/wiki/Kalman_filter.

[129] Wikipedia. Kernel method. https://en.wikipedia.org/wiki/Kernel_method.

[130] Wikipedia. Kernel perceptron. https://en.wikipedia.org/wiki/Kernel_perceptron.

[131] Wikipedia. Laplace transform. https://en.wikipedia.org/wiki/Laplace_transform.

[132] Wikipedia. Least squares. https://en.wikipedia.org/wiki/Least_squares.

[133] Wikipedia. Legendre transformation. https://en.wikipedia.org/wiki/Legendre_transformation.

[134] Wikipedia. Likelihood-ratio test. https://en.wikipedia.org/wiki/Likelihood-ratio_test.

[135] Wikipedia. Linear regression. https://en.wikipedia.org/wiki/Linear_regression.

[136] Wikipedia. Long short term memory. https://en.wikipedia.org/wiki/Long_short-term_memory.

[137] Wikipedia. Markov blanket. https://en.wikipedia.org/wiki/Markov_blanket.

[138] Wikipedia. Metropolis-Hastings method. https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm.

[139] Wikipedia. Minimum spanning tree. https://en.wikipedia.org/wiki/Minimum_spanning_tree.

[140] Wikipedia. Monte Carlo methods. https://en.wikipedia.org/wiki/Category:Monte_Carlo_methods.

[141] Wikipedia. Moving-average model. https://en.wikipedia.org/wiki/Moving-average_model.

[142] Wikipedia. Multinomial distribution. https://en.wikipedia.org/wiki/Multinomial_distribution.

[143] Wikipedia. Multinomial theorem. https://en.wikipedia.org/wiki/Multinomial_theorem.

[144] Wikipedia. Multivariate normal distribution. https://en.wikipedia.org/wiki/Multivariate_normal_distribution.

[145] Wikipedia. Natural experiment. https://en.wikipedia.org/wiki/Natural_experiment.

[146] Wikipedia. Non-negative matrix factorization. https://en.wikipedia.org/wiki/Non-negative_matrix_factorization.

[147] Wikipedia. Ordinary least squares. https://en.wikipedia.org/wiki/Ordinary_least_squares.

[148] Wikipedia. Program evaluation and review technique. https://en.wikipedia.org/wiki/Program_evaluation_and_review_technique.

[149] Wikipedia. Random forest. https://en.wikipedia.org/wiki/Random_forest.

[150] Wikipedia. Receiver operating characteristic. https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[151] Wikipedia. Rejection sampling. https://en.wikipedia.org/wiki/Rejection_sampling.

[152] Wikipedia. Score test. https://en.wikipedia.org/wiki/Score_test.

[153] Wikipedia. Signal flow graph. https://en.wikipedia.org/wiki/Signal-flow_graph.

[154] Wikipedia. Simple linear regression. https://en.wikipedia.org/wiki/Simple_linear_regression.

[155] Wikipedia. Simpson's paradox. https://en.wikipedia.org/wiki/Simpson's_paradox.

[156] Wikipedia. Spring system. https://en.wikipedia.org/wiki/Spring_system.

[157] Wikipedia. Student's t-distribution. https://en.wikipedia.org/wiki/Student%27s_t-distribution.

[158] Wikipedia. Support vector machine. https://en.wikipedia.org/wiki/Support-vector_machine.

[159] Wikipedia. Survival analysis. https://en.wikipedia.org/wiki/Survival_analysis.

[160] Wikipedia. Time series. https://en.wikipedia.org/wiki/Time_series.

[161] Wikipedia. Transfer learning. https://en.wikipedia.org/wiki/Transfer_learning.

[162] Wikipedia. Transformer (machine learning model). https://en.wikipedia.org/wiki/Transformer_(machine_learning_model).

[163] Wikipedia. Uplift modelling. https://en.wikipedia.org/wiki/Uplift_modelling.

[164] Wikipedia. Variational Bayesian methods. https://en.wikipedia.org/wiki/Variational_Bayesian_methods.

[165] Wikipedia. Vector autoregression. https://en.wikipedia.org/wiki/Vector_autoregression.

[166] Wikipedia. Viterbi algorithm. https://en.wikipedia.org/wiki/Viterbi_algorithm.

[167] Wikipedia. Wald test. https://en.wikipedia.org/wiki/Wald_test.

[168] Wikipedia. Z-transform. https://en.wikipedia.org/wiki/Z-transform.

[169] Hao Wu and Zhaohui Steve Qin. course notes, BIOS731: Advanced statistical computing, 2016 Emory Univ. http://web1.sph.emory.edu/users/hwu30/teaching/statcomp/statcomp.html.

[170] Ronghui (Lily) Xu. Lecture notes, MATH 284, Spring 2020, Survival analysis. https://mathweb.ucsd.edu/~rxu/math284/.

[171] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations, Mitsubishi Technical Report tr-2001-22. https://merl.com/publications/docs/TR2001-22.pdf.