Bayesian Transformers derivation

1 The Transformer Model

1.1 Full Transformer model with stacked encoders and decoders

We denote by L the number of encoder blocks in the full model, and by l a given Transformer block $(1 \leq l \leq L)$.

1.2 Transformer block

In this document, we will primarily focus on deriving the Variational Inference framework for one Transformer block. I is the features dimension of the input. J is the length of the input sequence. Hence, the input X has dimension $I \times J$. K is the dimension of the query and key vectors. D is the dimension of the value vector.

2 Normal Distribution over the Transformer's parameters

Each Transformer block contains the following parameters: $\{W_{hl}^Q, W_{hl}^K, W_{hl}^V\}_{h=1}^H, W_l^O, G_l^1, B_l^1, W_l^{F^1}, W_l^{F^2}, G_l^2, B_l^2$ where H is the number of attention heads. All parameters matrices are assumed to be independent to each other (but not the weights of a given matrix). We assume that each vectorized matrix is drawn from a vector normal distribution.

3 Variational Inference[1]

We set ourselves in the supervised learning framework. A dataset $D = \{(X, y)\}$ is given to us. This is the data we have observed so far, or the data that we can use to train our model i.e., the training set. D can also be seen as a random variable, and the training set is just a set of independent and identically distributed (i.i.d.) samples according to the distribution of D.

Now, a new input X^* is given to us. That means that we want to *generalize* our knowledge of D to a new input X^*. Our ultimate goal is to compute the *distribution* over the output, which is seen as a random variable and is denoted[2] by y^*. We want to know $p(y^*|X^*, D)$. Using the law of total probability and the fact that y^* and D are independent, we have that:

$$p(y^*|X^*, D) = \int_{\Omega} p(y^*|X^*, \omega)p(\omega|D)d\omega. \tag{1}$$

Eq. 1 is called **Bayesian inference**[3]. What is the intuition behind it? Let us explain the terms one by one.

- $p(y^*|X^*, D)$ is the distribution of the output y^* given the input X^* and the dataset D that we have learned.

[1] In this section, I will explain with (hopefully) simple words how I currently understand the variational inference framework.

[2] Using the classical abuse of notations, the probability density function of a random variable $\mathbf{X}$, rigorously defined as $f_{\mathbf{X}}(.)$, will be denoted as $p(X)$ where X indicates at the same time the random variable and the value where the density function is evaluated. We also refer as "distribution" the density function.

[3] I didn't mention the "generative parameters" since I don't really understand them. They may be useful in a more genral framework, but not here as far as I understand.

- To obtain this distribution, we compute a (weighted) sum over the whole parameter space Ω. Each $\omega \in \Omega$ is sampled according to the *posterior distribution* $p(\omega|D)$. This represents what we have learned, i.e., we have used the training set D to learn the distribution of the parameters ω.

- For each ω properly sampled according to our knowledge, we compute the *likelihood* of y^* given the input X^* and our current choice of ω. This likelihood is the term $p(y^*|X^*, \omega)$. This term is simply our (trained) model, our neural network, or whatever it is. Our neural network has a fixed set of parameters ω, and given an input X^*, it outputs a distribution over y^*. In fact, this is true only for Bayesian models. However, classical models can be seen as outputting only $E[y^*|X^*, \omega]$. Or, we can say that classical models only compute the first moment (the mean) of the distribution $p(y^*|X^*, \omega)$.

So finally, what is Bayesian inference? Bayesian inference consists in computing the predictions of *all* possible models (in the family indexed by ω). However, each prediction is weighted by our confidence in the considered model (i.e., $p(\omega|D)$ where each ω is a different model) reflecting our knowledge (the dataset D).

In practice, as far I know, the distribution of y^* is estimated using Monte Carlo estimation. That means that we draw a sample of N vectors of parameters ω according to the distribution $p(\omega|D)$. For each sampled ω, we set our network parameters to this value of ω and we compute the output y^* using the input X^*. Hence, we obtain a sample of y^* approximating the distribution $p(\omega|D)$.

So, the real question here is: how to compute the posterior distribution $p(\omega|D)$. If we replace D by y, X, and we use the fact that X and ω are independent, we get:

$$p(\omega|y, X) = \frac{p(y|\omega, X)p(\omega|X)}{p(y|X)} = \frac{p(y|X, \omega)p(\omega)}{p(y|X)} = \frac{p(y|X, \omega)p(\omega)}{\int_\Omega p(y|X, \omega')p(\omega')d\omega'}, \tag{2}$$

where $p(y|X, \omega)$ is once again the likelihood, or our neural network. $p(\omega)$ is our *prior distribution* over the parameters ω. This can be seen as how we initialize the parameters of our network before the training (before seeing the data). $p(\omega)$ is generally set to a standard normal distribution, or a centred normal distribution with "smart" variance (to be developed). The denominator, called the *marginal likelihood* or the *evidence*, is the real problem here. It is intractable ...

... so this is why we are not going to use this (exact) formula at all to estimate $p(\omega|y, X)$. Instead, we will use a method called **Variational Inference** (VI). VI approximates $p(\omega|y, X)$ by a simpler distribution $q_\theta(\omega)$ called a *approximating variational distribution*. $\{q_\theta(\omega)\}$ is a family of distributions over ω indexed by θ (generally Gaussian distributions). Our goal is to find the θ such that the distribution $q_\theta(\omega)$ is the "closest" to $p(\omega|y, X)$ among the chosen family. To find such a distribution, we will minimize the Kullback-Leibler divergence $KL(q_\theta(\omega)||p(\omega|y, X))$ according to θ. The KL divergence is defined by:

$$KL(q_\theta(\omega)||p(\omega|y, X)) = -\int_\Omega q_\theta(\omega) \log \frac{p(\omega|y, X)}{q_\theta(\omega)} d\omega. \tag{3}$$

Using Eq. 2, $p(\omega|y, X) = \frac{p(y|X,\omega)p(\omega)}{p(y|X)}$, we obtain:

$$\begin{aligned}
KL(q_\theta(\omega)||p(\omega|y, X)) &= -\int_\Omega q_\theta(\omega) \log \frac{p(y|X, \omega)p(\omega)}{p(y|X)q_\theta(\omega)} d\omega, \\
&= -\int_\Omega q_\theta(\omega) \log \frac{p(y|X, \omega)p(\omega)}{q_\theta(\omega)} d\omega + \log p(y|X) \int_\Omega q_\theta(\omega) d\omega, \\
&= -\int_\Omega q_\theta(\omega) \log \frac{p(y|X, \omega)p(\omega)}{q_\theta(\omega)} d\omega + \log p(y|X).
\end{aligned}$$

Hence, we can rewrite the log marginal likelihood $\log p(y|X)$ by:

$$\log p(y|X) = \text{constant} = KL(q_\theta(\omega)||p(\omega|y, X)) + L(\theta, \omega, D), \tag{4}$$

where:

$$L(\theta, \omega, D) = \int_\Omega q_\theta(\omega) \log \frac{p(y|X, \omega)p(\omega)}{q_\theta(\omega)} d\omega, \tag{5}$$

is the variational lower bound, or evidence lower bound (ELBO) on the log marginal likelihood. Indeed, given the fact that the KL-divergence is non-negative, we have: $\log p(y|X) \geq L(\theta, \omega, D)$. Since $\log p(y|X)$ is constant w.r.t θ, minimizing $KL(q_\theta(\omega)||p(\omega|y, X))$ is equivalent to maximizing the ELBO, $L(\theta, \omega, D)$. We can rewrite the ELBO as:

$$
\begin{aligned}
L(\theta, \omega, D) &= \int_\Omega q_\theta(\omega) \log p(y|X, \omega) d\omega + \int_\Omega q_\theta(\omega) \log \frac{p(\omega)}{q_\theta(\omega)} d\omega, \\
&= \mathrm{E}_{q_\theta(\omega)}[\log p(y|X, \omega)] - KL(q_\theta(\omega)||p(\omega)),
\end{aligned}
$$

where $\mathrm{E}_{q_\theta(\omega)}[\log p(y|X, \omega)]$ is the *expected log-likelihood*, and $-KL(q_\theta(\omega)||p(\omega))$ is a regularization term called *complexity loss*. The regularization term encourages the variational distribution $q_\theta(\omega)$ to remain not too far from the prior distribution $p(\omega)$. Since $p(\omega)$ is a simple distribution (like a standard Gaussian), the variational distribution is encouraged to remain not too "complex".

Since the data points of D are assumed to be i.i.d., the ELBO can be estimated by the sum of the individual ELBO of all data points:

$$
L(\theta, \omega, D) = \mathrm{E}_{\tilde{p}(X,y)}[L(\theta, \omega, y|X)] \approx \sum_{(X,y)\in D} L(\theta, \omega, y|X), \tag{6}
$$

where $\tilde{p}(X, y)$ is the empirical distribution from which the training set has been sampled.

4 Derivation of the Expected log-likelihood

To estimate the expected log-likelihood $\mathrm{E}_{q_\theta(\omega)}[\log p(y|X, \omega)]$, we use Monte Carlo estimation where M parameters points are drawn from the distribution $q_\theta(\omega)$:

$$
\mathrm{E}_{q_\theta(\omega)}[\log p(y|X, \omega)] \approx \frac{1}{M} \sum_{m=1}^{M} \log p(y|X, \omega^{(m)}) \tag{7}
$$

To compute $\log p(y|X, \omega^{(m)})$, we need to know the distribution $p(y|X, \omega)$ which depends on the distribution $q_\theta(\omega)$ over the parameters. By assuming that $p(y|X, \omega)$ is Gaussian, we only need to propagate the mean and covariance matrix of $q_\theta(\omega)$ through our neural network from the parameters ω to the outputs y. In this document, we propagate the mean and covariance matrix along one Transformer block with multi-head attention. The flow diagram of one Transformer block is shown in Figure 1.

4.1 Multiplication of two independent random matrices

We prove a result that is used several time in the derivation of the expected log-likelihood. Let A be a $p \times n$ random matrix and B a $p \times q$ random matrix. Let M^A be the mean of A (with dimension $p \times n$), M^B the mean of B (with dimension $p \times q$), Σ^A the covariance matrix of $\mathrm{vec}(A)$ (with dimension $np \times np$), and Σ^B the covariance matrix of $\mathrm{vec}(B)$ (with dimension $qp \times qp$). A and B are assumed to be independent. We denote by $a_1, \cdots, a_n$ the columns of A, and by $b_1, \cdots, b_q$ the columns of B. The elements of the mean matrices M^A and M^B are denoted by $\mu^A_{i,j}$ and $\mu^B_{i,j}$.

Proposition 1.

$$
E[A^T B] = (M^A)^T M^B. \tag{8}
$$

Proof. The expectation of a matrix is the expectation of each element of the matrix. Given a column a_i and a column b_j, we need to compute $E[a_i^T b_j]$. We use the trace trick. We have that $E[a_i^T b_j] = E[\mathrm{tr}(a_i^T b_j)]$ since $a_i^T b_j$ is a scalar. Then, $E[\mathrm{tr}(a_i^T b_j)] = E[\mathrm{tr}(b_j a_i^T)]$ using the fact that $\mathrm{tr}(UV) = \mathrm{tr}(VU)$. Hence, $E[a_i^T b_j] = E[\sum_{k=1}^p B_{k,j} A_{k,i}] = \sum_{k=1}^p E[B_{k,j} A_{k,i}]$ by linearity of expectation. Then, $E[a_i^T b_j] = \sum_{k=1}^p E[B_{k,j}] E[A_{k,j}]$ by independence of A and B. Finally, $E[a_i^T b_j] = \sum_{k=1}^p \mu^B_{k,j} \mu^A_{k,i} = (\mu^A_i)^T \mu^B_j$ where μ^A_i is the i-th column of M^A and μ^B_j is the j-th column of M^B. □

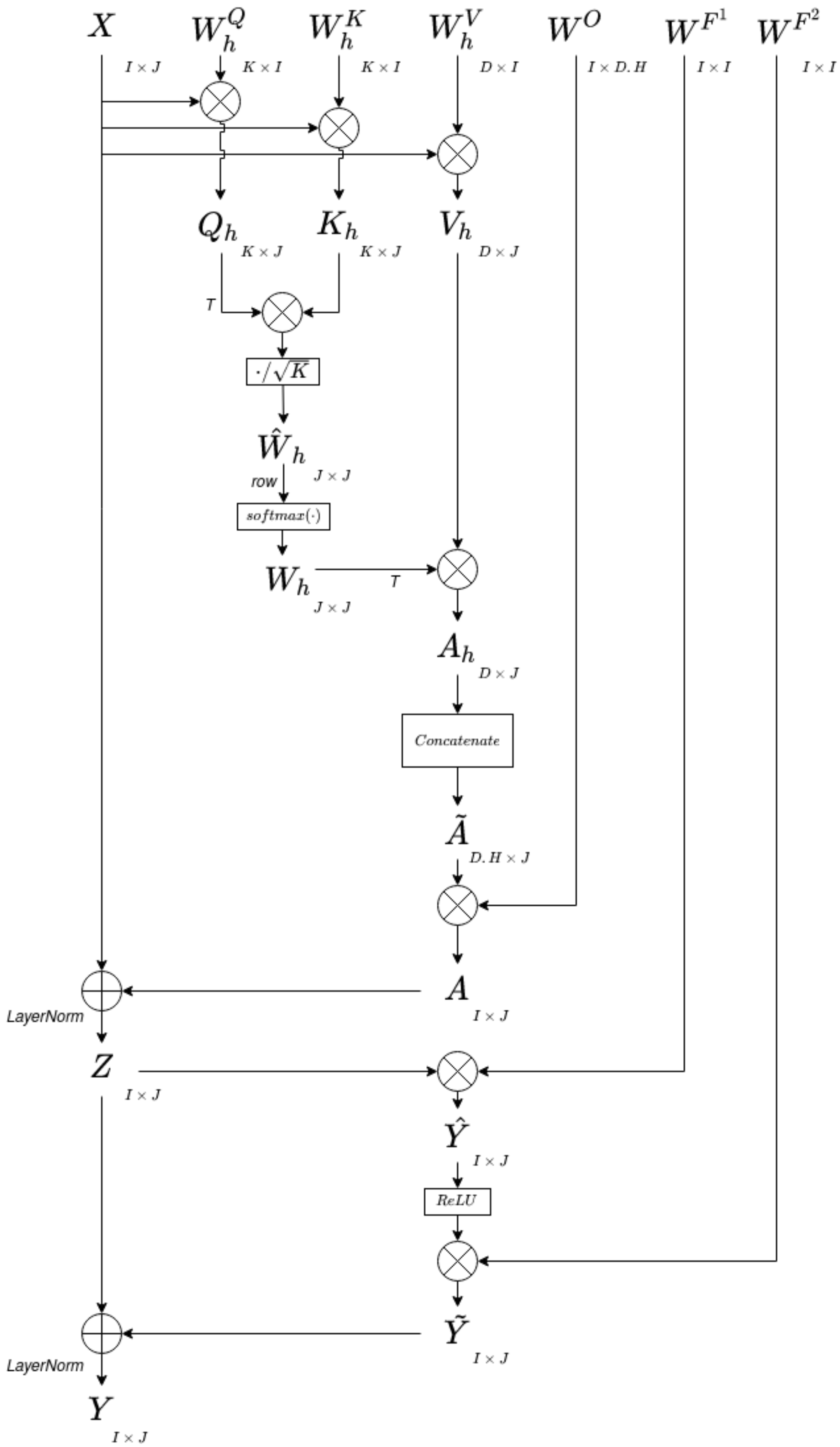

Figure 1: Flow diagram of one Transformer block. The Layer Normalization parameters are not represented.

The vec operation is done by stacking the columns of the matrix (and not the rows) as a column vector. If we denote the cross-covariance matrix of the columns a_i and a_j by Σ^{a_i,a_j} (with dimension $p \times p$), we can write:

$$\Sigma^A = \begin{bmatrix} \Sigma^A_{a_1a_1} & \Sigma^A_{a_1a_2} & \cdots & \Sigma^A_{a_1a_n} \\ \Sigma^A_{a_2a_1} & \Sigma^A_{q_2q_2} & \cdots & \Sigma^A_{a_2a_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma^A_{a_na_1} & \Sigma^A_{a_na_2} & \cdots & \Sigma^A_{a_na_n} \end{bmatrix}.$$

We can do the same thing for B. The covariance matrix of $\text{vec}(A^TB)$ is denoted by Σ with dimension $nq \times nq$ and with elements $\sigma_{i,j}$ such that $\text{Cov}[a^T_{k_1}b_{m_1}, a^T_{k_2}b_{m_2}] = \sigma_{i,j}$ where $k_1 = ((i-1) \bmod n)+1$, $m_1 = \lfloor \frac{i-1}{n} \rfloor + 1$, $k_2 = ((j-1) \bmod n) + 1$, $m_2 = \lfloor \frac{j-1}{n} \rfloor + 1$.

Proposition 2. For all $1 \leq i \leq np$ and $1 \leq j \leq np$, we have:

$$\sigma_{i,j} = \text{tr}(\Sigma^A_{a_{k_1}a_{k_2}} \Sigma^B_{b_{m_1}b_{m_2}}) + \left(\mu^A_{k_1}\right)^T \Sigma^B_{b_{m_1}b_{m_2}} \mu^A_{k_2} + \left(\mu^B_{m_1}\right)^T \Sigma^A_{a_{k_1}a_{k_2}} \mu^B_{m_2}, \tag{9}$$

where $k_1 = ((i-1) \bmod n) + 1$, $m_1 = \lfloor \frac{i-1}{n} \rfloor + 1$, $k_2 = ((j-1) \bmod n) + 1$, $m_2 = \lfloor \frac{j-1}{n} \rfloor + 1$ or equivalently $i = (m_1 - 1)n + k_1$, $j = (m_2 - 1)n + k_2$.

Proof.

$$\begin{aligned}
\text{Cov}[a^T_{k_1}b_{m_1}, a^T_{k_2}b_{m_2}] &= E[a^T_{k_1}b_{m_1}a^T_{k_2}b_{m_2}] - E[a^T_{k_1}b_{m_1}]E[a^T_{k_2}b_{m_2}] \\
&= E[a^T_{k_1}b_{m_1}b^T_{m_2}a_{k_2}] - E[a^T_{k_1}b_{m_1}]E[a^T_{k_2}b_{m_2}] \\
&= E[\text{tr}(a^T_{k_1}b_{m_1}b^T_{m_2}a_{k_2})] - E[a^T_{k_1}b_{m_1}]E[a^T_{k_2}b_{m_2}] \\
&= E[\text{tr}(b_{m_1}b^T_{m_2}a_{k_2}a^T_{k_1})] - E[a^T_{k_1}b_{m_1}]E[a^T_{k_2}b_{m_2}] \\
&= \text{tr}(E[b_{m_1}b^T_{m_2}a_{k_2}a^T_{k_1}]) - E[a^T_{k_1}b_{m_1}]E[a^T_{k_2}b_{m_2}] \\
&= \text{tr}(E[b_{m_1}b^T_{m_2}]E[a_{k_2}a^T_{k_1}]) - E[a^T_{k_1}b_{m_1}]E[a^T_{k_2}b_{m_2}] \\
&= \text{tr}((\Sigma^B_{b_{m_1}b_{m_2}} + \mu^B_{m_1}(\mu^B_{m_2})^T)(\Sigma^A_{a_{k_1}a_{k_2}} + \mu^A_{k_1}(\mu^A_{k_2})^T)) - (\mu^A_{k_1})^T\mu^A_{k_2}(\mu^B_{m_1})^T\mu^B_{m_2} \\
&= \text{tr}(\Sigma^A_{a_{k_1}a_{k_2}}\Sigma^B_{b_{m_1}b_{m_2}}) + \left(\mu^A_{k_1}\right)^T \Sigma^B_{b_{m_1}b_{m_2}}\mu^A_{k_2} + \left(\mu^B_{m_1}\right)^T \Sigma^A_{a_{k_1}a_{k_2}}\mu^B_{m_2}.
\end{aligned}$$

□

4.2 Linear layers: queries, keys, and values

Let X be the input of the Transformer model. X has dimension $I \times J$. The first operation is the computation of the queries, keys and values vectors:

$$\begin{aligned} Q_h &= W^Q_h X, \\ K_h &= W^K_h X, \\ V_h &= W^V_h X. \end{aligned}$$

To compute the means M^Q, M^K, M^V and covariance matrices Σ^Q, Σ^K, Σ^V, we apply Prop. 1 and Prop. 2 using the re-indexing presented in subsection 4.4 and with $M^B = X$ and $\Sigma^B = 0$ if X is a constant matrix.

4.3 Scaled dot-product attention

4.3.1 Mean and covariance of $Q^T_h K_h$

Since W^Q_h and W^K_h are independent, Q_h and K_h are also independent. Let M^Q be the mean of Q_h and M^K the mean of K_h, both of which are $K \times J$ matrices. Using Prop. 1, we have that:

$$E[Q^T_h K_h] = (M^Q)^T M^K.$$

The covariance matrix of $\text{vec}(Q_h)$ is Σ^Q with dimension $K.J \times K.J$. Similarly, the covariance matrix of $\text{vec}(K_h)$ is Σ^K. If we denote the columns of Q_h by $q_1, q_2, \cdots, q_J$, and the cross-covariance matrix of the columns q_i and q_j by $\Sigma^{q_i q_j}$, then we have:

$$\Sigma^Q = \begin{bmatrix} \Sigma^Q_{q_1 q_1} & \Sigma^Q_{q_1 q_2} & \cdots & \Sigma^Q_{q_1 q_J} \\ \Sigma^Q_{q_2 q_1} & \Sigma^Q_{q_2 q_2} & \cdots & \Sigma^Q_{q_2 q_J} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma^Q_{q_J q_1} & \Sigma^Q_{q_J q_2} & \cdots & \Sigma^Q_{q_J q_J} \end{bmatrix}.$$

Using Prop. 2, The covariance matrix of $Q_h^T K_h$ is $\Sigma^{Q^T K}$ (with dimension $J^2 \times J^2$) where the i, j-th element is:

$$\sigma^{Q^T K}_{i,j} = \text{tr}(\Sigma^Q_{q_a q_b} \Sigma^K_{k_c k_d}) + (\mu^Q_{q_a})^T \Sigma^K_{k_c k_d} \mu^Q_{q_b} + (\mu^K_{k_c})^T \Sigma^Q_{q_a q_b} \mu^K_{k_d},$$

where $a = ((i-1) \bmod J) + 1, b = ((j-1) \bmod J) + 1, c = \lfloor \frac{i-1}{J} \rfloor + 1, d = \lfloor \frac{j-1}{J} \rfloor + 1$.

4.3.2 Scaling and softmax activation

Let us denote $\frac{Q_h^T K_h}{\sqrt{K}} = \hat{W}_h$. Then, we have $E[\hat{W}_h] = \frac{(M^Q)^T M^K}{\sqrt{K}}$ and $\Sigma^{\hat{W}_h} = \frac{1}{K} \Sigma^{Q^T K}$. The softmax function is then applied to each row of $\hat{W}_h$. Let w be such a row (with dimension $1 \times J$), with mean μ^w and covariance matrix Σ^w. We will denote the softmax function by ϕ with ϕ_i being its i-th component. The first-order Taylor approximation of ϕ_i is[4]:

$$\begin{aligned} \phi_i(w) &\approx \phi_i(\mu^w) + \nabla \phi_i(\mu^w)^T (w - \mu^w), \\ &\approx \phi_i(\mu^w) + \sum_{j=1}^J \frac{\partial \phi_i}{\partial x_j}(\mu^w)(w_j - \mu^w_j). \end{aligned}$$

Hence, we have:

$$E[\phi_i(w)] \approx \phi_i(\mu^w). \tag{10}$$

Now, consider two rows w^a and w^b (where a and b can be equal).

Proposition 3.

$$\text{Cov}[\phi_i(w^a), \phi_k(w^b)] \approx \sum_{j=1}^J \sum_{l=1}^J \frac{\partial \phi_i}{\partial x_j}(\mu^{w^a}) \frac{\partial \phi_k}{\partial x_l}(\mu^{w^b}) \text{Cov}[w^a_j, w^b_l], \tag{11}$$

where $\text{Cov}[w^a_j, w^b_l] = \sigma_w^{a+(j-1)J, b+(l-1)J}$.

Proof.

$$\begin{aligned} \text{Cov}[\phi_i(w^a), \phi_k(w^b)] &\approx E[(\phi_i(\mu^w) + \sum_{j=1}^J \frac{\partial \phi_i}{\partial x_j}(\mu^w)(w_j - \mu^w_j))(\phi_j(\mu^w) + \sum_{l=1}^J \frac{\partial \phi_j}{\partial x_l}(\mu^w)(w_l - \mu^w_l))] \\ &\quad - E[\phi_i(\mu^w) + \sum_{j=1}^J \frac{\partial \phi_i}{\partial x_j}(\mu^w)(w_j - \mu^w_j)] E[\phi_j(\mu^w) + \sum_{l=1}^J \frac{\partial \phi_j}{\partial x_l}(\mu^w)(w_l - \mu^w_l)] \\ &\approx E[(\sum_{j=1}^J \frac{\partial \phi_i}{\partial x_j}(\mu^w)(w_j - \mu^w_j))(\sum_{l=1}^J \frac{\partial \phi_j}{\partial x_l}(\mu^w)(w_l - \mu^w_l))] \\ &\approx E[\sum_{j=1}^J \sum_{l=1}^J \frac{\partial \phi_i}{\partial x_j}(\mu^w)(w_j - \mu^w_j) \frac{\partial \phi_j}{\partial x_l}(\mu^w)(w_l - \mu^w_l)] \\ &\approx \sum_{j=1}^J \sum_{l=1}^J \frac{\partial \phi_i}{\partial x_j}(\mu^w) \frac{\partial \phi_j}{\partial x_l}(\mu^w) \text{Cov}[w^a_j, w^b_l] \end{aligned}$$

□

[4] As a remainder: $\frac{\partial \phi_i}{\partial x_j}(w) = \phi_i(w)(\mathbf{1}_{i=j} - \phi_j(w))$

We finally obtain the $J \times J$ matrix W_h, such that $E[W_h^{r,i}] = E[\phi_i(w^r)]$.

4.3.3 Multiplication with V_h

We are looking for the mean and covariance of $A_h = V_h W_h^T$ (dimension $D \times J$). From Prop. 1, we get $E[A_h] = M^V (M^W)^T$. For the covariance matrix, we want to directly apply Prop. 2 but there is a problem of indexation. What we have now are the covariance matrices of $\mathrm{vec}(V_h)$ and $\mathrm{vec}(W_h)$. However, what we need here are the covariance matrices of $\mathrm{vec}(V_h^T)$ and $\mathrm{vec}(W_h^T)$. We denote by i_1, j_1 the indexes of an element of the covariance matrix of $\mathrm{vec}(V_h)$ (remind that V_h has dimension $D \times J$). Then the new indexes of this element in the covariance matrix of $\mathrm{vec}(V_h^T)$ are $i_2 = ((i_1 - 1) \bmod D)J + \lfloor \frac{i_1 - 1}{D} \rfloor + 1$ and $j_2 = ((j_1 - 1) \bmod D)J + \lfloor \frac{j_1 - 1}{D} \rfloor + 1$. Using these new covariance matrices (and the transpose of the expectation matrices), we can apply Prop. 2.

4.4 Concatenation of the A_h and multiplication by W^O

The A_h from the various attention heads are concatenated into $\tilde{A}$. The mean of $\tilde{A}$ is the concatenation of the means of the A_h. To get the covariance matrix of $\tilde{A}$, let us consider for all h the block covariance matrix of A_h:

$$\Sigma^h = \begin{bmatrix} \Sigma^h_{1,1} & \Sigma^h_{1,2} & \cdots & \Sigma^h_{1,J} \\ \Sigma^h_{2,1} & \Sigma^h_{2,2} & \cdots & \Sigma^h_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma^h_{J,1} & \Sigma^h_{J,2} & \cdots & \Sigma^h_{J,J} \end{bmatrix},$$

where each block is a $D \times D$ matrix. The covariance matrix $\Sigma^{\tilde{A}}$ of $\tilde{A}$ is a $DHJ \times DHJ$ matrix that can be seen as a block matrix with $HJ \times HJ$ blocks of dimension $D \times D$. Let $\Sigma^{\tilde{A}}_{i,j}$ be the block with indexes i, j in $\Sigma^{\tilde{A}}$. If $(i-1) \bmod D \neq (j-1) \bmod D$, then $\Sigma^{\tilde{A}}_{i,j} = \mathbf{0}$ since the attention heads are assumed to be independent to each other[5]. Else if $(i-1) \bmod D = (j-1) \bmod D$, then let $h = (i-1) \bmod D$, $k = \lfloor \frac{i-1}{D} \rfloor + 1$, $l = \lfloor \frac{j-1}{D} \rfloor + 1$ and we have $\Sigma^{\tilde{A}}_{i,j} = \Sigma^h_{k,l}$.

We now consider $A = W^O \tilde{A}$. We have $E[A] = M^O M^A$. To compute the covariance matrix of $\mathrm{vec}(A)$ with Prop. 2, we have to rearrange the elements of Σ^O (covariance matrix of $\mathrm{vec}(W^O)$) as in subsection 4.3.3. Then we can apply Eq. 9 using this new covariance matrix as well as the transpose of M^O.

4.5 Residual Connection and Layer Normalization

Let $\hat{A} = A + X$. We have that $E[\hat{A}] = M^A + M^X$ and $\mathrm{Cov}[\mathrm{vec}(\hat{A})] = \Sigma^A + \Sigma^X$. Let a be a column of $\hat{A}$. Let us consider $\mu = \frac{1}{I}\sum_{i=1}^{I} E[a_i]$ and $\sigma = \frac{1}{I}\sum_{i=1}^{I}(E[a_i] - \mu)^2$. The layer normalization operation is the following: $z_i = g_i \frac{a_i - \mu}{\sigma} + b_i$ where g_i and b_i are learnable parameters from G^1 and B^1. Using the assumption that g_i is independent from a_i, we have that $E[z_i] = E[g_i]\frac{E[a_i] - \mu}{\sigma} + E[b_i]$. Let us rewrite this formula with matrices. Let $\mathbf{J}$ be the $I \times I$ matrix full of 1. Let $M^\mu = \frac{1}{I}\mathbf{J}M^{\hat{A}}$ and $M^\sigma = \frac{1}{I}\mathbf{J}(M^{\hat{A}} - M^\mu)^{.2}$ where the square operation is applied element-wise. Let $M^{G^1} = \begin{bmatrix} \mu^{G^1} & \cdots & \mu^{G^1} \end{bmatrix}$ a $I \times J$ matrix and M^{B^1} defined similarly. We obtain

$$M^Z = M^{G^1} \odot (M^{\hat{A}} - M^\mu)./M^\sigma + M^{B^1},$$

where $\odot$ is the Hadamard product and the division is applied element-wise.

Proposition 4.

$$\mathrm{Cov}[Z_{i,j}, Z_{k,l}] = \frac{\mathrm{Cov}[g_i, g_k]\{\mathrm{Cov}[A_{i,j}, A_{k,l}] + (E[A_{i,j}] - \mu_j)(E[A_{k,l}] - \mu_l)\} + \mathrm{Cov}[A_{i,j}, A_{k,l}]E[g_i]E[g_k]}{\sigma_j \sigma_l} + \mathrm{Cov}[b_i, b_k]$$

where μ_j and σ_j are the average and variance of the j-th column of $\hat{A}$ (similarly for l).

[5] which is not true since they all depend on the input X.

4.6 Feed-forward layer

The mean and covariance matrix of $\hat{Y} = W^{F^1} Z$ are computed by the same method as the moments of A in subsection 4.4. Then, we use Prop. 3 (with $J = 1$) to propagate the mean and covariance through the ReLU activation. Once again, we apply the method of subsection 4.4 to compute the moments of $\tilde{Y} = W^{F^2}\text{ReLU}(\hat{Y})$. Finally, we apply subsection 4.5 to obtain the moments of the output Y. The only difference is that the residual connection is with Z (a random matrix) and not with X (a constant matrix). This implies that $E[\tilde{Y} + Z] = M^{\tilde{Y}} + M^Z$ and $\text{Cov}[\text{vec}(\tilde{Y} + Z)] = \Sigma^{\tilde{Y}} + \Sigma^Z$ assuming that $\tilde{Y}$ and Z are independent (which is not true).

Hence, we have obtain the mean M^Y and the covariance matrix Σ^Y of Y.

5 Derivation of the Complexity Loss

We now derive the complexity loss $KL(q_\theta(\omega)||p(\omega))$. Let us assume that the priors are $\mathcal{N}(\mathbf{0}, \mathbf{I})$. We have one term for each layer. Let W^R be one of these parameter matrices with R rows. The complexity loss term for such parameters is[6]

$$\sum_{r=1}^{R} KL(\mathcal{N}(\mu_r, \sigma_r^2\mathbf{I})||\mathcal{N}(\mathbf{0}, \mathbf{I})) = \frac{1}{2}\sum_{r=1}^{R}(n_c\sigma_r^2 + \|\mu_r\|_F^2 - n_c - n_c\log(\sigma_r^2)),$$

where n_c is the number of columns and where we assume that the covariance matrix is diagonal with the same variance. The covariance matrix of $\text{vec}(W^R)$ is a block diagonal matrix of dimension $n_cR \times n_cR$ with n_c identical blocks of the form $\text{diag}(\sigma_1, \cdots, \sigma_R)$. For the Layer Normalization parameters, we have the same formula with $R = 1$. Denoting the total number of parameters matrices by P we have:

$$KL(q_\theta(\omega)||p(\omega)) = \frac{1}{2}\sum_{p=1}^{P}\sum_{r=1}^{R_p}(n_{p,c}\sigma_{p,r}^2 + \|\mu_{p,r}\|_F^2 - n_{p,c} - n_{p,c}\log(\sigma_{p,r}^2)). \tag{12}$$

Given a mini-batch of N i.i.d. datapoint, the ELBO can be written as:

$$\begin{aligned} L(\theta, \omega, D) = &-\frac{NIJ}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{N}[\log(|\Sigma^{Y_i}|) + (Y_i - M^{Y_i})^T(\Sigma^{Y_i})^{-1}(Y_i - M^{Y_i})] \\ &- \frac{1}{2}\sum_{p=1}^{P}\sum_{r=1}^{R_p}(n_{p,c}\sigma_{p,r}^2 + \|\mu_{p,r}\|_F^2 - n_{p,c} - n_{p,c}\log(\sigma_{p,r}^2)), \end{aligned} \tag{13}$$

where the Y_i and M^{Y_i} have been vectorized.

[6]The proof will be provided later.