Notations and Definitions

Throughout this work we will make use of various concepts and their notations, which we will define here for easier reference, and to avoid redundancy. Bold symbols (e.g. $\boldsymbol{v}$) denote vectors, and v_i the i-th element of the respective vector. Upper-cased bold symbols (e.g. $\boldsymbol{M}$) denote matrices, and M_{ij} the element in the i-th row and j-th column of the respective matrix.

Loss Functions

Mean Squared Error (MSE)

The Mean Squared Error (MSE) is a loss function used in regression tasks, and describes the average of the squared differences between the prediction $\hat{\boldsymbol{y}} \in \mathbb{R}^d$ and the target $\boldsymbol{y} \in \mathbb{R}^d$. Since in this work the predictions and targets will exclusively be in the form of d-dimensional vectors, the MSE is defined as:

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 = \frac{1}{d} \sum_{j=1}^{d} \left(y_j - \hat{y}_j\right)^2 \tag{1}$$

Kullback-Leibler Divergence (KL-Divergence)

The Kullback-Leibler Divergence (KL-Divergence) is used to measure the difference between two probability distributions. Specifically, in the context of Machine Learning, we are comparing a predicted probability distribution $\boldsymbol{q} \in \mathbb{R}^d$ with a target distribution $\boldsymbol{p} \in \mathbb{R}^d$. Since we are using the KL-Divergence in the context of classification tasks, which are discrete distributions over classes, the KL-Divergence is defined as:

$$\mathcal{L}_{\text{KD}}(\boldsymbol{p} \parallel \boldsymbol{q}) = D_{\text{KL}}(\boldsymbol{p} \parallel \boldsymbol{q}) = \sum_j p_j \log \frac{p_j}{q_j} \tag{2}$$

p_j and q_j are the probabilities of class j according to the target and predicted distribution, respectively. For both distributions, there are potentially multiple classes with a non-zero probability:

$$\forall j (p_j \in [0, 1]) \land \sum_j p_j = 1 \tag{3}$$

Cross-Entropy Loss (CE)

The Cross-Entropy Loss (CE) is quite similar to the KL-Divergence in that it compares two probability distributions in classification tasks. It is defined as:

$$\mathcal{L}_{\text{CE}}(\boldsymbol{p}, \boldsymbol{q}) = H(\boldsymbol{p}, \boldsymbol{q}) = H(\boldsymbol{p}) + D_{\text{KL}}(\boldsymbol{p} \parallel \boldsymbol{q}) = -\sum_j p_j \log p_j + \sum_j p_j \log \frac{p_j}{q_j} \tag{4}$$

Here $H(\boldsymbol{p})$ denotes the entropy of the target distribution $\boldsymbol{p}$, and $D_{\text{KL}}(\boldsymbol{p} \parallel \boldsymbol{q})$ the KL-Divergence between the target and predicted distribution.

The difference between KL-Divergence and cross-entropy is that the latter is used in traditional classification tasks, where the target distribution $\boldsymbol{p}$ is fixed and one-hot encoded, meaning that there is only one correct class:

$$\exists! j (p_j = 1) \land \forall k (k \neq j \rightarrow p_k = 0) \tag{5}$$

This strengthens the condition of the KL-Divergence, which we defined previously in Equation 3. Since the goal is to minimize the cross-entropy loss $H(\boldsymbol{p}, \boldsymbol{q})$ and $\boldsymbol{p}$ is fixed, the entropy of the target distribution $H(\boldsymbol{p})$ is a constant, and does not affect the minimization. Moreover, given the constraint in Equation 5, only one term in the sum of the KL-Divergence is non-zero. Consequently, we can simplify the cross-entropy loss, so that the training objective for classification tasks is:

$$\begin{aligned}\min H(\boldsymbol{p},\boldsymbol{q}) &= H(\boldsymbol{p}) + D_{\mathrm{KL}}(\boldsymbol{p} \parallel \boldsymbol{q}) \\ &= D_{\mathrm{KL}}(\boldsymbol{p} \parallel \boldsymbol{q}) \\ &= \sum_j p_j \log \frac{p_j}{q_j} \\ &= \log \frac{1}{q_i} \\ &= -\log q_i\end{aligned} \tag{6}$$

The cross entropy loss therefore minimizes the negative log-likelihood of the correct class i.

Often times, the prediction of a model $\boldsymbol{x}$ is returned as raw logits, and not as probabilities. To convert logits into probabilities, the softmax function is used. For ease of use, without having to mention a softmax-normalization every time we make use of the cross-entropy loss, we redefine the cross-entropy loss *actually used in this work* as:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{p},\boldsymbol{x}) = H(\boldsymbol{p},\boldsymbol{x}) = -\log \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{7}$$

We denote $\boldsymbol{x}$ as the raw logits (the model prediction), and $\boldsymbol{p}$ as the one-hot encoded target distribution. i is the index of the correct class, and hence each element in $\boldsymbol{x}$ corresponds to the raw logit for one class.

A comparison between the target distribution predicted using KL-Divergence, and another predicted by cross-entropy is shown in the following figure.

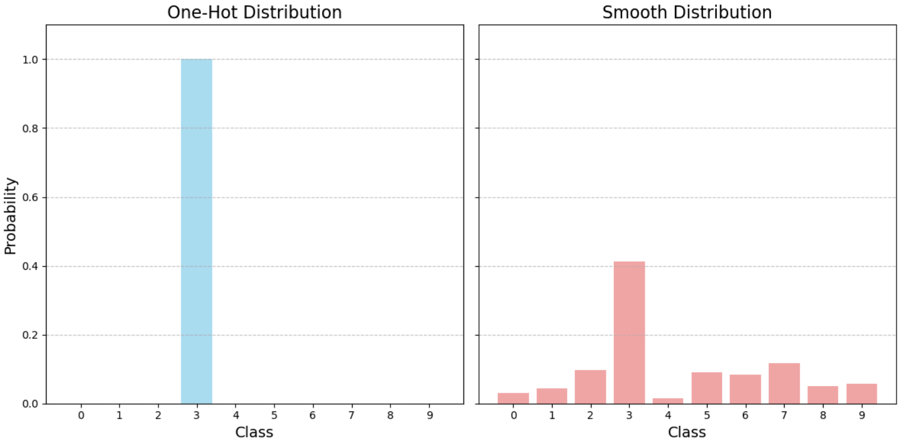

Figure 1: Comparison between the distributions with 10 classes. The one-hot distribution (left) is used for classification tasks with the cross-entropy loss. The KL-Divergence is used when predicting a smooth distribution (right). A smooth distribution usually results from a model prediction, and is a popular target distribution for knowledge distillation, introduced in a later section.

Modality Representations

Since the architectures used in the experiments of this work are based on the Transformer [1] and vision Transformer [2] architecture, both image and text are represented as sequences of embeddings, which are processed by the Transformer blocks.

Image Representation

We define an image as a 3-dimensional tensor $\boldsymbol{v} \in \mathbb{R}^{C \times H \times W}$. Because we will use the base variant of the vision Transformer, ViT-B/16 [2], the image is patchified into 14x14 patches, each being a square of size 16x16 pixels. Each image patch represents one timestep in the sequence, and the number of patches N is given by $N = H \times \frac{W}{P^2}$, with P being the number of patches per dimension, and $P = 14$. Since we use an image size of 224x224 pixels, so $\boldsymbol{v} \in \mathbb{R}^{3 \times 244 \times 244}$, we will have $N = 244 \times \frac{244}{14^2} = 196$ patches, or timesteps respectively. Each patch is flattened into a 256-dimensional vector, and then projected into a 768 dimensions $\boldsymbol{e}_i^v \in \mathbb{R}^{768}$, using a fully connected layer. The image sequence is prepended with a special learnable $[\texttt{I\_CLS}] \in \mathbb{R}^{768}$ token, which is, following [2], used to aggregate the global information/content of the image. The result is a sequence of patch embeddings, which we define as $\boldsymbol{E}_v$, where v indicates the image modality:

$$\boldsymbol{E}_v = \left[\boldsymbol{e}^v_{[\texttt{I\_CLS}]}, \boldsymbol{e}^v_1, \boldsymbol{e}^v_2, ..., \boldsymbol{e}^v_N\right] \tag{8}$$

To give the Transformer a sense of order in the image patches/timestep, a unique positional encoding is added to each patch embedding. This can either be learned or fixed, with the latter being for example a sinusoidal positional encoding [1]. This positional encoding is also represented as a sequence of 768-dimensional vectors:

$$\boldsymbol{T}^{\text{pos}}_v = \left[0, \boldsymbol{t}^v_{\text{pos}_1}, \boldsymbol{t}^v_{\text{pos}_2}, ..., \boldsymbol{t}^v_{\text{pos}_N}\right] \tag{9}$$

Since the [I_CLS] token is not part of the image, the positional encoding for the [I_CLS] token is set to zero, so nothing is added to it. An image representation is defined as:

$$\boldsymbol{H}^s_{v,l} = \left[\boldsymbol{h}^s_{v,l,[\texttt{I\_CLS}]}, \boldsymbol{h}^s_{v,l,1}, ..., \boldsymbol{h}^s_{v,l,N}\right] \tag{10}$$

In Equation 10, l denotes the layer of the Transformer block that returned the image representation, and v indicates that the representation is an image. Since we use Knowledge Distillation (KD) in some parts of this thesis, representations will be, if neccessary, superscripted with s or t, for a student and teacher representation, respectively.

We define $l = 0$ as the input to the Transformer, and $l = L$ as the output of the Transformer, where L is the number of layers in the Transformer. Consequently, the image input to the Transformer is defined as:

$$\boldsymbol{H}^s_{v,0} = \left[\boldsymbol{h}^s_{v,0,[\texttt{I\_CLS}]}, \boldsymbol{h}^s_{v,0,1}, ..., \boldsymbol{h}^s_{v,0,N}\right] = \boldsymbol{E}_v + \boldsymbol{T}^{\text{pos}}_v \tag{11}$$

The output of the Transformer is defined as:

$$\boldsymbol{H}^s_{v,L} = \left[\boldsymbol{h}^s_{v,L,[\texttt{I\_CLS}]}, \boldsymbol{h}^s_{v,L,1}, ..., \boldsymbol{h}^s_{v,L,N}\right] \tag{12}$$

Text Representation

We define a text as a sequence of discrete tokens, which are, similiar to image patches, embedded into 768-dimensional vectors using an embedding matrix. A single token i is represented as $\boldsymbol{e}^t_i \in \mathbb{R}^{768}$, and the sequence of tokens, representing the text, is prepended with a start-of-sequence token $[\texttt{T\_CLS}] \in \mathbb{R}^{768}$, and appended with an end-of-sequence token $[\texttt{T\_SEP}] \in \mathbb{R}^{768}$. The purpose of the [T_CLS] token is, as with [I_CLS], to aggregate the global information/content of the text. The [T_SEP] token is used to indicate the end of the text sequence. A text sequence consists of M tokens, and we use w to denote a text sequence:

$$\boldsymbol{E}_w = \left[\boldsymbol{e}^w_{[\texttt{T\_CLS}]}, \boldsymbol{e}^w_1, \boldsymbol{e}^w_2, ..., \boldsymbol{e}^w_M, \boldsymbol{e}^w_{[\texttt{T\_SEP}]}\right] \tag{13}$$

The maximum text sequence length M is not fixed, and will be defined when neccessary in the experimental part of this work.

A positional encoding is also added to the text embeddings, to give the Transformer a sense of order in the text sequence. Since the special token [T_SEP] denotes the end of the text sequence, it is part of the sequence, and therefore has a positional encoding. The latter does not hold for the [T_CLS] token, as it is used to aggregate the global information/content of the text.

$$\boldsymbol{T}_w^{\text{pos}} = \left[0, \boldsymbol{t}_{\text{pos}_1}^w, \boldsymbol{t}_{\text{pos}_2}^w, ..., \boldsymbol{t}_{\text{pos}_M}^w, \boldsymbol{t}_{\text{pos}_{[\text{T\_SEP}]}}^w\right] \tag{14}$$

A text representation is defined as:

$$\boldsymbol{H}_{w,l}^s = \left[\boldsymbol{h}_{w,l,[\text{T\_CLS}]}^s, \boldsymbol{h}_{w,l,1}^s, ..., \boldsymbol{h}_{w,l,M}^s, \boldsymbol{h}_{w,l,[\text{T\_SEP}]}^s\right] \tag{15}$$

Equation 15 denotes the representation denoted by a student model s, but it can also be a teacher representation t.

The input to the Transformer for text is Equation 15 with $l = 0$, and the output of the Transformer is Equation 15 with $l = L$.

Transformer Block

Unless we use pretrained architectures that follow a different architecture, which we will then specify, we follow the Pre-LayerNorm definition of the Transformer block as given in [3]. As the name suggests, it applies LayerNorm before the Multi-Head Attention, instead of after.

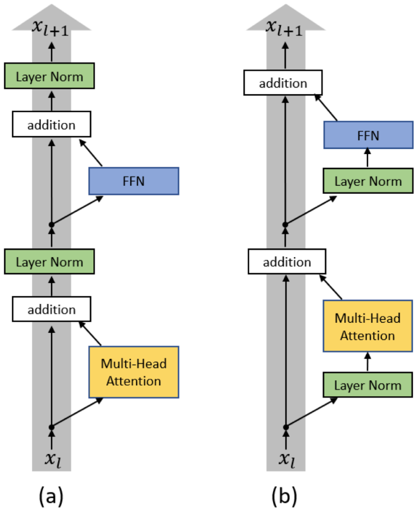

Figure 2: Comparison of a Post-Norm Transformer block/layer (a), and a Pre-Norm Transformer block/layer (b). (a) is the architecture as defined in the original “Attention is all you need” paper [1]. We follow the Pre-Norm architecture [3].

One Transformer block performs the following operations:

$$\boldsymbol{H}_l' = \text{MHA}(\text{LN}(\boldsymbol{H}_{l-1})) + \boldsymbol{H}_{l-1} \tag{16}$$

$$\boldsymbol{H}_l = \text{FFN}(\text{LN}(\boldsymbol{H}_l')) + \boldsymbol{H}_l' \tag{17}$$

We denote LN as LayerNorm, MHA as Multi-Head Attention, and FFN as a 2 layer MLP, all following the original Transformer of [1]. As previously mentioned, the only difference is the order of operations [3]. $\boldsymbol{H}_{v,l}^s$ and $\boldsymbol{H}_{w,l}^s$ can be used as a drop-in replacement for image and text, respectively. Both equations are, with slight adjustment, taken from VLMo [4].

We define a Transformer as multiple Transformer blocks stacked on top of each other.

Bibliography

[1] A. Vaswani *et al.*, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[2] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” *ICLR*, 2021.

[3] R. Xiong *et al.*, “On layer normalization in the transformer architecture,” in *Proceedings of the 37th International Conference on Machine Learning*, in ICML'20. JMLR.org, 2020.

[4] H. Bao *et al.*, “VLMo: Unified Vision-Language Pre-Training with Mixture-of-Modality-Experts,” in *Advances in Neural Information Processing Systems*, 2022. [Online]. Available: https://openreview.net/forum?id=bydKs84JEyw