Unimodal Knowledge Distillation

To validate whether traditional unimodal knowledge distillation, an undoubtedly simpler task than multimodal knowledge distillation, even works, we will first conduct experiments on unimodal knowledge distillation. We will then build on the results to develop a multimodal knowledge distillation.

Vision

Method

Our approach to vision KD involves using a pretrained Data2Vec2 [1] image model as the teacher model, and distilling a shallow version of this model, which is the student. We attribute our choice of Data2Vec2 to its effectiveness and consistency in self-supervised learning across image, text and audio. Data2Vec2 is a general framework to pretrained *unimodal* image, text, and audio models using self-supervised learning [2], [1], which fits our philosophy of aligning modalities.

We approach the distillation by taking the first 6 Transformer blocks of the *pretrained* teacher model, which are exactly half of the 12 Transformer blocks in the teacher, and organize them into a smaller model. This smaller model also includes the pretrained cls token, patch projection, and positional encodings. Consequently, the student model is not trained from scratch, but already initialized with a subset of the teacher's weights. This is inspired by DistilBERT [3], a small BERT variant distilled from the normal BERT model, selecting every second layer from a pretrained BERT [4] model and organizing them into a smaller model. As mentioned before, we use the first 6 Transformer blocks of the teacher model, which we found leads to better results than using every second layer. The resulting student model is with 43.1M parameters almost half the size of the teacher model, which has 85.9M parameters.

Data2Vec2 is a self-supervised model [1], and therefore does not provide a probability distribution over classes that can be predicted using KL-Divergence. Instead, we only have access to the model's activations for each layer, so we have to resort to feature-based knowledge distillation. One option would be to predict the teacher's output for the cls token $\boldsymbol{h}^{t}_{v,L,[\texttt{I\_CLS}]}$, which aggregates the high level content of the image, and then use the mean squared error as the loss function. However, this neglects the activations for individual image patches and activations of intermediate layers.

This argument is quite similar to that of Data2Vec. The authors introduce "contextualized representations", which are the activations of all layers of a model for each time step of the input. Because of Self-Attention in Transformers, the activations for each image patch (time step) are influenced by all other image patches, and therefore not only encode information about a patches content, but also about its context in the image, i.e. the relationship to other patches [2], [1]. Consequently, contextualized representations are more informative than a single cls token, as they encode information about the image at different levels of abstraction, and how the model aggregates low level features to high level concepts. Since the goal of KD is to "mimic" the behavior of a teacher model for a given input in a compressed way, this is the exact information that should be transferred from the teacher to the student. Simply predicting the cls token would only "mimic" what information the teacher extracts from the image, but not how the information is extracted.

While the dimensions of our student model match those of the teacher model, they both have a hidden size of $D = 768$ and intermediate size of $D_{\text{ff}} = 3072$, the number of layers in the student model ($L_s = 6$) is only half of that of the teacher model ($L_t = 12$). It is therefore not possible for each layer of the student model to mimic the behavior of the corresponding layer in the teacher model. Fortunately, experiments of the Data2Vec authors show that predicting the mean of all layer activations for each time step (or image patch, respectively) works as well as predicting the activations of each layer individually [2]. This suits our approach well, as the only mismatch between the teacher and student model is the number of layers, which is irrelevant when predicting the mean of all layer activations for each time step. The authors apply instance normalization to the activations of each layer before averaging, which is a normalization technique that works on each dimension of a sequence independently.

For a sequence of embeddings/representations with length T, instance normalization is defined as follows:

$$h'_{j,d} = \frac{h_{j,d} - \mu_d}{\sqrt{\sigma_d^2 + \varepsilon}}, \qquad \mu_k = \frac{1}{T}\sum_{t=1}^{T} h_{t,k}, \qquad \sigma_k^2 = \frac{1}{T}\sum_{t=1}^{T} \left(h_{t,k} - \mu_k\right)^2 \tag{1}$$

Even though the formula might look complicated, it is quite simple in practice. For each embedding dimension d, the mean μ_d and standard deviation σ_d are calculated over all time steps T. In the case of an embedding dimension of $D = 768$, this means for one sample (e.g. a sequence representing an image) 768 means and standard deviations are calculated, one for each embedding dimension. Then, for each time step j, the embedding at time step j is normalized by normalizing each dimension of the embedding independently, using the corresponding mean and standard deviation computed for that dimension [5]. During the normalization, a small epsilon, e.g. $1e^{-8} = 10^{-8}$, is added to the standard deviation to prevent division by zero. For an illustrative comparison between instance normalization, batch normalization and layer normalization, see (TODO: cite normalization) in the appendix. We define the operation $\text{InstanceNorm}(\cdot)$ as instance normalization on a sequence of embeddings $\boldsymbol{H}$.

$$\text{InstanceNorm}(\boldsymbol{H}) = [\boldsymbol{h}'_1, \boldsymbol{h}'_2, ..., \boldsymbol{h}'_T] \tag{2}$$

After instance norm and averaging, parameter-less layer normalization is performed [2], [1]. We perform all three operations likewise. The target and prediction are therefore given by:

$$\begin{aligned} \boldsymbol{H'}^t_{v,l} &= \text{InstanceNorm}\left(\boldsymbol{H}^t_{v,l}\right), l \in \{1, 2, ..., L_t\} \\ \boldsymbol{H'}^s_{v,l} &= \text{InstanceNorm}\left(\boldsymbol{H}^s_{v,l}\right), l \in \{1, 2, ..., L_s\} \end{aligned} \tag{3}$$

$$\begin{aligned} \widehat{\boldsymbol{H}}^t_v &= \frac{1}{L_t}\sum_{l=1}^{L_t} \boldsymbol{H'}^t_{v,l} \\ \widehat{\boldsymbol{H}}^s_v &= \frac{1}{L_s}\sum_{l=1}^{L_s} \boldsymbol{H'}^s_{v,l} \end{aligned} \tag{4}$$

$$\begin{aligned} \boldsymbol{Y} &= \left[\boldsymbol{y}_{[\texttt{I\_CLS}]}, \boldsymbol{y}_1, ..., \boldsymbol{y}_N\right] = \text{LayerNorm}\left(\widehat{\boldsymbol{H}}^t_v\right) \\ \widehat{\boldsymbol{Y}} &= \left[\hat{\boldsymbol{y}}_{[\texttt{I\_CLS}]}, \hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_N\right] = \text{LayerNorm}\left(\widehat{\boldsymbol{H}}^s_v\right) \end{aligned} \tag{5}$$

The loss for a single sample (image) is defined in the following:

$$\mathcal{L}_{\text{KD}}\left(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}\right) = \|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\|_2^2 = \frac{1}{N+1}\left(\sum_{n=1}^{N} \mathcal{L}_{\text{MSE}}(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n) + \mathcal{L}_{\text{MSE}}\left(\boldsymbol{y}_{[\texttt{I\_CLS}]}, \hat{\boldsymbol{y}}_{[\texttt{I\_CLS}]}\right)\right) \tag{6}$$

We denote $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$ as the average representation for image patch i over all layers from the teacher and student model, respectively. This includes instance norm before averaging, and layer norm after averaging. For the definition of $\text{LayerNorm}(\cdot)$, see (TODO: cite notation). $\mathcal{L}_{\text{MSE}}(\cdot, \cdot)$ is the mean squared error between two vectors, defined in (TODO: cite equation).

Distillation

We distill the student model by minimizing the loss defined in Equation 6 using the AdamW optimizer [6] with a base learning rate of 5e-4. We train for 10 epochs with a batch size of 256 on the training set of ImageNet-1K [7], and run validation after every epoch on the validation set of ImageNet-1K. As Data2Vec2 our approach does not involve labels, we use the loss defined in Equation 6 also for validation. The total number of parameters involved in the distillation process is 129M, of which 43.1M trainable belong to the student model, and 85.9M frozen parameters to the teacher model.

Since we use the same architecture as Data2Vec2 for our teacher model, the images, being of size 224 $\times$224, which is the size we will consistently use for all experiments, are split into 16x16 patches, which results in a sequence length of $N = 196$. A cls token is added to the sequence, which results in a total sequence length of $N + 1 = 197$. All embeddings have a dimension of $D = 768$.

For data augmentation we decide to use the same augmentation strategy using during the training of the teacher model. This ensures that we get the training targets from the same distribution the teacher has seen, and that we do not generate data for which the teacher might give inaccurate representations. The augmentations involve (1) cropping a random portion of an image and resizing it back to the original image resolution (224$\times$224), (2) performing a random horizontal flip with probability 0.5, and (3) normalizing the image RGB channels with the mean and standard deviation of the ImageNet-1K dataset [1].

Detailed hyperparameters are provided in (TODO: cite hyperparameters).

We show the evolution of the training and validation loss during training in Figure 1. We observe the traditional convergence behavior of a model during training, and the validation loss is consistently lower than the training loss, which is a sign of good generalization. There are some peaks in the training loss, which are likely due to a high learning rate, but they do not affect the validation loss, which is why do not investigate them further. As we do not have access to any other metric than the MSE loss during training we have to evaluate the student by finetuning on downstream tasks, which follows in the next section. This will answer whether the distillation actually yields a model competetive in performance to the teacher model and if the knowledge transfer was successful.

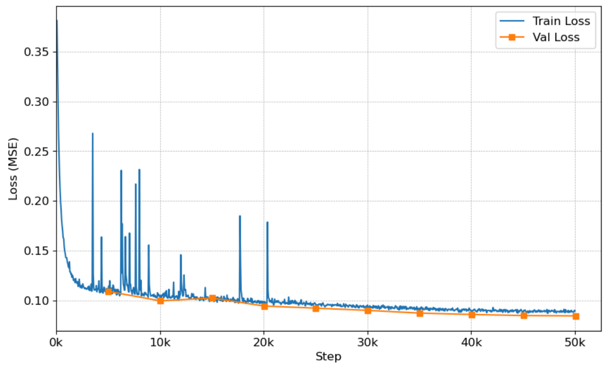

Figure 1: Training loss vs. validation loss during distillation of the Data2Vec2 image model.

Finetuning

To get a sense of how well the student model has learned from the teacher, we evaluate the student model by finetuning it on the downstream image classification tasks of CIFAR-10 [8], CIFAR-100 [8] and ImageNet-1K [7]. For that, we load the trained student model and add Layer Normalization and a linear classifier on top of it. The output of the student model is a sequence of embeddings, one embedding for each image patch, and one cls token embedding. We follow the approach of Data2Vec

[2], [1] and BEiTv2 [9], and take the mean over all patch embeddings as the output of the student model, which is then passed to the layer normalization and linear classifier (cls token embedding is ingnored). For all three tasks we perform full finetuning, i.e. we finetune all layers of the student model on the downstream task, and linear probing, we only train the added layer norm and linear classifier on top of the student model. For pytorch pseudocode of linear probing and full finetuning see (TODO: cite pseudocode).

For data augmentation during finetuning we use RandAugment [10], mixup [11] and cutmix [12] augmentation, and random erasing [13]. The hyperparameters for these augmentations are provided in (TODO: cite hyperparameters), and have been selected based on the values used in BEiTv2 [9], Data2Vec [2], and Data2Vec2 [1]. We refrain from explaining the augmentation techniques in detail here, as they are well documented in the respective papers.

For finetuning on ImageNet-1K we use a base learning rate of 1-e3 in combination with layer decay. Layer decay is a technique to reduce the base learning rate for each layer of the model by a certain factor. The goal is to have lower learning rates for layers closer to the input, and higher learning rates for layers closer to the output. This ensures that low-level features learned during pretraining or distillation are not destroyed during finetuning. We use a decay factor of 0.81, which is derived by scaling the layer decay used in Data2Vec2 [1], from which we extract layers for the student model, by the square root. We use scaling instead of the value used in Data2Vec2, which is 0.65 ($\sqrt{0.65} \approx 0.81$), as we only have half the number of layers in the student model, and can therefore afford larger learning rates for the lower layers. The actual learning rate for a layer l is then calculated by:

$$\mathrm{lr}_l = \mathrm{base\_lr} * \mathrm{layer\_decay}^{L_s+1-l} \tag{7}$$

The learning rates can be seen in the following table:

Layer no.	0	*1*	*2*	*3*	*4*	*5*	*6*	7
Learning rate	2.3e-4	2.8e-4	3.5e-4	4.3e-4	5.3e-4	6.6e-4	8.1e-4	1e-3

Table 1: Learning rates for different blocks/layers when finetuning on ImageNet-1K. Cursive layer numbers indicate Transformer blocks. The learning rates are calculated using a base learning rate of 1e-3 and a layer decay of 0.81.

We show the learning rates for 8 layers in total, even though the student model only has 6 Transformer blocks. This is because we count all weights before the first Transformer block as layer 0, which includes the weights used for projecting patches to embeddings, the cls token, and the positional encodings. Correspondingly, layer 7 includes the weights for the layer norm and linear classifier on top of the student model, which are initialized randomly and can be assigned a higher learning rate than the other layers.

For all hyperparameters used on the downstream tasks, see (TODO: cite hyperparameters).

The results, displayed in Table 2 and Table 3, show that while the student model is not able to outperform the teacher model (Data2Vec2), as well as all other models we compare to, it is able to achieve acceptable performance on all 6 evaluations considering both BEiTv2 and Data2Vec2 are based on the ViT-B/16 architecture [1], [9], which is twice as large as the student model. We also compare to the ResNet-101 model from the original ResNet paper [14], which has 44.5M parameters, and is therefore comparable in size to the student model, but has been trained only supervised.

Method	Finetune	Linear Probe
Data2Vec2	84.5	-
BEiTv2	85.0	80.1
ResNet-101	80.1	-
DistilData2Vec2 (ours)	75.0	56.2

Table 2: Comparison of finetuning and linear probing results with SOTA self-supervised models on ImageNet-1K.

Dataset	Approach	Accuracy
CIFAR-10	Finetune	97.0
	Linear Probe	68.4
CIFAR-100	Finetune	85.1
	Linear Probe	46.2

Table 3: Results of finetuning and linear probing of our distilled Data2Vec2 image model on CIFAR-10 and CIFAR-100.

Language

Method

For knowledge distillation of a language model, we decide against the intuitive choice of distilling the corresponding Data2Vec2 language model, and opt for distilling a BERT model instead. We do this for two reasons. First, for reasons later explained later, we will also use BERT as the text encoder in our multimodal model, so for consistency we will use BERT for the unimodal distillation as well. Second, as mentioned before, there already exists a distilled version of BERT, DistilBERT [3], to which we can directly compare our results.

We use the same approach as for the image model, and distill a smaller version of BERT from a pretrained BERT model. Different to DistilBERT, we again take the first 6 Transformer blocks of the teacher model, and organize them into a smaller model together with the embedding layer and positional encodings. The student model is therefore, again, initialized with a subset of the teacher's weights.

The distillation loss is defined analogously to the distilled image model, and is defined in Equation 6. We do not need to change anything, as the loss is applicable to any Transformer, regardless of the modality, making it a universal loss function for feature-based knowledge distillation. This follows Data2Vec2, which uses the same loss [1].

Distillation

The BERT model is distilled on a subset of the OpenWebText dataset [15], introduced in (TODO: cite data), of which the text is tokenized into subwords using the BERT tokenizer. During training, the tokenized text of the dataset is split into sequences of 196 tokens, which is the maximum sequence length that can fit on a single GPU with 24GB of memory (RTX 4090).

We validate the student model on the dedicated validation dataset of OWT we introduced in (TODO: cite data). The same loss as used as for training.

Finetuning

	MNLI	QNLI	RTE	MRPC	QQP	STS-B	CoLA	SST	**Score**
BERT	86.7	91.8	69.3	88.6	89.6	92.7	56.3	92.7	83.5
DistilBERT	82.2	89.2	59.9	87.5	88.5	86.9	51.3	91.3	79.6
DistilData2Vec2 (ours)	-	-	-	-	-	-	-	-	-

Table 4: Results for BERT and DistilBERT are taken from the DistilBERT paper [3].

Bibliography

[1] A. Baevski, A. Babu, W.-N. Hsu, and M. Auli, “Efficient Self-supervised Learning with Contextualized Target Representations for Vision, Speech and Language.” 2022.

[2] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, “data2vec: A general framework for self-supervised learning in speech, vision and language,” *arXiv abs/2202.03555*, 2022.

[3] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” in *NeurIPS EMC^2 Workshop*, 2019.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. doi: 10.18653/v1/N19-1423.

[5] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance Normalization: The Missing Ingredient for Fast Stylization.” [Online]. Available: https://arxiv.org/abs/1607.08022

[6] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization.” 2017.

[7] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[8] A. Krizhevsky, “Learning multiple layers of features from tiny images,” 2009.

[9] Z. Peng, L. Dong, H. Bao, Q. Ye, and F. Wei, “BEiT v2: Masked Image Modeling with Vector-Quantized Visual Tokenizers,” 2022.

[10] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “Randaugment: Practical automated data augmentation with a reduced search space,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 3008–3017. doi: 10.1109/CVPRW50498.2020.00359.

[11] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[12] S. Yun, D. Han, S. Chun, S. Oh, Y. Yoo, and J. Choe, “CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2019, pp. 6022–6031. doi: 10.1109/ICCV.2019.00612.

[13] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random Erasing Data Augmentation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 13001–13008, Apr. 2020, doi: 10.1609/aaai.v34i07.7000.

[14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. doi: 10.1109/cvpr.2016.90.

[15] A. Gokaslan and V. Cohen, “OpenWebText Corpus.” 2019.