DIVE-Doc: Downscaling foundational Image Visual Encoder into hierarchical architecture for DocVQA Supplementary Materials

Rayane Benchareff[1,2], Abderrahmane Rahiche[1], Mohamed Cheriet[1]
[1]Synchromedia Laboratory, École de Technologie Supérieure (ETS), Montreal, Canada
[2]INU Champollion, ISIS Castres, Université de Toulouse, France

Abstract

We provide here supplementary details to ensure the reproducibility of our experiments.

1. Teacher-free VS Connected-Teacher distillation

Teacher	HDD Storage	VRAM (MiB)	Required GPU	Training time (H)	Results (ANLS)
On-stream	20 (GB)	25000	3	40	81
Off-stream	200 (GB)	21300	2	36	81

Table 1. **Comparison on-stream/off-stream distillation on DocVQA dataset.** The off-stream method is less resource-constrained while achieving the same performance as on-stream distillation.

In order to save some VRAM footprints, we stored all teacher's embeddings in a database (h5 file) and created a mapping table that stores all image IDs of the original dataset and their corresponding row indices in the vector database to make the retrieval easier during the training process. Therefore, we avoid the need to load and run the teacher as it is already trained.
Table 1 shows the results of teacher-free and connected teacher approaches. As shown, both approaches achieve the same results. However, the off-stream method saves 4 hours of training and 3700 MiB of memory allocation while adding 180 GB to save the vector database on the disk. Therefore, this approach can be beneficial for laboratories or independent which have large disk storage capabilities but limited computational resources.

Email contact: rayane.benchareff.1@ens.etsmtl.ca

2. Training Hyperparameter

Table 2 provides hyperparameter details for each training stage. During the distillation stage, models are pretrained for 20 epochs with an initial learning rate of $3e{-}4$. Then for the fine-tuning stage, we train the whole models of each resolution for 3 epochs using a maximum learning rate of $3e{-}5$ and a warm-up scheduler with a ratio of 0.15. For the visual encoder evaluation, we train the added decoders of specific tasks for 5 and 3 epochs, using learning rates of $3e{-}4$ and $9e{-}4$, for document classification and layout analysis, respectively. We use Adam optimizer with $\beta = (0.9, 0.999)$, $\epsilon = 1e{-}8$ for all experiments and a weight decay of 0.01 for the fine-tuning step. We also set the batch size to 16 for all experiments.

Task/training stage	Learning Rate	Optimizer	Epochs	Batch Size
VQA (Distillation)	$3e{-}4$	Adam	20	16
VQA (Finetuning)	$3e{-}5$	Adam	3	16
Classification	$3e{-}4$	Adam	5	16
Layout Analysis	$9e{-}4$	Adam	3	16

Table 2. **Hyperparameters.** The hyperparameters used for each training stage.

3. Distillation Algorithm

Here we provide a snap-code algorithm of the distillation step (Stage I).

Algorithm 1: Distillation Training Stage

Input : Student encoder f_S, Student sequence lengths N_S, Teacher sequence length N_T, Dataset $\mathcal{D}$, Batch size B, Dataloading function $Dataloader$, Optimizer function $Adam$, Learning rate η, Epochs E, Reshape function 1d patch to 2d feature map $reshape$, Flatten function 2d feature map to 1d patch $flatten$

Output: Trained student encoder f_S^*

optimizer $\leftarrow$ Adam(f_S.parameters(), η);
dataloader $\leftarrow$ Dataloader($\mathcal{D}, B$);
if $N_S < N_T$ **then**
    $f_{\text{map}} \leftarrow$ bicubic; `// upsample mapping`
else if $N_S > N_T$ **then**
    $f_{\text{map}} \leftarrow$ bilinear; `// downsample mapping`
for *epoch = 1* **to** E **do**
    foreach *batch* (I, v^T) ***in*** *dataloader* **do**
        $v^S \leftarrow f_S.ve(I)$;
        if $N_S \neq N_T$ **then**
            $v^S \leftarrow reshape(v^S)$;
            $v^S \leftarrow f_{\text{map}}(v^S, N_T)$;
            $v^S \leftarrow flatten(v^S)$;
        $v^{S'} \leftarrow f_S.lproj(v^S)$;
        $\mathcal{L} \leftarrow$ MSE($v^{S'}, v^T$);
        optimizer.zero_grad();
        $\mathcal{L}$.backward();
        optimizer.step();
return f_S^*