Multi-Level Feature Distillation of Joint Teachers Trained on Distinct Image Datasets

Adrian Iordache[1], Bogdan Alexe[1,2], Radu Tudor Ionescu[1]
[1]Department of Computer Science, University of Bucharest, Bucharest, Romania
[2]"Gheorghe Mihoc – Caius Iacob" Institute of Mathematical Statistics and Applied Mathematics of the Romanian Academy, Bucharest, Romania.

Abstract

We propose a novel teacher-student framework to distill knowledge from multiple teachers trained on distinct datasets. Each teacher is first trained from scratch on its own dataset. Then, the teachers are combined into a joint architecture, which fuses the features of all teachers at multiple representation levels. The joint teacher architecture is fine-tuned on samples from all datasets, thus gathering useful generic information from all data samples. Finally, we employ a multi-level feature distillation procedure to transfer the knowledge to a student model for each of the considered datasets. We conduct image classification experiments on seven benchmarks, and action recognition experiments on three benchmarks. To illustrate the power of our feature distillation procedure, the student architectures are chosen to be identical to those of the individual teachers. To demonstrate the flexibility of our approach, we combine teachers with distinct architectures. We show that our novel Multi-Level Feature Distillation (MLFD) can significantly surpass equivalent architectures that are either trained on individual datasets, or jointly trained on all datasets at once. Furthermore, we confirm that each step of the proposed training procedure is well motivated by a comprehensive ablation study. We publicly release our code at `https://github.com/AdrianIordache/MLFD`.

1. Introduction

The usual paradigm in machine learning is to train a model on a single dataset with the desired goal that the trained model should be able to generalize to unseen examples for a specific task. The training algorithm and the dataset, the two key ingredients used in training the model, should be specifically tailored to the task at hand in order to facilitate obtaining a robust model that generalizes to the training data distribution. Although the size of the datasets used in training various models for different visual tasks has increased exponentially in the last twenty years, ranging from an order of tens of thousand of images in Caltech-101 [19] and PASCAL VOC 2012 [10] to hundreds of thousands of images in COCO [23] and millions of images in ImageNet [8], no dataset is sufficiently rich enough to capture the entire variability and expressiveness of the visual world. Moreover, each dataset comes with an implicit bias introduced by the underlying distribution of the sampled examples it contains [34], which might be inherited and even amplified by the trained models [36]. Thus, when trained and evaluated on different datasets, models might exhibit a significant difference in performance, showing poor generalization capacity across datasets [34]. This was even observed for large-scale benchmarks such as ImageNet [31]. One way to circumvent the problem of overfitting when training on a single dataset and build more robust models is to transfer knowledge from multiple datasets. Each dataset comes with its own view of the visual world through its training examples and holds the promise of providing a new and complementary perspective in solving a specific visual task, with respect to other datasets. To this end, we propose a Multi-Level Feature Distillation (MLFD) method to distill knowledge from multiple teachers trained on multiple distinct datasets into student models. As illustrated in Figure 1, our method is composed of three stages. In the first stage, individual teachers are trained from scratch on their own distinct dataset. In the second stage, the individual teachers are combined into a joint teacher, which is trained on all datasets. In the third stage, our method leverages the knowledge learned by the joint teacher by distilling the acquired knowledge from all datasets into dataset-specific student models. The proposed method enables distilling the knowledge using multiple levels of representations, which increases the performance of the student models.

We demonstrate the usefulness of our MLFD method by conducting image classification experiments on seven benchmarks, and action recognition experiments on three benchmarks. A direct way to take advantage of multiple datasets would be to train joint models on all datasets at once. This procedure poses an important challenge, as each dataset defines a different classification task through

2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) | 979-8-3315-1083-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/WACV61041.2025.00693

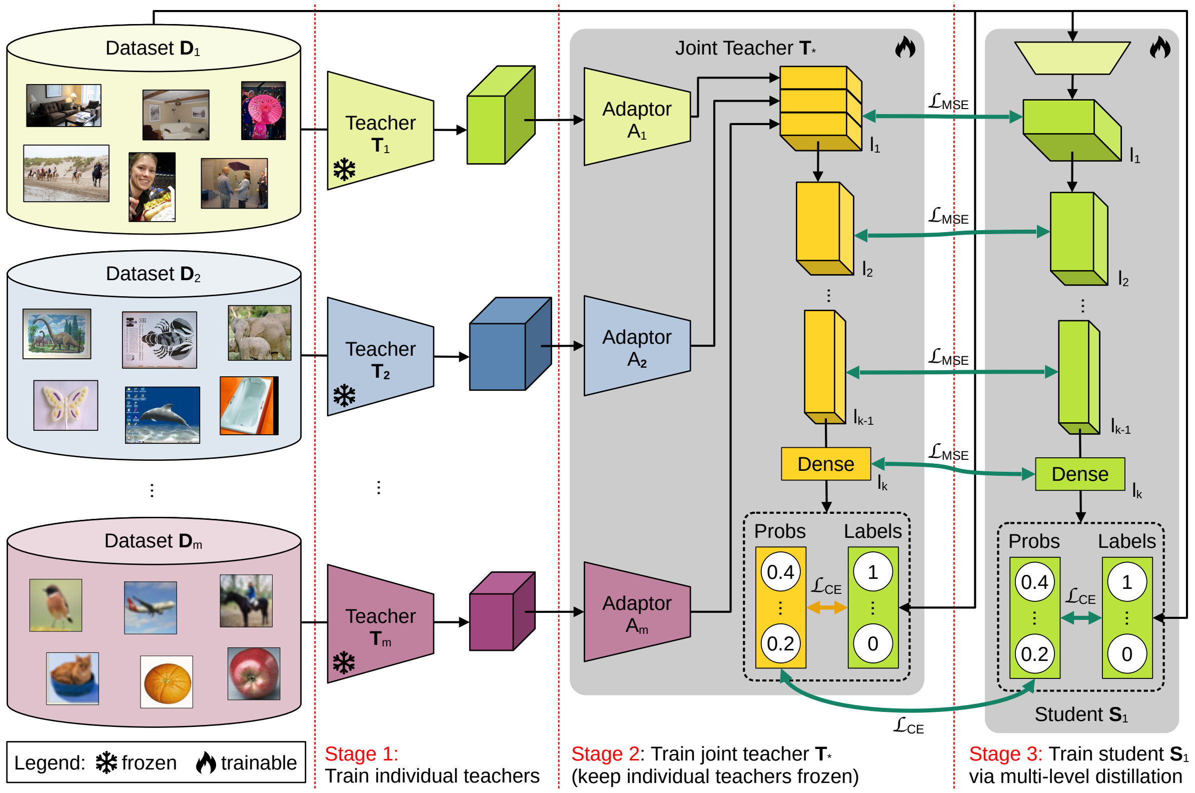

Figure 1. Our multi-level feature distillation (MLFD) framework is based on three stages. In the first stage, individual teachers are trained on each dataset. In the second stage, the individual teachers are merged at a certain representation level (l_1) into a joint teacher $\mathbf{T}_*$, which comprises levels l_1, l_2, ..., l_k. The joint teacher is trained on all datasets $\mathbf{D}_1, \mathbf{D}_2, ...\mathbf{D}_m$, while the individual teachers are kept frozen for efficiency reasons. In the third stage, each student $\mathbf{S}_i$ is trained via multi-level feature distillation from the joint teacher $\mathbf{T}_*$, for all $i \in \{1, 2, ..., m\}$. To simplify the visualization, only the first student $\mathbf{S}_1$ is illustrated in this figure. Best viewed in color.

the distinct class labels associated with each training example. We test two different baselines that mitigate this challenge, showing that our approach significantly surpasses these multi-dataset baselines. Moreover, we thoroughly evaluate our method on both simple and complex architectures.

To summarize, our contributions are the following:

- ◇ We propose a multi-level distillation method that sits at the core of a new teacher-student learning paradigm, where the knowledge acquired by several multiple teachers, grouped into a joint teacher, is distilled into dataset-specific student models.
- ◇ We show that knowledge from multiple datasets distilled into student architectures brings significant performance gains across multiple architectures, without any extra inference cost.

2. Related Work

Knowledge distillation. The goal of Knowledge Distillation (KD) is to transfer knowledge from a large teacher model to a small student model, such that the obtained student model mimics the behavior of the larger model [4, 5, 13, 15, 26, 29]. The work of [13] defines the knowledge in the form of the output of the large model (logits). The basic form of knowledge distillation consists in training the small student model to reproduce the logits of the large teacher model. This can be done either in the unsupervised [1, 18] or supervised [22] scenario. In the supervised scenario, with ground-truth labels available at training time, Hinton *et al.* [13] show that significant improvements can be obtained by minimizing an objective function that takes into account the cross-entropy between the logits of the two models, but also another term that enforces the student model to predict ground-truth labels. Other works [3, 32, 40] consider knowledge at the feature level, employing distillation by matching feature distributions as well as logit outputs between the teacher and student models. Romero *et al.* [32] use intermediate-level hints from the teacher hidden layers to guide the training process of the student, enforcing a deeper and thinner student network to learn an intermediate representation that is predictive of the intermediate representations of the wide and shallower (but still deep) teacher network. Park *et al.* [30] transfer knowledge in the form of mutual relations of data examples, by including distance-wise and angle-wise distillation losses that penalize structural differences in relations. Li [28] reuses channel-wise and layer-wise meaningful features within the student to provide teacher-like knowledge without an additional model, in a teacher-free feature distillation framework. Liu *et al.* [24] propose a two-stage knowledge distillation method, which relies on a simple Feature Transform module consisting of two linear layers. Our proposed multi-level feature distillation method considers knowledge both at the logits level and feature level, but, unlike other approaches, it uses multiple teachers, each of them trained on a distinct dataset. Usually, knowledge distillation is applied for model compression [13], where the teacher model has a much larger capacity and memory footprint w.r.t. the student model. Our method allows student architectures to be identical to those of the individual teachers. Moreover, our method is very flexible, since it can combine teachers with distinct architectures.

A different strand of work uses search techniques, such as Monte Carlo tree search [21] or evolutionary search [9, 20], to investigate optimal knowledge distillation designs for distilling a given teacher-student. These approaches are orthogonal to our work, and their combination can further improve the result of knowledge distillation.

Dataset distillation. The goal of dataset distillation [2, 38] is to synthesize an original large dataset with a small dataset, such that models trained on the smaller dataset exhibit similar performance to models trained on the original one. Although our framework enables using multiple distinct datasets, we do not aim to compress these datasets. Instead, we distill knowledge from a joint teacher obtained by combining several individual teachers, each trained on a distinct dataset. This is an orthogonal approach to dataset distillation.

3. Method

We propose a new multi-level distillation method to improve the generalization capacity and performance of multiple models trained on distinct datasets, at no additional costs during inference time. We achieve this by combining multiple teacher models, each trained on a distinct dataset, into a joint teacher, which is trained on data samples from the combined datasets. Then, the knowledge from the joint teacher is distilled into dataset-specific students. The distillation step is carried out at multiple representation levels, resulting in a multi-level feature distillation (MLFD) framework. Not only do the students receive information from different teachers, but they also capture information from distinct datasets, which significantly boosts their generalization capacities. Our MLFD framework is formally presented in Algorithm 1.

The MLFD algorithm essentially takes as input a set of datasets $\mathcal{D}$, a set of teacher architectures $\mathcal{T}$, and a set of student architectures $\mathcal{S}$, where $|\mathcal{D}| = |\mathcal{T}| = |\mathcal{S}| = m$. We underline that each dataset can comprise a different number of images. Moreover, the teacher and student architectures can be diverse, which enables the use of a suitable architecture for each specific dataset. There are some additional hyperparameters given as input, such as the learning rates and the set of layer indexes $\mathbf{L}$ at which the multi-level fusion and distillation is performed.

In steps 1-7, we train the set of teachers from scratch via a variant of stochastic gradient descent (SGD), such that each teacher is trained on its own dataset. Since our algorithm does not impose the use of the same architecture for every teacher, we are free to harness existing pre-trained architectures from the public web domain. This means that the individual teacher training step can be omitted, whenever this is desired.

In step 8, the individually trained teachers are fused into a joint architecture, denoted as $\mathbf{T}_*$. The fusion is performed at a certain representation level, denoted as l_1. To simplify the notations, we use l_1 to denote the index of the layer that is fused from each teacher, disregarding the fact that l_1 can actually represent a different index for each teacher architecture. Formally, the *fuse* function used in step 8 of Algorithm 1 is defined as follows:

$$\mathbf{T}_* = \text{NN}^{l_k}\Big(\dots\ \text{NN}^{l_2}\Big(\text{NN}^{l_1}\Big(\boldsymbol{e}_j^{\mathbf{T}_*^{l_1}}, \theta_{\mathbf{T}_*}^{l_1}\Big), \theta_{\mathbf{T}_*}^{l_2}\Big) \dots, \theta_{\mathbf{T}_*}^{l_k}\Big), \quad (1)$$

where NN^{l_c} is a neural network block that transforms the features at layer l_c into a set of features that are given as input to layer l_{c+1}, $\theta_{\mathbf{T}_*}^{l_c}$ are the features of the neural block NN^{l_c}, and j iterates through all images in $\mathcal{D}$. We note that the neural blocks NN^{l_c} are typically shallow, comprising either a conv and a pooling operation, or one or two dense layers. The type of block (convolutional or dense) depends on the desired architectural design of the joint teacher. The embedding $\boldsymbol{e}_j^{\mathbf{T}_*^{l_1}}$ given as input to layer l_1 of the joint teacher $\mathbf{T}_*$, corresponding to the j-th training image, is computed as follows:

$$\boldsymbol{e}_j^{\mathbf{T}_*^{l_1}} = \Big[A_1\Big(\boldsymbol{e}_j^{\mathbf{T}_1^{l_1}}, \theta_{A_1}\Big), A_2\Big(\boldsymbol{e}_j^{\mathbf{T}_2^{l_1}}, \theta_{A_2}\Big), ..., A_m\Big(\boldsymbol{e}_j^{\mathbf{T}_m^{l_1}}, \theta_{A_m}\Big)\Big], \quad (2)$$

where $[\cdot, \cdot]$ is the channel-wise concatenation operation, $\boldsymbol{e}_j^{\mathbf{T}_1^{l_1}}, \boldsymbol{e}_j^{\mathbf{T}_2^{l_1}}, ..., \boldsymbol{e}_j^{\mathbf{T}_m^{l_1}}$ are the embeddings returned by the individual teachers for the j-th image, and $A_1, A_2, ..., A_m$ are adaptor blocks that bring the features at layer l_1 from all teachers to the same spatial dimensions, so that they can be concatenated. Each adaptor block A_i is parameterized by the weights $\theta_{A_i}, \forall i \in \{1, 2, ..., m\}$.

In steps 9-14, we train the joint teacher on all datasets, via SGD. We note that the trainable weights $\theta_{\mathbf{T}_*}$ of the joint teacher are composed of the weights that correspond to the neural blocks NN^{l_c}, denoted by $\theta_{\mathbf{T}_*}^{l_c}$, and the weights that correspond to the adaptor blocks A_i, denoted by θ_{A_i}. In

Algorithm 1: Multi-Level Feature Distillation

```
Input: $\mathcal{D} = \{\mathbf{D}_i | \mathbf{D}_i = \{(\boldsymbol{I}_1^i, y_1^i), (\boldsymbol{I}_2^i, y_2^i)..,(\boldsymbol{I}_{n_i}^i, y_{n_i}^i)\}, \forall i \in \{1,2,...,m\}\}$ - the set of training sets of labeled images,
    $\mathcal{T} = \{\mathbf{T}_1, \mathbf{T}_2, ..., \mathbf{T}_m\}$ - the set of teacher models, $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_m\}$ - the set of student models, $\eta_{\mathbf{T}_i}$ - the learning rate
    of teacher $\mathbf{T}_i$, $\eta_{\mathbf{T}_*}$ - the learning rate of the joint teacher, $\eta_{\mathbf{S}_i}$ - the learning rate of the student $\mathbf{S}_i$, $\mathbf{L} = \{l_1, l_2, ..., l_k\}$ - the
    set of layer indexes at which the multi-level fusion and distillation is performed.
Output: $\theta_{\mathbf{S}_i}, \forall i \in \{1,2,...,m\}$ - the trained weights of each student model $\mathbf{S}_i$.
1  foreach $i \in \{1,2,...,m\}$ do
2      $\theta_{\mathbf{T}_i} \sim \mathcal{N}\left(0, \frac{2}{d_{in_i}+d_{out_i}}\right)$ ; ◁ initialize weights of teacher $\mathbf{T}_i$ using Xavier [11]
3      repeat
4          foreach $j \in \{1,2,...,n_i\}$ do
5              $t_j \leftarrow \mathbf{T}_i(\boldsymbol{I}_j^i, \theta_{\mathbf{T}_i})$; ◁ get class probabilities predicted by teacher $\mathbf{T}_i$
6              $\theta_{\mathbf{T}_i} \leftarrow \theta_{\mathbf{T}_i} - \eta_{\mathbf{T}_i} \cdot \nabla \mathcal{L}_{\text{CE}}(y_j^i, t_j)$; ◁ train the teacher $\mathbf{T}_i$
7      until convergence;
8  $\mathbf{T}_* \leftarrow fuse(\mathcal{T}, \mathbf{L})$ ; ◁ apply the fusion in Eq. (1) to obtain joint teacher
9  repeat
10     foreach $i \in \{1,2,...,m\}$ do
11         foreach $j \in \{1,2,...,n_i\}$ do
12             $t_j \leftarrow \mathbf{T}_*(\boldsymbol{I}_j^i, \theta_{\mathbf{T}_*})$; ◁ get class probabilities predicted by teacher $\mathbf{T}_*$
13             $\theta_{\mathbf{T}_*} \leftarrow \theta_{\mathbf{T}_*} - \eta_{\mathbf{T}_*} \cdot \nabla \mathcal{L}_{\text{CE}}(y_j^i, t_j)$; ◁ train the joint teacher $\mathbf{T}_*$
14 until convergence;
15 foreach $i \in \{1,2,...,m\}$ do
16     $\theta_{\mathbf{S}_i} \sim \mathcal{N}\left(0, \frac{2}{d_{in_i}+d_{out_i}}\right)$ ; ◁ initialize weights of student $\mathbf{S}_i$ using Xavier [11]
17     foreach $j \in \{1,2,...,n_i\}$ do
18         $t_j, \boldsymbol{e}_j^{\mathbf{T}_*^{l_1}}, ..., \boldsymbol{e}_j^{\mathbf{T}_*^{l_k}} \leftarrow \mathbf{T}_*(\boldsymbol{I}_j^i, \theta_{\mathbf{T}_*})$; ◁ get class probabilities and multi-level embeddings from joint teacher
19     repeat
20         foreach $j \in \{1,2,...,n_i\}$ do
21             $s_j, \boldsymbol{e}_j^{\mathbf{S}_i^{l_1}}, ..., \boldsymbol{e}_j^{\mathbf{S}_i^{l_k}} \leftarrow \mathbf{S}(\boldsymbol{I}_j^i, \theta_{\mathbf{S}_i})$; ◁ get class probabilities and multi-level embeddings from student $\mathbf{S}_i$
22             $\mathcal{L}_{\text{KD}} \leftarrow \mathcal{L}_{\text{CE}}(y_j^i, s_j) + \alpha \cdot \mathcal{L}_{\text{CE}}^{\mathbf{T}_*}(t_j, s_j) + \sum_{c=1}^{k} \beta_c \cdot \mathcal{L}_{\text{MSE}}^{\mathbf{T}_*}\left(\boldsymbol{e}_j^{\mathbf{T}_*^{l_c}}, \boldsymbol{e}_j^{\mathbf{S}_i^{l_c}}\right)$; ◁ apply Eq. (3)
23             $\theta_{\mathbf{S}_i} \leftarrow \theta_{\mathbf{S}_i} - \eta_{\mathbf{S}_i} \cdot \nabla \mathcal{L}_{\text{KD}}$; ◁ train the student using the joint loss
24     until convergence;
```

other words, the weights of individual teachers that precede the fusion layer are kept frozen. This allows us to extract the features $\boldsymbol{e}_j^{\mathbf{T}_i^{l_1}}$ of the j-th training image in one go, before training the joint teacher, which significantly reduces the training time and the memory footprint, while training the joint teacher.

After training the joint teacher, it is time to distill its knowledge into the students. Our multi-level distillation is performed in steps 15-24 of Algorithm 1. In step 16, we initialize the weights of the student $\mathbf{S}_i$. Then, in steps 17-18, we extract the embeddings at multiple representation levels from the joint teacher $\mathbf{T}_*$, where the chosen representation levels are specified through the set $\mathbf{L}$. The embeddings $\boldsymbol{e}_j^{\mathbf{T}_*^{l_c}}$ returned by the joint teacher in step 18 represent the features used inside the block NN^{l_c}, where $c \in \{1,2,...,k\}$. Finally, in steps 19-24, we distill the knowledge learned by the joint teacher into the student $\mathbf{S}_i$. The student is trained end-to-end (there are no frozen weights) via SGD. To perform the multi-level knowledge distillation, we employ the following loss function:

$$\mathcal{L}_{\text{KD}} = \mathcal{L}_{\text{CE}}(y_j^i, s_j) + \alpha \cdot \mathcal{L}_{\text{CE}}^{\mathbf{T}_*}(t_j, s_j) + \sum_{c=1}^{k} \beta_c \cdot \mathcal{L}_{\text{MSE}}^{\mathbf{T}_*}\left(\boldsymbol{e}_j^{\mathbf{T}_*^{l_c}}, \boldsymbol{e}_j^{\mathbf{S}_i^{l_c}}\right), \quad (3)$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy (CE) loss between the ground-truth one-hot class label y_j^i and the class probabilities s_j predicted by the student, $\mathcal{L}_{\text{CE}}^{\mathbf{T}_*}$ is the cross-entropy loss between the class probabilities t_j returned by the joint teacher and those of the student, and $\mathcal{L}_{\text{MSE}}^{\mathbf{T}_*}$ is the mean squared error (MSE) between the embedding $\boldsymbol{e}_j^{\mathbf{T}_*^{l_c}}$ at layer l_c returned by the joint teacher and the embedding $\boldsymbol{e}_j^{\mathbf{S}_i^{l_c}}$ at layer l_c returned by the student, respectively. In general, $\boldsymbol{e}_j^{\mathbf{T}_*^{l_c}}$ might not have compatible dimensions with $\boldsymbol{e}_j^{\mathbf{S}_*^{l_c}}$. When this happens, an adaptor layer can be inserted in order to bring the shape of $\boldsymbol{e}_j^{\mathbf{T}_*^{l_c}}$ to the desired size. The hyperparameters $\alpha \geq 0$ and $\beta_c \geq 0$ control the importance of the distillation losses with respect to the classification loss. Note that the loss at a certain representation level c has its own contribution to the overall loss defined in Eq. (3), which is determined by β_c. For optimal results, these hyperparameters can be fine-tuned on the validation set of each dataset $\mathbf{D}_i \in \mathcal{D}$. The algorithm returns the weights $\theta_{\mathbf{S}_i}$ for each student $\mathbf{S}_i \in \mathcal{S}$.

4. Experiments

4.1. Datasets

We next present experiments on three datasets for image classification: CIFAR-100 [17], ImageNet-Sketch [37] and TinyImageNet [8]. Results on additional datasets and tasks are discussed in the supplementary.

CIFAR-100. The CIFAR-100 dataset consists of $60,000$ color images of 32×32 pixels, grouped into 100 classes. We use the official split with 500 training images and 100 testing images per class, for a total of $50,000$ training images and $10,000$ test images.

ImageNet-Sketch. The ImageNet-Sketch dataset consists of black and white images from 1000 classes, corresponding to the ImageNet validation classes. The dataset is intended to be used for evaluating the ability of models to generalize to out-of-domain data, represented by sketch-like images. The dataset comprises $41,088$ training images and $10,752$ test images.

TinyImageNet. The TinyImageNet dataset is a subset of the ImageNet [8] dataset, containing $100,000$ training images from 200 object classes. All images have a resolution of 64×64 pixels. As the labels for official test images are not publicly available, we use the $10,000$ validations images as test data.

4.2. Baselines

In our experiments, we consider three types of baselines that are described below.

Dataset-specific models. As primary baselines for our method, we consider models that have identical architectures to our student models. These baseline models are trained from scratch on individual datasets, so they do not benefit from information gathered from other datasets. An accuracy boost over these baselines will demonstrate the benefit of our multi-dataset distillation approach.

Multi-head multi-dataset models. As a second baseline, we propose an architecture composed of a shared backbone and an independent prediction head for each dataset. During every iteration, we train our model by sequentially forwarding a batch from every dataset, and accumulating the gradients from all batches to be updated at once. Each batch is equally weighted in the final loss.

Joint-head multi-dataset models. The third baseline is an alternative approach for joint dataset training. It is represented by a model with a single classification head, containing the total number of classes across all datasets. During training, each batch contains a mixture of images from all the available datasets, where each label is a one-hot vector over the joint set of classes from all datasets.

4.3. Experimental Setup

We evaluate our method on two sets of individual teachers, denoted as $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. $\mathcal{T}_1$ contains three models, namely a ResNet-18 [12] trained on CIFAR-100, an EfficientNet-B0 [33] trained on TinyImageNet, and a SEResNeXt-26D [14] trained on ImageNet-Sketch. $\mathcal{T}_2$ contains a different lineup of models, namely a ConvNeXt-V2 Tiny [39] trained on CIFAR-100, a SwinTransformer-V2 Tiny [25] trained on TinyImageNet, and a FastViT SA24 [35] trained on ImageNet-Sketch. In summary, $\mathcal{T}_1$ and $\mathcal{T}_2$ contain a variety of state-of-the-art image classification models, being based on both convolutional and transformer architectures. This is to demonstrate that the individual teachers can have distinct architectures. To demonstrate the full power of our multi-dataset distillation pipeline, we train the individual teachers from scratch and refrain from using weights pre-trained on ImageNet. Moreover, the student models, as well as the dataset-specific baseline models, have identical architectures to the individual teachers in $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. This allows us to attribute the reported accuracy improvements to our method, and not to the architectural choices.

We run the experiments on a machine with an NVIDIA GeForce RTX 3090 GPU, an Intel i9-10940X 3.3 GHz CPU, and 128 GB of RAM. Due to hardware limitations, we employ techniques such as gradient accumulation, mixed precision, and offline distillation. For the same reason, we cache the embeddings from the layers at index l_1 of the individual teachers, before training the joint teacher model.

4.4. Hyperparameter Tuning

We train each model from $\mathcal{T}_1$ and $\mathcal{T}_2$ in a different manner, adapted to the model and the target dataset. For models in $\mathcal{T}_1$, we apply a training procedure based on a constant learning rate, using the Adam optimizer [16] with weight decay for regularization, and the AutoAugment [6] with ImageNet policy for augmentation. We tried replacing the constant learning rate with one-cycle learning for all models in $\mathcal{T}_1$, without observing any significant improvements. All models are trained for 200 epochs on images with a resolution of 224×224 pixels.

For models in $\mathcal{T}_2$, each architecture has a specific training recipe described in the corresponding original papers. We start with almost the same hyperparameters, and make minimal changes in order to adapt the hyperparameters to each

Architecture	Learning rate	Batch size	Stochastic depth	Dropout	Weight decay	Temperature	α and β_c coefficients
ResNet-18	1e-4	768	0	-	0.001	2	0.6, 0.2, 0.2
EfficientNet-B0	1e-4	240	0.2	0.2	0.001	2	0.6, 0.2, 0.2
SEResNeXt-26D	1e-4	192	0	-	0.001	2	0.6, 0.2, 0.2
ConvNeXt-V2 Tiny	1e-5	96	0.6	-	0.5	5	0.6, 02, 0.2
SwinTransformer-V2 Tiny	4e-4	176	0.6	-	0.5	2	0.6, 0.2, 0.2
FastViT SA24	1e-4	102	0.3	0.2	0.5	2	0.4, 0.4, 0.2

Table 1. Optimal hyperparameter settings for the various neural architectures used in our experiments.

Model	Distillation levels	CIFAR-100		TinyImageNet		ImageNet-Sketch	
		`acc@1`	`acc@5`	`acc@1`	`acc@5`	`acc@1`	`acc@5`
Dataset-specific models	-	55.07	81.62	45.09	70.52	49.67	71.39
Multi-head model	-	36.61	65.00	35.26	63.64	21.19	43.47
Joint-head model	-	59.46	85.98	45.13	73.09	41.59	66.79
Students (our)	$\mathbf{L}_1$	58.65	83.97	50.86	76.07	51.64	72.81
	$\mathbf{L}_2$	**62.25**	**84.64**	**51.61**	**76.46**	**61.31**	**78.36**
Joint teacher (ours)	$\mathbf{L}_1$	61.00	85.44	46.97	**72.53**	56.35	75.60
	$\mathbf{L}_2$	**63.34**	**87.03**	**47.78**	71.71	**61.45**	**79.38**

Table 2. Results for the set $\mathcal{T}_1$, including the three baselines, the students obtained by employing multi-level distillation using embeddings extracted at one ($\mathbf{L}_1$) or two levels ($\mathbf{L}_2$), and the corresponding joint teachers. The results of the best student and the best teacher are highlighted in bold.

dataset, aiming for a better performance. Several choices led to significant improvements of each model: adding stochastic depth, label smoothing, and dropout in the final layers. The models in $\mathcal{T}_2$ are trained with AdamW [27] and weight decay. We tried several well-established ImageNet augmentation techniques such as RandAugment [7], MixUp [42], CutMix [41], and RandomErasing [43], but all of them led to performance drops. Thus, we settle for using the AutoAugment technique. To maintain compatibility with the included transformer models, we use 256×256 input images for all models in $\mathcal{T}_2$. Regarding the learning rate schedule, the one-cycle procedure provides a boost in performance for the second set of models, but downgrades the quality of fused features leading to worse performance in the joint teacher. For that reason, we either maintain the learning rate constant or employ a one-cycle learning rate procedure with the final learning rate having the same value as the constant one.

In Table 1, we report specific values of optimal hyperparameter choices. All other hyperparameters are set to their default values.

4.5. Joint Teacher Architecture

We explore several architectures for the joint teacher, depending on the type of layers that need to be fused. When combining embeddings from dense layers of individual teachers, we initially explored variations of feed-forward layers, 1D convolutional layers, and multi-head attention layers. For each alternative, we considered versions with and without residual connections. However, the best results were obtained by using a single highly regularized linear layer to project the concatenated embedding vectors from our individual teachers. To regulate the single layer, we employ dropout with a probability ranging between 0.8 and 0.9. Mixing embeddings from convolutional or transformer layers of individual teachers is typically performed via pointwise convolutional layers. Subsequent layers in the joint teacher follow a basic architecture comprising convolutional layers, batch normalization, and adaptive average pooling. Gaussian error linear unit (GeLU) is the activation function employed in all neurons. A dense layer with a high dropout rate is added just before the classification head.

In general, the depth of the trainable joint teacher is chosen such that, when combined with the frozen layers of the individual teachers, it results in a depth that roughly matches that of the individual teachers. Although a deeper joint teacher might bring additional performance gains, we refrain from using deeper or complex joint teachers to ensure that the reported performance gains are due to the proposed training procedure, and not due to the higher capability of the joint teacher architecture.

4.6. Results

We evaluate the performance of our approach in terms of two metrics: top-1 accuracy (`acc@1`) and top-5 accuracy (`acc@5`). We present quantitative results of our experiments in Table 2 with models from $\mathcal{T}_1$, and Table 3 with models from $\mathcal{T}_2$. Let $\mathbf{L}_k = \{l_1, l_2, \ldots, l_k\}$ denote the set

Model	Distillation levels	CIFAR-100 acc@1	CIFAR-100 acc@5	TinyImageNet acc@1	TinyImageNet acc@5	ImageNet-Sketch acc@1	ImageNet-Sketch acc@5
Dataset-specific models	-	65.50	87.00	62.91	83.66	67.36	81.59
Multi-head model	-	65.12	88.66	56.66	81.37	38.07	62.12
Joint-head model	-	64.43	88.48	55.52	79.94	63.40	78.78
Students (our)	$\mathbf{L}_1$	68.81	90.79	**66.78**	**87.10**	69.31	**84.79**
	$\mathbf{L}_2$	**71.23**	**91.73**	66.30	86.84	**69.53**	84.13
Joint teacher (ours)	$\mathbf{L}_1$	67.64	**89.81**	64.08	**85.44**	68.30	**82.90**
	$\mathbf{L}_2$	**70.16**	88.99	**65.02**	84.08	**68.35**	82.69

Table 3. Results for the set $\mathcal{T}_2$, including the dataset-specific baselines, the students obtained by employing multi-level distillation using embeddings extracted at one ($\mathbf{L}_1$) or two levels ($\mathbf{L}_2$), and the corresponding joint teachers. The results of the best student and the best teacher are highlighted in bold.

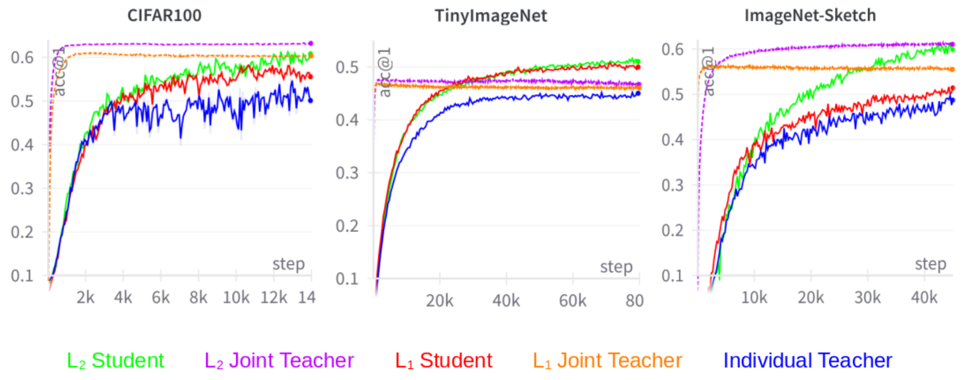

Figure 2. Top-1 accuracy evolution during the training process for models in $\mathcal{T}_1$. Best viewed in color.

of k representation levels used in our multi-level distillation scheme. Thus, $\mathbf{L}_1$ and $\mathbf{L}_2$ represent sets of one and two representation levels, respectively.

Students vs baselines. In terms of the top-1 accuracy, *all* students surpass the performance of the dataset-specific models. The results are consistent over all three datasets and over all models in $\mathcal{T}_1$ and $\mathcal{T}_2$. The improvements range between 2% and 12%, depending on the number of representations levels. With respect to the dataset-specific models, our student models yield an average improvement of 3.4% when the distillation uses a single representation level ($\mathbf{L}_1$), and 6.1% when the distillation is performed at two representation levels ($\mathbf{L}_2$). We also consider two baselines jointly trained over all three datasets, as explained earlier. The multi-head multi-dataset model performs much worse than the dataset-specific models, while the joint-head multi-dataset model is comparable with the dataset-specific models. Both multi-dataset baselines are clearly outperformed by the students trained with our novel distillation method, with average improvements ranging between 5% and 27% in favor of our method.

Importance of multi-level distillation. In five out of six cases, student models based on two-level distillation ($\mathbf{L}_2$) outperform those based on one-level distillation ($\mathbf{L}_1$), showing that employing knowledge distillation at multiple representations levels is beneficial. Notably, the students based on two-level distillation usually surpass the corresponding joint teacher models, with an average performance gain of 1%.

Performance evolution during training. Figures 2 and 3 show the performance evolution of models from $\mathcal{T}_1$ and $\mathcal{T}_2$ during training, in terms of the top-1 accuracy. The plots show the accuracy on test data, after each training epoch. Both joint teachers converge very fast, while the corresponding students begin at the same pace with the individual teachers, but after a few epochs, they pick out the knowledge via the multi-level feature distillation method, and quickly catch up with the joint teachers. Interestingly, the student based on $\mathbf{L}_2$ is almost equal to its joint teacher and always outperforms the joint teacher based on $\mathbf{L}_1$, showing, once again, the advantage of using multi-level representations in our framework.

4.7. Ablation Study

We motivate the choice of the set of layer indexes $\mathbf{L}_i$ at which the multi-level feature fusion and distillation is performed in our method through an ablation study, using models in the set $\mathcal{T}_1$. We evaluate the performance of the joint teachers considering representation levels in the following sets: $\mathbf{L}_1$, $\mathbf{L}_2$, $\mathbf{L}_3$ and $\mathbf{L}_4$. $\mathbf{L}_1$ contains embedding vectors

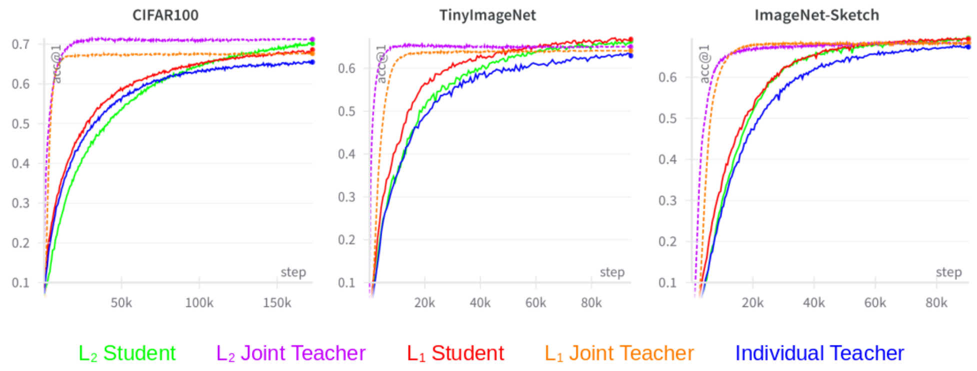

Figure 3. Top-1 accuracy during the training process for models in $\mathcal{T}_2$. Best viewed in color.

Model	Representation levels	CIFAR-100		TinyImageNet		ImageNet-Sketch	
		acc@1	acc@5	acc@1	acc@5	acc@1	acc@5
Dataset-specific models	-	55.07	81.62	45.09	70.52	49.67	71.39
Joint teacher	$\mathbf{L}_1$	60.90	84.56	47.10	72.43	54.85	75.20
	$\mathbf{L}_2$	**62.28**	**85.75**	**47.52**	**73.24**	**61.81**	**78.94**
	$\mathbf{L}_3$	62.12	85.71	44.10	70.40	55.72	76.28
	$\mathbf{L}_4$	48.65	78.78	29.93	58.01	30.69	47.50

Table 4. Results with multiple variants of joint teachers build with models from $\mathcal{T}_1$, obtained by varying the representation levels from $\mathbf{L}_1$ to $\mathbf{L}_4$.

from the last layer. $\mathbf{L}_2$ is obtained by adding feature maps of size 7×7 to $\mathbf{L}_1$. Similarly, $\mathbf{L}_3$ is obtained by adding feature maps of size 14×14 to $\mathbf{L}_2$, and $\mathbf{L}_4$ is obtained by adding feature maps of size 28×28 to $\mathbf{L}_3$. As shown in Table 4, training joint teachers based on features extracted from layers closer to the input ($\mathbf{L}_3$ or $\mathbf{L}_4$) results in poor performance. Results are typically better when considering only a reduced number of layers, *i.e*. the last layer ($\mathbf{L}_1$) or the last two layers ($\mathbf{L}_2$) before the classification head. A possible explanation would be that layers farther from the output learn more generic (low-level) features, which do not bring new (dataset-specific) information into the joint teacher. Figure 4 illustrates the evolution of the top-1 accuracy of the joint teachers learned in these four scenarios, confirming our observations.

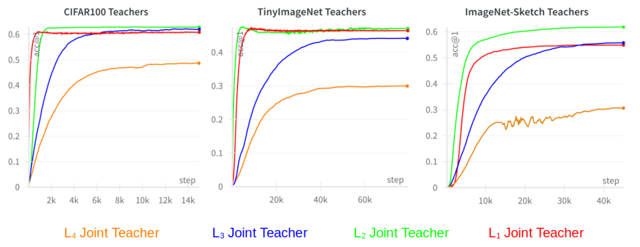

Figure 4. Performance evolution of joint teachers when using different sets of layers (from $\mathbf{L}_1$ to $\mathbf{L}_4$) to extract features. Best viewed in color.

4.8. Supplementary

In the supplementary material, we show that our method applies to a distinct collection of datasets, as well as a different task (action recognition). Moreover, we show that the feature space learned by our framework is more discriminative via a t-SNE visualization of the embeddings. We also discuss the limitations of our method.

5. Conclusion

In this paper, we presented a novel distillation method, coined Multi-Level Feature Distillation, that uses multiple teacher models combined into a joint teacher to distill the knowledge acquired at multiple representation levels from distinct datasets into student models. Our experimental evaluation for the task of image classification on three benchmark datasets demonstrated the benefits of our novel method in increasing the accuracy of several different state-of-the-art models, that are based on convolutional or transformer architectures. Additionally, we presented t-SNE visualizations that show the effectiveness of the knowledge distillation approach in learning robust latent representations, and explaining the superior results of our student models.

In future work, we aim to apply our method beyond image classification and explore new vision tasks, *e.g*. object detection and image segmentation, as well as new domains, *e.g*. natural language processing.

References

[1] Yutong Bai, Zeyu Wang, Junfei Xiao, Chen Wei, Huiyu Wang, Alan L. Yuille, Yuyin Zhou, and Cihang Xie. Masked autoencoders enable efficient knowledge distillers. In *Proceedings of CVPR*, pages 24256–24265, 2023. 2

[2] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of CVPR*, pages 10718–10727, 2022. 3

[3] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *Proceedings of CVPR*, 2021. 2

[4] Xu Cheng, Zhefan Rao, Yilan Chen, and Quanshi Zhang. Explaining knowledge distillation by quantifying the knowledge. In *Proceedings of CVPR*, pages 12925–12935, 2020. 2

[5] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of ICCV*, pages 4793–4801, 2019. 2

[6] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Strategies From Data. In *Proceedings of CVPR*, pages 113–123, 2019. 5

[7] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In *Proceedings of NeurIPS*, volume 33, pages 18613–18624, 2020. 6

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database, year=2009. In *Proceedings of CVPR*, pages 248–255. 1, 5

[9] Peijie Dong, Lujun Li, and Zimian Wei. DisWOT: Student Architecture Search for Distillation WithOut Training. In *Proceedings of CVPR*, pages 11898–11908, 2023. 3

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 1

[11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256, 2010. 4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of CVPR*, pages 770–778, 2016. 5

[13] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 3

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *Proceedings of CVPR*, pages 7132–7141, 2018. 5

[15] Guangda Ji and Zhanxing Zhu. Knowledge distillation in wide neural networks: Risk bound, data efficiency and imperfect teacher. In *Proceedings of NeurIPS*, volume 33, pages 20823–20833, 2020. 2

[16] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *Proceedings of ICLR*, 2015. 5

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 5

[18] Shanshan Lao, Guanglu Song, Boxiao Liu, Yu Liu, and Yujiu Yang. Masked autoencoders are stronger knowledge distillers. In *Proceedings of ICCV*, pages 6384–6393, 2023. 2

[19] Fei-Fei Li, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006. 1

[20] Lujun Li, Peijie Dong, Anggeng Li, Zimian Wei, and Yang Ya. KD-zero: Evolving knowledge distiller for any teacher-student pairs. In *Proceedings of NeurIPS*, pages 69490–69504, 2023. 3

[21] Lujun Li, Peijie Dong, Zimian Wei, and Ya Yang. Automated Knowledge Distillation via Monte Carlo Tree Search. In *Proceedings of ICCV*, pages 17413–17424, October 2023. 3

[22] Han Lin, Guangxing Han, Jiawei Ma, Shiyuan Huang, Xudong Lin, and Shih-Fu Chang. Supervised masked knowledge distillation for few-shot transformers. In *Proceedings of CVPR*, pages 19649–19659, 2023. 2

[23] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. *arXiv preprint arXiv:1405.0312*, 2014. 1

[24] Xiaolong Liu, Lujun Li, Chao Li, and Anbang Yao. NORM: Knowledge Distillation via N-to-One Representation Matching. In *Proceedings of ICLR*, 2023. 3

[25] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of CVPR*, pages 12009–12019, 2022. 5

[26] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015. 2

[27] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proceedings of ICLR*, 2019. 6

[28] Li Lujun. Self-regulated feature learning via teacher-free feature distillation. In *Proceedings of ECCV*, pages 347–363, 2022. 3

[29] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of AAAI*, volume 34, pages 5191–5198, 2020. 2

[30] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of CVPR*, pages 3967–3976, 2019. 2

[31] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *Proceedings of ICML*, pages 5389–5400, 2019. 1

[32] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for Thin Deep Nets. In *Proceedings of ICLR*, 2015. 2

[33] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of ICML*, pages 6105–6114, 2019. 5

[34] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *Proceedings of CVPR*, 2011. 1

[35] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. FastViT: A Fast Hybrid Vision Transformer Using Structural Reparameterization. In *Proceedings of ICCV*, pages 5785–5795, 2023. 5

[36] Angelina Wang, Arvind Narayanan, and Olga Russakovsky. REVISE: A tool for measuring and mitigating bias in visual datasets. In *Proceedings of ECCV*, pages 1790–1810, 2020. 1

[37] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P. Xing. Learning robust global representations by penalizing local predictive power. In *Proceedings of NeurIPS*, pages 10506–10518, 2019. 5

[38] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018. 3

[39] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-Designing and Scaling ConvNets With Masked Autoencoders. In *Proceedings of CVPR*, pages 16133–16142, 2023. 5

[40] K. Xu, L. Rui, Y. Li, and L. Gu. Feature normalized knowledge distillation for image classification. In *Proceedings of ECCV*, pages 664–680, 2020. 2

[41] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *Proceedings of ICCV*, pages 6022–6031, 2019. 6

[42] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of ICLR*, 2018. 6

[43] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of AAAI*, pages 13001–13008, 2020. 6