A. Full results of training on ImageNet and ImageNet$^+$, compared with Knowledge Distillation

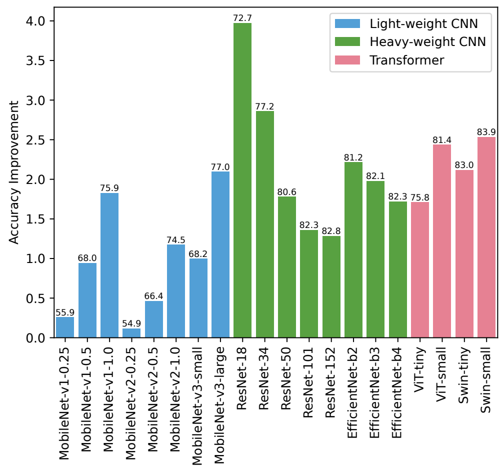
Table 11 provides the full results of training with ImageNet$^+$ compared with ImageNet and Knowledge Distillation (KD). We choose *RRC+RA/RE* that provides a balanced trade-off across architectures and training durations, and call it ImageNet$^+$. Results in Tab. 11 are without some state-of-the-art training features that are further improved in Tab. 12.

Table 12 provides improved results using state-of-the-art training recipes from the CVNets library [42]. We use the exact same ImageNet$^+$ variant and only write a new dataset class in CVNets, further confirming our minimal code change claim. We note the training changes that help each model:

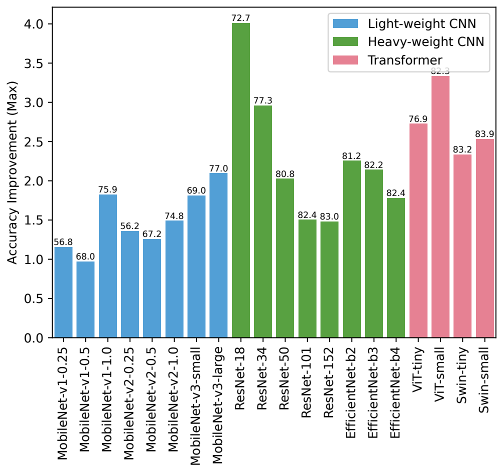
- Higher resolution training: EfficientNets, ViT-Base, Swin-Base. We observe that ImageNet$^+$ reinforcements are resolution independent and provide improvements even if the resolution is different from the one used to generate them.
- Variable resolution with variable batch size training (VBS): ViTs, EfficientNets, Swin.
- Mixed-precision: ViTs, Swin.
- Multi-node training: EfficientNets (resolution larger than 224).
- Exponential Model Averaging (EMA): MobileViTs.
- New results for MobileViT.

A.1. Aggregated improvements of ImageNet$^+$ across models

To better demonstrate the scale of accuracy improvements, we plot the results of training on ImageNet$^+$ (*RRC+RA/RE*) from Tab. 11 in Fig. 6a (300 epochs). *RRC+RA/RE* balances the tradeoff between various architectures. Given prior knowledge of architecture characteristics or enough training resources, we can select the dataset optimal for any architecture. Figure 6b shows the best accuracy achieved for each model when we train on all 4 of our reinforced datasets for 300 epochs (maximum of the four numbers). We observe that alternative reinforced datasets can provide 1-2% additional improvement, especially for light-weight CNNs and Transformers. In practice, given the knowledge of the complexity of the model architecture, one can decide to use alternative datasets (*RRC* for light-weight and *RRC+M* +R** for heavy-weight models or Transformers). Otherwise, given additional compute resources, one can train on all 4 datasets and choose the best model according to the validation accuracy.



(a) ImageNet$^+$ (RRC+RA/RE)



(b) Maximum of 4 ImageNet$^+$ variants

Figure 6: **ImageNet$^+$ training improves validation accuracy** compared with ImageNet training (Epochs=300). To train models using the ImageNet$^+$ dataset, we use the same publicly-available ImageNet training recipes with *no hyperparameter tuning* on ImageNet$^+$. We use the same hyperparameters tuned for ImageNet with *no hyperparameter tuning* on ImageNet$^+$. ImageNet$^+$ provides a balanced tradeoff with more improvements for Heavy-weight CNNs and Transformers. Figure 6b: further improvements are achieved using the best out of our 4 proposed datasets (See Tab. 11b for details).

Model	ImageNet	KD	RRC	RRC+Mixing	RRC+RA/RE	RRC+M*+R*
MobileNetv1-0.25	54.5	56.5$_{+2.0}$	**56.5**$_{+2.0}$	55.2$_{+0.7}$	55.7$_{+1.2}$	53.9$_{-0.6}$
MobileNetv1-0.5	66.3	66.3$_{-0.0}$	**67.6**$_{+1.3}$	67.1$_{+0.8}$	66.9$_{+0.6}$	66.3$_{-0.0}$
MobileNetv1-1.0	73.6	74.6$_{+1.0}$	**75.2**$_{+1.6}$	**75.0**$_{+1.4}$	**75.0**$_{+1.4}$	74.3$_{+0.7}$
MobileNetv2-0.25	54.3	56.9$_{+2.6}$	**55.8**$_{+1.5}$	54.1$_{-0.2}$	53.8$_{-0.5}$	52.5$_{-1.8}$
MobileNetv2-0.5	65.3	66.2$_{+0.9}$	**66.0**$_{+0.7}$	65.7$_{+0.4}$	**65.8**$_{+0.4}$	64.0$_{-1.3}$
MobileNetv2-1.0	72.7	72.8$_{+0.1}$	**73.8**$_{+1.1}$	73.5$_{+0.8}$	73.5$_{+0.8}$	72.9$_{+0.2}$
MobileNetv3-Small	66.3	65.8$_{-0.5}$	**68.0**$_{+1.7}$	67.1$_{+0.8}$	67.3$_{+1.0}$	66.4$_{+0.2}$
MobileNetv3-Large	74.7	75.5$_{+0.9}$	**76.0**$_{+1.4}$	**76.0**$_{+1.4}$	**76.2**$_{+1.6}$	75.5$_{+0.8}$
ResNet-18	67.8	72.1$_{+4.3}$	**72.3**$_{+4.5}$	71.7$_{+4.0}$	71.9$_{+4.1}$	71.0$_{+3.2}$
ResNet-34	73.2	76.4$_{+3.2}$	**76.2**$_{+3.0}$	**76.3**$_{+3.1}$	**76.2**$_{+3.0}$	75.8$_{+2.6}$
ResNet-50	77.4	80.3$_{+2.9}$	**79.6**$_{+2.3}$	**79.7**$_{+2.3}$	**79.6**$_{+2.3}$	**79.6**$_{+2.2}$
ResNet-101	79.8	81.7$_{+1.9}$	81.2$_{+1.4}$	**81.7**$_{+1.8}$	**81.5**$_{+1.7}$	81.3$_{+1.5}$
ResNet-152	80.8	82.3$_{+1.4}$	81.6$_{+0.8}$	**82.0**$_{+1.2}$	**82.0**$_{+1.1}$	**81.9**$_{+1.0}$
EfficientNet-B2	77.9	80.0$_{+2.1}$	80.2$_{+2.3}$	**80.6**$_{+2.7}$	**80.7**$_{+2.7}$	80.4$_{+2.4}$
EfficientNet-B3	79.3	80.9$_{+1.6}$	81.1$_{+1.8}$	**81.5**$_{+2.2}$	**81.6**$_{+2.3}$	**81.5**$_{+2.2}$
EfficientNet-B4	79.4	81.8$_{+2.4}$	81.0$_{+1.6}$	**81.3**$_{+1.9}$	**81.5**$_{+2.1}$	**81.3**$_{+1.9}$
ViT-Tiny	71.5	72.0$_{+0.5}$	71.5$_{+0.0}$	74.1$_{+2.6}$	74.0$_{+2.6}$	**74.6**$_{+3.1}$
ViT-Small	78.4	80.2$_{+1.7}$	77.0$_{-1.5}$	79.9$_{+1.4}$	79.7$_{+1.2}$	**80.8**$_{+2.4}$
Swin-Tiny	79.9	81.7$_{+1.7}$	81.3$_{+1.4}$	**82.1**$_{+2.2}$	**82.0**$_{+2.1}$	**82.2**$_{+2.2}$
Swin-Small	80.6	83.4$_{+2.9}$	81.9$_{+1.3}$	**82.9**$_{+2.3}$	**82.9**$_{+2.4}$	**83.1**$_{+2.6}$

(a) 150 epochs

Model	ImageNet	KD	RRC	RRC+Mixing	RRC+RA/RE	RRC+M*+R*
MobileNetv1-0.25	55.7	57.4$_{+1.8}$	**56.8**$_{+1.2}$	56.2$_{+0.5}$	55.9$_{+0.3}$	55.1$_{-0.6}$
MobileNetv1-0.5	67.1	67.5$_{+0.4}$	**68.0**$_{+1.0}$	67.7$_{+0.7}$	**68.0**$_{+0.9}$	66.9$_{-0.1}$
MobileNetv1-1.0	74.0	75.9$_{+1.9}$	75.6$_{+1.6}$	**75.8**$_{+1.8}$	**75.9**$_{+1.8}$	75.4$_{+1.3}$
MobileNetv2-0.25	54.8	57.7$_{+2.9}$	**56.2**$_{+1.4}$	55.2$_{+0.4}$	54.9$_{+0.1}$	53.1$_{-1.7}$
MobileNetv2-0.5	66.0	66.9$_{+0.9}$	**67.2**$_{+1.3}$	66.5$_{+0.5}$	66.4$_{+0.5}$	65.3$_{-0.7}$
MobileNetv2-1.0	73.3	74.3$_{+1.0}$	**74.8**$_{+1.5}$	74.2$_{+0.9}$	74.5$_{+1.2}$	73.9$_{+0.6}$
MobileNetv3-Small	67.2	67.0$_{-0.2}$	**69.0**$_{+1.8}$	68.1$_{+0.8}$	68.2$_{+1.0}$	67.1$_{-0.1}$
MobileNetv3-Large	74.9	76.4$_{+1.5}$	76.6$_{+1.7}$	**76.9**$_{+2.0}$	**77.0**$_{+2.1}$	76.5$_{+1.6}$
ResNet-18	68.7	73.5$_{+4.8}$	**72.7**$_{+4.0}$	**72.7**$_{+4.0}$	**72.7**$_{+4.0}$	72.1$_{+3.4}$
ResNet-34	74.3	77.9$_{+3.6}$	76.9$_{+2.6}$	**77.3**$_{+3.0}$	**77.2**$_{+2.9}$	76.9$_{+2.6}$
ResNet-50	78.8	81.5$_{+2.8}$	80.3$_{+1.5}$	**80.8**$_{+2.0}$	**80.6**$_{+1.8}$	80.5$_{+1.7}$
ResNet-101	80.9	83.0$_{+2.1}$	81.8$_{+0.9}$	**82.3**$_{+1.4}$	**82.3**$_{+1.4}$	**82.4**$_{+1.5}$
ResNet-152	81.5	83.4$_{+1.9}$	82.5$_{+1.0}$	**83.0**$_{+1.5}$	**82.8**$_{+1.3}$	**82.9**$_{+1.4}$
EfficientNet-B2	78.9	81.3$_{+2.3}$	80.9$_{+1.9}$	**81.2**$_{+2.3}$	**81.2**$_{+2.2}$	**81.1**$_{+2.2}$
EfficientNet-B3	80.1	82.1$_{+2.0}$	81.7$_{+1.6}$	**82.2**$_{+2.1}$	**82.1**$_{+2.0}$	**82.1**$_{+2.0}$
EfficientNet-B4	80.6	83.0$_{+2.3}$	82.1$_{+1.5}$	**82.4**$_{+1.8}$	**82.3**$_{+1.7}$	**82.4**$_{+1.7}$
ViT-Tiny	74.1	75.5$_{+1.4}$	73.5$_{-0.6}$	76.2$_{+2.0}$	75.8$_{+1.7}$	**76.9**$_{+2.7}$
ViT-Small	78.9	82.3$_{+3.4}$	79.2$_{+0.2}$	81.6$_{+2.7}$	81.4$_{+2.4}$	**82.3**$_{+3.3}$
Swin-Tiny	80.9	83.0$_{+2.1}$	82.4$_{+1.5}$	**83.1**$_{+2.2}$	**83.0**$_{+2.1}$	**83.2**$_{+2.3}$
Swin-Small	81.4	84.4$_{+3.0}$	82.5$_{+1.1}$	**83.7**$_{+2.3}$	**83.9**$_{+2.5}$	**83.9**$_{+2.5}$

(b) 300 epochs

Model	ImageNet	RRC	RRC+Mixing	RRC+RA/RE	RRC+M*+R*
MobileNetv1-0.25	56.7	**57.6**$_{+0.9}$	57.1$_{+0.4}$	57.1$_{+0.4}$	56.3$_{-0.4}$
MobileNetv1-0.5	67.8	**69.2**$_{+1.4}$	68.8$_{+1.0}$	68.5$_{+0.7}$	68.0$_{+0.2}$
MobileNetv1-1.0	74.1	76.4$_{+2.2}$	**76.7**$_{+2.6}$	**76.7**$_{+2.5}$	76.4$_{+2.3}$
MobileNetv2-0.25	55.7	**56.7**$_{+1.0}$	55.8$_{+0.1}$	55.2$_{-0.5}$	55.0$_{-0.7}$
MobileNetv2-0.5	66.8	**68.2**$_{+1.4}$	67.3$_{+0.5}$	67.1$_{+0.3}$	65.9$_{-0.9}$
MobileNetv2-1.0	73.9	**75.4**$_{+1.5}$	**75.3**$_{+1.4}$	**75.5**$_{+1.6}$	74.7$_{+0.8}$
MobileNetv3-Small	67.9	69.0$_{+1.1}$	68.4$_{+0.5}$	**69.4**$_{+1.4}$	68.4$_{+0.4}$
MobileNetv3-Large	75.1	77.2$_{+2.1}$	77.4$_{+2.3}$	**77.9**$_{+2.9}$	77.5$_{+2.4}$
ResNet-18	69.9	73.6$_{+3.6}$	**73.9**$_{+4.0}$	**73.8**$_{+3.8}$	73.3$_{+3.4}$
ResNet-34	75.6	77.8$_{+2.2}$	**78.4**$_{+2.8}$	**78.4**$_{+2.8}$	78.1$_{+2.6}$
ResNet-50	79.6	81.1$_{+1.4}$	**81.8**$_{+2.2}$	**81.7**$_{+2.1}$	**81.8**$_{+2.2}$
ResNet-101	81.4	82.7$_{+1.3}$	**83.6**$_{+2.2}$	83.2$_{+1.8}$	**83.4**$_{+2.0}$
ResNet-152	81.7	83.4$_{+1.7}$	**84.0**$_{+2.3}$	**83.8**$_{+2.2}$	**83.9**$_{+2.3}$
EfficientNet-B2	79.3	81.5$_{+2.2}$	**81.9**$_{+2.7}$	**81.9**$_{+2.7}$	**81.7**$_{+2.5}$
EfficientNet-B3	79.6	82.3$_{+2.7}$	**82.9**$_{+3.3}$	**82.8**$_{+3.3}$	**82.7**$_{+3.2}$
EfficientNet-B4	81.2	82.9$_{+1.8}$	**83.2**$_{+2.1}$	**83.1**$_{+1.9}$	**83.2**$_{+2.0}$
ViT-Tiny	75.9	76.6$_{+0.7}$	**78.5**$_{+2.6}$	78.1$_{+2.1}$	**78.7**$_{+2.8}$
ViT-Small	78.4	80.8$_{+2.4}$	83.2$_{+4.8}$	82.6$_{+4.2}$	**83.7**$_{+5.3}$
Swin-Tiny	80.9	83.4$_{+2.5}$	**84.1**$_{+3.1}$	83.8$_{+2.8}$	**84.0**$_{+3.1}$
Swin-Small	81.9	83.9$_{+1.9}$	84.5$_{+2.6}$	84.4$_{+2.5}$	**84.8**$_{+2.9}$

(c) 1000 epochs

Table 11: Comparison of training different models using knowledge distillation and different ImageNet/ImageNet$^+$ datasets. Subscripts show the improvement on top of the ImageNet accuracy. We highlight the best accuracy on each row from our proposed datasets and any number that is within 0.2 of the best. Knowledge distillation results are not reported for E=1000 (Tab. 11c) as it is computationally very expensive.

Model	Base Recipes (Tab. 11)		CVNets		CVNets-EMA	
	ImageNet	ImageNet$^+$	ImageNet	ImageNet$^+$	ImageNet	ImageNet$^+$
MobileNetV1-0.25	54.5	**55.7**$_{+1.2}$	55.2	55.4$_{+0.2}$	55.4	55.4$_{+0.0}$
MobileNetV1-0.5	66.3	**66.9**$_{+0.6}$	66.2	**67.1**$_{+0.9}$	66.4	**67.1**$_{+0.7}$
MobileNetV1-1.0	73.6	**75.0**$_{+1.4}$	73.5	**75.1**$_{+1.5}$	73.6	**75.1**$_{+1.5}$
MobileNetV2-0.25	54.3	53.8$_{-0.5}$	**54.7**	54.2$_{-0.5}$	**54.7**	54.2$_{-0.5}$
MobileNetV2-0.5	65.3	**65.8**$_{+0.4}$	**65.7**	**65.7**$_{-0.0}$	**65.7**	**65.7**$_{+0.0}$
MobileNetV2-1.0	72.7	73.5$_{+0.8}$	72.8	**73.8**$_{+1.0}$	72.8	**73.9**$_{+1.1}$
MobileNetV3-Small	66.3	67.3$_{+1.0}$	66.6	**67.7**$_{+1.1}$	66.7	**67.7**$_{+1.1}$
MobileNetV3-Large	74.7	76.2$_{+1.6}$	74.7	**76.5**$_{+1.8}$	74.8	**76.5**$_{+1.7}$
MobileViT-XXSmall	-	-	66.0	67.4$_{+1.5}$	66.7	**67.9**$_{+1.2}$
MobileViT-XSmall	-	-	72.6	74.0$_{+1.4}$	73.3	**74.7**$_{+1.3}$
MobileViT-Small	-	-	76.3	78.3$_{+2.0}$	76.7	**78.6**$_{+1.9}$
ResNet-18	67.8	71.9$_{+4.1}$	69.9	**73.2**$_{+3.3}$	69.8	**73.2**$_{+3.4}$
ResNet-34	73.2	76.2$_{+3.0}$	74.6	**76.9**$_{+2.3}$	74.7	**76.9**$_{+2.3}$
ResNet-50	77.4	79.6$_{+2.3}$	79.0	**80.3**$_{+1.3}$	79.1	**80.3**$_{+1.2}$
ResNet-101	79.8	81.5$_{+1.7}$	80.5	**81.8**$_{+1.3}$	80.5	**81.9**$_{+1.3}$
ResNet-152	80.8	82.0$_{+1.1}$	81.3	**82.2**$_{+1.0}$	81.3	**82.3**$_{+0.9}$
EfficientNet-B2	77.9	80.7$_{+2.7}$	79.5	**81.5**$_{+2.1}$	79.5	**81.6**$_{+2.1}$
EfficientNet-B3	79.3	81.6$_{+2.3}$	80.9	**82.4**$_{+1.6}$	80.8	**82.5**$_{+1.6}$
EfficientNet-B4	79.4	81.5$_{+2.1}$	82.7	**83.6**$_{+1.0}$	82.7	**83.7**$_{+1.0}$
ViT-Tiny	71.5	74.0$_{+2.6}$	72.1	**74.3**$_{+2.2}$	72.1	**74.4**$_{+2.3}$
ViT-Small	78.4	**79.7**$_{+1.2}$	78.4	**79.8**$_{+1.4}$	78.7	**79.9**$_{+1.2}$
ViT-Base	-	-	79.5	**81.7**$_{+2.3}$	80.6	**81.7**$_{+1.1}$
ViT-384-Base	-	-	80.5	**83.0**$_{+2.5}$	81.9	**83.1**$_{+1.1}$
Swin-Tiny	79.9	**82.0**$_{+2.1}$	80.5	**82.1**$_{+1.6}$	80.3	**81.9**$_{+1.6}$
Swin-Small	80.6	82.9$_{+2.4}$	82.2	**83.6**$_{+1.4}$	81.9	83.3$_{+1.4}$
Swin-Base	-	-	82.7	**83.9**$_{+1.2}$	82.2	**83.7**$_{+1.4}$
Swin-384-Base	-	-	82.6	**83.2**$_{+0.6}$	82.4	**83.0**$_{+0.6}$

(a) 150 epochs

Model	Base Recipes (Tab. 11)		CVNets		CVNets-EMA	
	ImageNet	ImageNet$^+$	ImageNet	ImageNet$^+$	ImageNet	ImageNet$^+$
MobileNetV1-0.25	55.7	55.9$_{+0.3}$	56.0	**56.5**$_{+0.6}$	56.1	**56.6**$_{+0.5}$
MobileNetV1-0.5	67.1	**68.0**$_{+0.9}$	67.0	**68.0**$_{+1.0}$	67.0	**68.1**$_{+1.1}$
MobileNetV1-1.0	74.0	**75.9**$_{+1.8}$	74.0	**76.0**$_{+2.0}$	74.1	**76.0**$_{+1.9}$
MobileNetV2-0.25	54.8	54.9$_{+0.1}$	**55.4**	55.1$_{-0.3}$	**55.5**	55.1$_{-0.4}$
MobileNetV2-0.5	66.0	**66.4**$_{+0.5}$	65.8	**66.5**$_{+0.7}$	65.9	**66.6**$_{+0.7}$
MobileNetV2-1.0	73.3	**74.5**$_{+1.2}$	73.7	**74.5**$_{+0.8}$	73.7	**74.5**$_{+0.8}$
MobileNetV3-Small	67.2	68.2$_{+1.0}$	67.4	**68.6**$_{+1.2}$	67.4	**68.5**$_{+1.1}$
MobileNetV3-Large	74.9	**77.0**$_{+2.1}$	74.9	**77.2**$_{+2.3}$	75.1	**77.2**$_{+2.1}$
MobileViT-XXSmall	-	-	67.5	68.8$_{+1.3}$	68.6	**69.7**$_{+1.1}$
MobileViT-XSmall	-	-	74.0	75.6$_{+1.6}$	74.9	**76.3**$_{+1.4}$
MobileViT-Small	-	-	77.3	79.6$_{+2.3}$	77.9	**80.1**$_{+2.2}$
ResNet-18	68.7	72.7$_{+4.0}$	71.2	**74.2**$_{+3.0}$	71.1	**74.2**$_{+3.1}$
ResNet-34	74.3	77.2$_{+2.9}$	75.6	**77.8**$_{+2.1}$	75.6	**77.8**$_{+2.2}$
ResNet-50	78.8	80.6$_{+1.8}$	79.6	**81.2**$_{+1.6}$	79.7	**81.2**$_{+1.6}$
ResNet-101	80.9	82.3$_{+1.4}$	81.3	**82.6**$_{+1.3}$	81.3	**82.7**$_{+1.4}$
ResNet-152	81.5	82.8$_{+1.3}$	81.8	**83.1**$_{+1.3}$	81.8	**83.1**$_{+1.3}$
EfficientNet-B2	78.9	81.2$_{+2.2}$	80.7	**82.1**$_{+1.4}$	80.8	**82.1**$_{+1.3}$
EfficientNet-B3	80.1	82.1$_{+2.0}$	81.8	**83.3**$_{+1.5}$	81.8	**83.3**$_{+1.5}$
EfficientNet-B4	80.6	82.3$_{+1.7}$	82.8	**84.4**$_{+1.6}$	82.7	**84.4**$_{+1.7}$
ViT-Tiny	74.1	**75.8**$_{+1.7}$	74.8	**76.0**$_{+1.2}$	74.9	**76.0**$_{+1.1}$
ViT-Small	78.9	**81.4**$_{+2.4}$	79.1	**81.4**$_{+2.3}$	79.9	**81.5**$_{+1.5}$
ViT-Base	-	-	78.6	**84.1**$_{+5.5}$	81.0	**84.1**$_{+3.1}$
ViT-384-Base	-	-	80.0	**84.5**$_{+4.6}$	82.6	**84.5**$_{+1.9}$
Swin-Tiny	80.9	**83.0**$_{+2.1}$	81.2	**83.2**$_{+2.1}$	80.8	82.8$_{+2.0}$
Swin-Small	81.4	83.9$_{+2.5}$	82.4	**84.4**$_{+2.0}$	82.2	84.1$_{+1.9}$
Swin-Base	-	-	82.7	**84.7**$_{+2.0}$	82.5	84.4$_{+1.9}$
Swin-384-Base	-	-	83.9	**84.4**$_{+0.5}$	83.7	**84.2**$_{+0.5}$

(b) 300 epochs

Model	Base Recipes (Tab. 11)		CVNets		CVNets-EMA	
	ImageNet	ImageNet$^+$	ImageNet	ImageNet$^+$	ImageNet	ImageNet$^+$
MobileNetV1-0.25	56.7	**57.1**$_{+0.4}$	**56.9**	**57.1**$_{+0.2}$	**56.9**	**57.1**$_{+0.2}$
MobileNetV1-0.5	67.8	68.5$_{+0.7}$	68.1	**68.7**$_{+0.6}$	68.1	**68.8**$_{+0.7}$
MobileNetV1-1.0	74.1	**76.7**$_{+2.5}$	74.1	**76.8**$_{+2.7}$	74.4	**76.8**$_{+2.4}$
MobileNetV2-0.25	**55.7**	55.2$_{-0.5}$	**55.7**	**55.7**$_{-0.0}$	**55.8**	**55.7**$_{-0.1}$
MobileNetV2-0.5	66.8	**67.1**$_{+0.3}$	66.8	**67.1**$_{+0.3}$	66.8	**67.2**$_{+0.4}$
MobileNetV2-1.0	73.9	**75.5**$_{+1.6}$	74.0	**75.5**$_{+1.5}$	74.1	**75.6**$_{+1.5}$
MobileNetV3-Small	67.9	**69.4**$_{+1.4}$	68.1	**69.6**$_{+1.5}$	68.1	**69.6**$_{+1.5}$
MobileNetV3-Large	75.1	**77.9**$_{+2.9}$	74.8	**77.9**$_{+3.1}$	75.8	**77.9**$_{+2.1}$
MobileViT-XXSmall	-	-	68.6	69.5$_{+0.9}$	70.3	**71.5**$_{+1.2}$
MobileViT-XSmall	-	-	74.8	76.2$_{+1.4}$	76.1	**77.5**$_{+1.4}$
MobileViT-Small	-	-	77.7	80.5$_{+2.8}$	79.2	**81.4**$_{+2.2}$
ResNet-18	69.9	73.8$_{+3.8}$	72.3	**75.0**$_{+2.7}$	72.2	**75.1**$_{+2.9}$
ResNet-34	75.6	78.4$_{+2.8}$	76.6	**78.8**$_{+2.2}$	76.6	**78.9**$_{+2.3}$
ResNet-50	79.6	81.7$_{+2.1}$	80.0	**82.0**$_{+1.9}$	80.1	**82.0**$_{+1.9}$
ResNet-101	81.4	83.2$_{+1.8}$	80.9	**83.5**$_{+2.6}$	81.4	**83.5**$_{+2.1}$
ResNet-152	81.7	**83.8**$_{+2.2}$	81.3	**83.9**$_{+2.6}$	82.0	**83.9**$_{+2.0}$
EfficientNet-B2	79.3	81.9$_{+2.7}$	81.1	**83.1**$_{+2.0}$	81.3	**83.2**$_{+1.9}$
EfficientNet-B3	79.6	82.8$_{+3.3}$	81.7	**83.9**$_{+2.2}$	82.1	**83.9**$_{+1.8}$
EfficientNet-B4	81.2	83.1$_{+1.9}$	82.2	**85.0**$_{+2.9}$	83.4	**85.0**$_{+1.6}$
ViT-Tiny	75.9	**78.1**$_{+2.1}$	76.7	**77.9**$_{+1.2}$	76.9	**78.0**$_{+1.1}$
ViT-Small	78.4	82.6$_{+4.2}$	78.5	**82.8**$_{+4.2}$	80.6	**82.9**$_{+2.3}$
ViT-Base	-	-	76.8	**85.1**$_{+8.3}$	80.8	**85.1**$_{+4.3}$
ViT-384-Base	-	-	79.4	**85.4**$_{+6.0}$	83.1	**85.5**$_{+2.5}$
Swin-Tiny	80.9	**83.8**$_{+2.8}$	81.3	**84.0**$_{+2.7}$	80.5	83.5$_{+3.0}$
Swin-Small	81.9	84.4$_{+2.5}$	81.3	**85.0**$_{+3.7}$	81.9	84.5$_{+2.6}$
Swin-Base	-	-	81.5	**85.4**$_{+3.9}$	81.8	**85.2**$_{+3.5}$
Swin-384-Base	-	-	83.6	**85.8**$_{+2.2}$	83.8	85.5$_{+1.7}$

(c) 1000 epochs

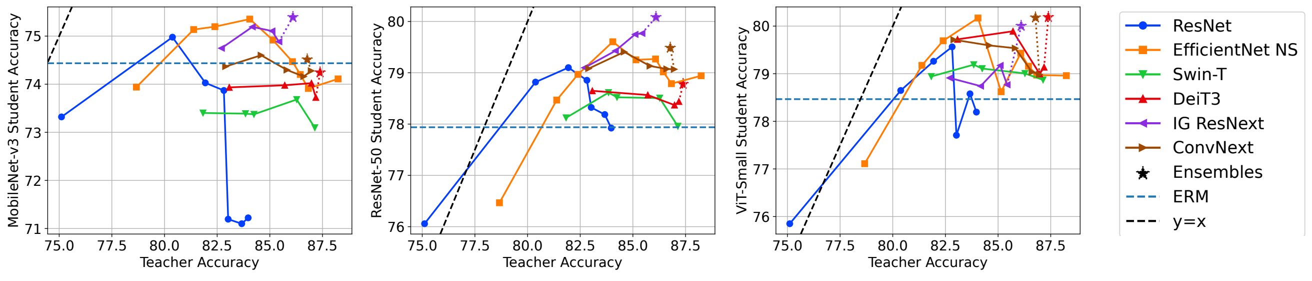
Table 12: **Improved results with state-of-the-art training recepies in CVNets library.** Subscripts show the improvement on top of the ImageNet accuracy. We highlight the best accuracy on each row from our proposed datasets and any number that is within 0.2 of the best.

B. Expanded study on what is a good teacher?

In this section we provide additional results and studies such as super ensembles as teachers.

B.1. Additional results of knowledge distillation with pretrained Timm models

Figure 7 (E=150) complements Fig. 3 (E=300) demonstrating the validation accuracy using knowledge distillation for a variety of teachers from the Timm library [63]. Table 13 shows the results in detail. For both 150 and 300 epoch training durations, we observe that ensembles of the state-of-the-art models in the Timm library perform best as the teachers across different student architectures. We choose the IG-ResNext ensemble for dataset reinforcement.



(a) MobileNetV3 (b) ResNet50 (c) ViT-Small

Figure 7: Knowledge distillation accuracy of representative student architectures (ResNet-50, ViT-Small, MobileNetV3) for pretrained teachers from Timm library. We train for 150 epochs and Fig. 3 shows results for 300 epoch training.

Teacher Name	Teacher Accuracy	Student Top-1 Accuracy					
		ResNet-50		**ViT-Small**		**MobileNetv3-Large**	
		300	150	300	150	300	150
beit_large_patch16_512	88.58	41.76	41.45	37.03	35.87	—	—
convnext_tiny_in22ft1k	82.90	80.13	79.08	80.82	79.71	75.41	74.37
convnext_large + convnext_base + convnext_small + convnext_tiny + convnext_nano	84.34	80.34	78.94	80.73	79.82	—	—
convnext_small_in22ft1k	84.59	80.63	79.41	81.16	79.59	75.70	74.60
convnext_base_in22ft1k	85.81	80.56	79.13	81.36	79.54	75.44	74.30
convnext_large_in22ft1k	86.61	80.17	79.07	80.77	79.04	75.35	74.16
convnext_xlarge_in22ft1k + convnext_large_in22ft1k + convnext_base_in22ft1k + convnext_small_in22ft1k + convnext_tiny_in22ft1k	86.78	80.78	79.48	81.67	80.18	75.82	74.51
convnext_xlarge_in22ft1k	86.96	80.07	79.07	80.69	78.98	75.26	74.28
convnext_xlarge_384_in22ft1k	87.53	79.67	78.50	79.92	78.38	—	—
deit3_small_patch16_224_in21ft1k	83.07	79.60	78.65	81.23	79.72	75.04	73.93
deit3_huge_patch14_224 + deit3_large_patch16_224 + deit3_base_patch16_224 + deit3_small_patch16_224	85.30	79.69	78.82	80.25	79.55	—	—
deit3_base_patch16_224_in21ft1k	85.71	79.98	78.57	81.24	79.89	75.16	73.98
deit3_large_patch16_224_in21ft1k	86.98	79.61	78.37	80.81	79.00	75.08	74.02
deit3_huge_patch14_224_in21ft1k	87.18	79.52	78.44	80.59	79.14	74.79	73.73
deit3_huge_patch14_224_in21ft1k + deit3_large_patch16_224_in21ft1k + deit3_base_patch16_224_in21ft1k + deit3_small_patch16_224_in21ft1k	87.39	80.25	78.78	81.26	80.19	75.28	74.24
deit3_large_patch16_384_in21ft1k	87.73	79.06	78.00	79.71	78.80	—	—
ig_resnext101_32x8d	82.70	80.32	79.10	80.30	78.91	76.03	74.75
ig_resnext101_32x16d	84.17	80.78	79.43	80.93	78.74	76.37	75.19
ig_resnext101_32x32d	85.10	81.04	79.75	81.03	79.17	76.42	75.11
ig_resnext101_32x48d	85.43	80.93	79.77	80.67	78.77	76.01	74.88
ig_resnext101_32x48d + ig_resnext101_32x32d + ig_resnext101_32x16d + ig_resnext101_32x8d	86.10	81.30	80.08	82.01	80.00	76.70	75.39
ig_resnext101_32x48d + convnext_xlarge_in22ft1k + volo_d5_224 + deit3_huge_patch14_224	87.39	80.71	79.63	81.65	80.00	—	—
resnet18	69.74	71.29	71.29	71.29	71.18	—	—
resnet34	75.11	76.46	76.06	76.36	75.85	73.90	73.32
resnet50	80.38	79.83	78.82	79.81	78.65	75.63	74.98
resnet101	81.94	80.07	79.10	80.82	79.26	74.92	74.03
resnet152	82.82	79.88	78.85	79.72	79.56	74.82	73.87
resnet101d	83.02	79.75	78.32	78.92	77.71	72.78	71.19
resnet152d	83.67	79.62	78.19	78.77	78.58	73.05	71.10
resnet200d	83.97	79.40	77.92	80.18	78.19	73.10	71.22
resnetv2_152x2_bitm	84.46	79.57	78.63	80.04	78.52	75.12	73.87
resnetv2_152x4_bitm	84.94	—	78.09	—	78.97	—	73.69
swinv2_cr_tiny_ns_224	81.79	79.41	78.47	80.56	79.29	74.23	73.43
swinv2_tiny_window8_256	81.83	79.30	78.12	80.32	78.94	74.46	73.40
swinv2_tiny_window16_256	82.82	79.43	78.31	80.59	79.30	74.31	73.57
swinv2_cr_small_224	83.12	79.15	78.33	79.78	78.64	74.59	73.42
swinv2_cr_small_ns_224	83.48	79.43	78.62	80.38	78.94	74.56	73.62
swinv2_small_window8_256	83.84	79.66	78.61	80.35	79.19	74.44	73.39
swinv2_small_window16_256	84.22	79.21	78.41	80.00	78.30	74.84	73.45
swinv2_base_window8_256	84.25	79.57	78.52	80.32	79.11	74.73	73.37
swinv2_base_window16_256	84.59	78.94	78.30	79.50	78.32	74.50	73.63
swinv2_base_window12to16_192to256_22kft1k	86.27	79.76	78.51	80.80	79.01	74.35	73.68
swinv2_large_window12to16_192to256_22kft1k	86.94	79.31	78.38	79.86	78.43	74.39	73.51
swinv2_base_window12to24_192to384_22kft1k	87.14	79.18	77.96	80.01	78.87	—	73.10
swinv2_large_window12to24_192to384_22kft1k	87.47	78.48	77.65	78.33	78.38	—	73.07
tf_efficientnet_b0	76.85	—	75.10	—	75.97	73.68	72.93
tf_efficientnet_b0_ns	78.67	77.27	76.47	77.96	77.11	74.94	73.94
tf_efficientnet_b1	78.83	—	77.45	—	77.97	75.48	74.69
tf_efficientnet_b2	80.08	—	78.17	—	78.88	75.97	75.01
tf_efficientnet_b1_ns	81.38	79.52	78.47	80.38	79.18	76.14	75.14
tf_efficientnet_b3	81.65	—	78.88	—	79.56	76.39	75.38
tf_efficientnet_b2_ns	82.39	80.17	78.97	80.94	79.69	76.43	75.20
tf_efficientnet_b4	83.03	—	78.91	—	78.56	75.64	74.65
tf_efficientnet_b5	83.81	—	79.20	—	79.18	76.01	75.06
tf_efficientnet_b3_ns	84.05	80.71	79.60	81.72	80.17	76.44	75.35
tf_efficientnet_b6	84.11	—	78.92	—	79.21	75.58	74.41
tf_efficientnet_b7	84.93	—	79.16	—	79.24	75.42	74.43
tf_efficientnet_b4_ns	85.14	80.83	79.25	81.51	78.62	75.81	74.92
tf_efficientnet_b8	85.35	—	78.84	—	78.17	75.15	73.86
tf_efficientnet_b5_ns	86.08	80.71	79.27	81.05	79.43	75.57	74.47
tf_efficientnetv2_xl_in21ft1k	86.41	11.58	12.46	8.40	7.69	—	—
tf_efficientnet_b6_ns	86.44	80.21	79.02	80.91	79.16	75.36	74.20
tf_efficientnet_b7_ns	86.83	80.34	78.79	80.89	78.97	74.93	73.91
tf_efficientnet_l2_ns_475	88.24	—	78.94	—	78.96	—	74.11
vit_large_patch16_384	87.09	79.63	78.45	80.48	78.88	—	—
volo_d5_224 + volo_d4_224 + volo_d3_224 + volo_d2_224 + volo_d1_224	86.09	80.45	79.31	80.81	79.16	—	—
volo_d5_512	87.04	—	78.04	—	76.61	—	—

Table 13: Effect of distillation from pretrained teachers (Timm library) on the performance of MobileNetV3-large, ResNet-50, ViT-Small trained for 150 and 300 epochs. This table includes the details of Figs. 3 and 7.

B.2. Super ensembles on ImageNet

It is common to limit the number of models in an ensemble to less than 10 members and typically only 4. The reason is partly that larger ensembles are more expensive to evaluate at test time as well as training with knowledge distillation. Dataset reinforcement allows us to consider expensive teachers such as super ensembles with significantly more than 10 members. In Tab. 14 we present results for super ensembles on CIFAR-100 and in Fig. 8 we present results on ImageNet. On CIFAR-100 we create super ensembles by training 128 models in parallel for ResNet-18, ResNet-50, and ResNet-152 architectures. To increase diversity, we train models with 16 choices of enable/disable 4 augmentations (CutMix, MixUp, RandAugment, and Label Smoothing) and train with 8 different random seeds for each choice. In total we train $8 \times 16 = 128$ models. Tab. 15 shows the accuracy of the super ensembles while Tab. 14 shows the accuracy of distillation with the super ensembles. We observe that the best student accuracy is achieved with the largest ensemble 128xR152. Interestingly, super ensembles of small models (128xR50) are better than standard ensembles of large models (10xR152). With super ensembles we achieve strong accuracies for ResNet-50 at 86.30 and ResNet-152 at 87.03.

We also consider super ensembles for ImageNet using dataset reinforcement. Knowledge distillation with super ensembles of larger than 10 members on ImageNet becomes challenging and resource demanding. Fig. 8 shows the validation accuracy of the ensemble and Fig. 9 shows the accuracy of student training with the reinforced ImageNet dataset using the super ensemble. We observe that the entropy and confidence of the teacher on the validation set are not more correlated with the distillation accuracy than the accuracy of the validation teacher. In particular, large ensembles are more accurate but not necessarily better teachers. In summary, we observe that the optimal ensemble size for KD is around 4.

	10xR18	128xR18	10xR50	128xR50	10x152	128xR152
ResNet-18	83.57	84.24	83.43	84.07	83.65	**84.25**
ResNet-50	84.40	85.16	84.33	86.38	86.03	**86.30**
ResNet-152	85.01	85.74	85.00	86.80	86.85	**87.03**

Table 14: Distillation on CIFAR-100 with super diverse ensembles.

	Single best	Ensemble (128x)
ResNet-18	81.57	85.88
ResNet-50	83.43	87.29
ResNet-152	84.44	87.92

Table 15: Accuracy of super diverse ensembles on CIFAR-100.

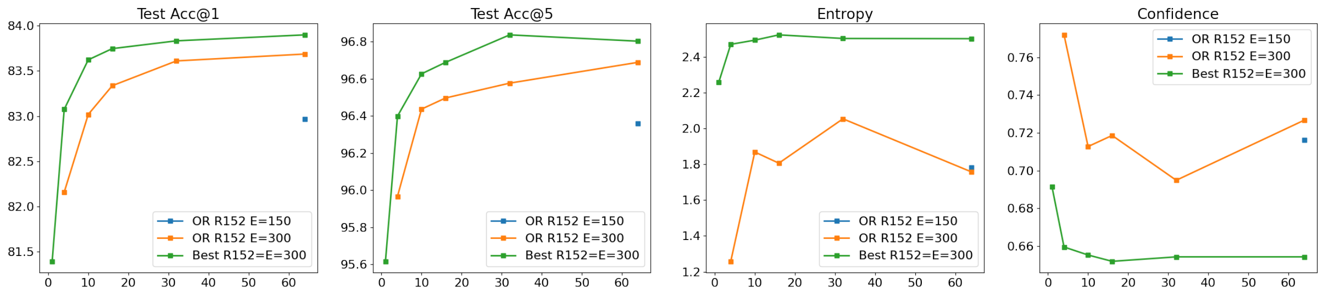


Figure 8: ImageNet accuracy of super ensembles as the size of the ensemble is increased. OR means the ensemble is created from diverse augmentation choices while Best means only the random seed is different between models.

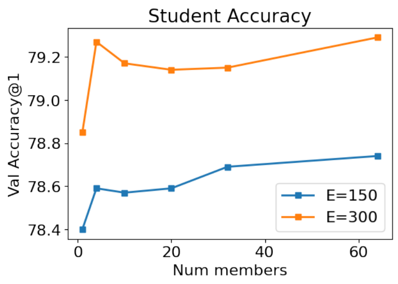
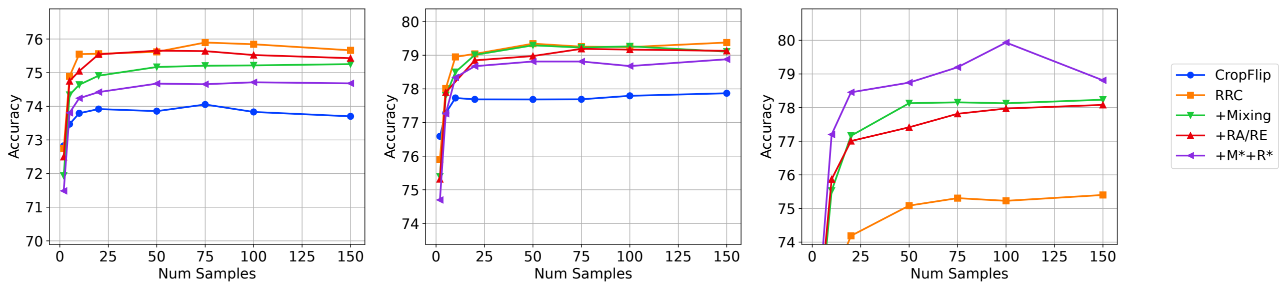


Figure 9: ImageNet super ensemble distillation accuracy for ResNet-50 facilitated by dataset reinforcement.



(a) Light-weight CNN (MobileNetV3) (b) Heavy-weight CNN (ResNet-50) (c) Transformer (ViT-Small)

Figure 10: **Light-weight CNNs prefer easy while Transformers prefer difficult reinforcements and we balance the tradeoff.** ImageNet validation accuracy of three representative architectures trained on reinforcements of ImageNet. We use ConvNext-Base-IN22FT1K as the teacher and train for 150 epochs. The x-axis is the number of augmentations stored per original sample in the ImageNet training set. In favor of dataset reinforcement, we observe that training with $25 - 50$ samples provides similar gains to training with more samples. The baseline augmentation is Fixed Resize-RandomCrop and horizontal flip (CropFlip). In addition we consider the following augmentations for reinforcement: Random-Resize-Crop and horizontal flip (*RRC*), MixUp and CutMix (*Mixing*), RandomAugment/RandomErase (*RA/RE*) and Mixing+RA/RE (*M**+*R**). We add these augmentations on top of *RRC* and for clarity add + as shorthand for *RRC*+.

C. Expanded study on reinforcing ImageNet

In this section, we provide ablations on the number and type of augmentations using a single relatively cheap teacher (ConvNext-Base-IN22FT1K) that still provides comparatively good improvements across all students.

C.1. What is the best combination of augmentations for reinforcement?

To recap, using our selected teachers from Sec. 2.1, we investigate the choice of augmentations for dataset reinforcement. Utilizing Fast Knowledge Distillation [55], we store the sparse outputs of a teacher on multiple augmentations. For efficiency, we store top 10 probabilities predicted by the teacher, along with the augmentation parameters and reapply augmented images in the data loader of the student. We observe that light-weight CNNs perform best on easier reinforcements while transformers perform best with difficult reinforcements. We balance this tradeoff using a mid-difficulty reinforcement.

We refer to the combination of baseline augmentations fixed resize, random crop and horizontal flip by CropFlip. In addition, we consider the following augmentations for dataset reinforcement: Random-Resize-Crop (*RRC*), MixUp [72] and CutMix [70] (*Mixing*), and RandomAugment [14] and RandomErase (*RA/RE*). We also combine *Mixing* with *RA/RE* and refer to it as $M^*{+}R^*$. We add all augmentations on top of *RRC* and for clarity add + as shorthand for *RRC*+. Except for mixing augmentations, reapplying all augmentations has zero overhead compared to standard training with the same augmentations. For mixing augmentations, our current implementation has approximately 30% wall-clock time overhead because of the extra load time of mixing pairs stored with each reinforced sample. We discuss efficient alternatives in Appendix C.3. Our balanced solution, *RRC*+*RA/RE*, does not use mixing and has zero overhead.

Figure 10 shows the accuracy of various models trained on reinforced datasets. We observe that the light-weight CNN performs best with *RRC* as the most simple augmentation after CropFlip while the transformer performs best with the most difficult set of reinforcements in *RRC*+M^* +R^*. This observation matches the standard state-of-the-art recipes for training these models. At the same time, we observe that *RRC*+*RA/RE* provides nearly the best performance for all models without the extra overhead of mixing methods in our implementation.

Consistent across three models and reinforcements, we observe that even though we train for 150 epochs, at most $25 - 50$ different augmentations of each training sample is enough to achieve the best accuracy for almost all methods. This gives at least $\times 3$ reduction in the number of samples we can take advantage of given a fixed training budget. Based on this observation and following [6], in Sec. 3 we train models for up to 1000 epochs while reinforce datasets with 400 augmentation samples.

C.2. Augmentation: invariance vs imitation

Data augmentation is crucial to train generalizable models in various domains. The key objective is to make the model invariant to content-preserving transformations. In knowledge distillation, however, it is not clear whether the student benefits more from being invariant to data augmentations as in Eq. (2) or from imitating teacher's variations on augmented data as in

Eq. (3). The training objective for each case is as follows:

$$\text{(Invariance)} \qquad \min_{\theta} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\hat{\boldsymbol{x}}\sim\mathcal{A}(\boldsymbol{x})} \mathcal{L}(f_\theta(\hat{\boldsymbol{x}}), g(\boldsymbol{x})) \tag{2}$$

$$\text{(Imitation)} \qquad \min_{\theta} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\hat{\boldsymbol{x}}\sim\mathcal{A}(\boldsymbol{x})} \mathcal{L}(f_\theta(\hat{\boldsymbol{x}}), g(\hat{\boldsymbol{x}})) \tag{3}$$

where, $\mathcal{D}$ is the training dataset, $\mathcal{A}$ is augmentation function, f_θ is the student model parameterized with θ, g is the teacher model, and $\mathcal{L}$ is the loss function between student and teacher outputs.

In Fig. 11, we compare the above training objectives for a wide range of augmentations in computer vision. For most augmentations, we observe imitation is more effective than invariance. This is consistent with observation in [6]. Therefore, in our setup we use augmentations only for imitation (and not invariance).

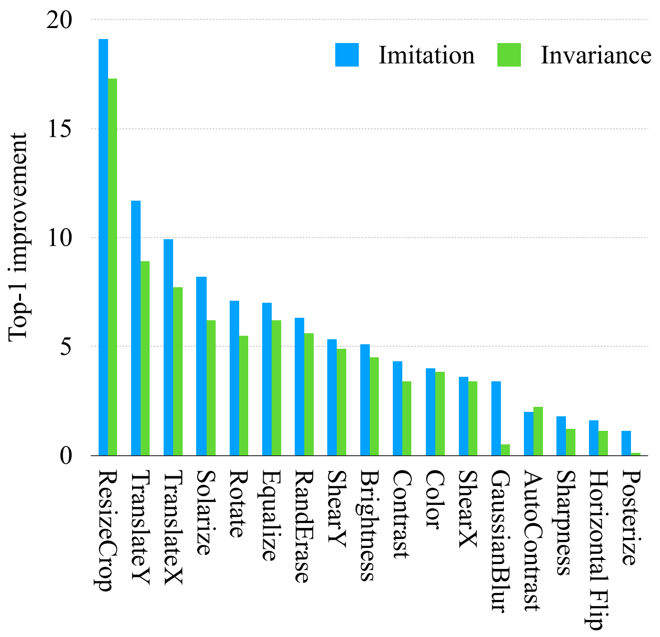


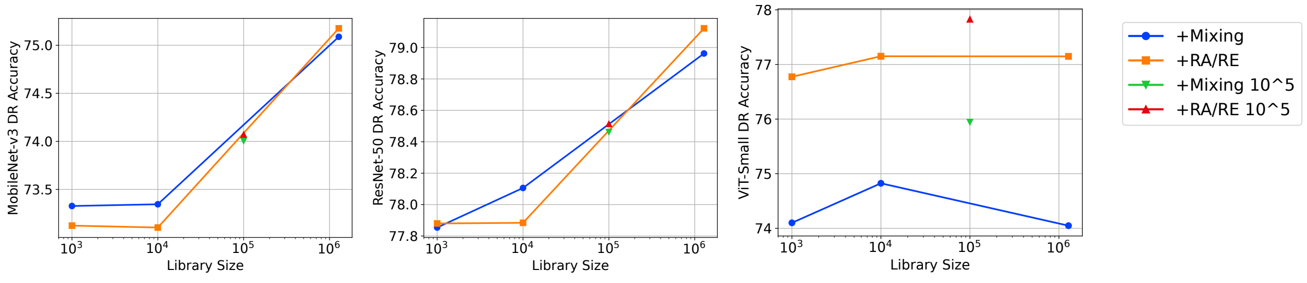
Figure 11: ImageNet top-1% accuracy improvement when distilling knowledge from a ConvNext (Base-IN22FT1K) teacher to a ViT-tiny student using a single augmentation with training objectives in Eq. (2) and Eq. (3). No augmentation top-1% accuracy is 54%.

C.3. Library size: Can we limit the mixing pairs?

Mixing augmentations have the extra overhead of the load time for the corresponding pair in each mini-batch. Standard training does not have such an overhead because the mixing is performed on random pairs within a mini-batch. In dataset reinforcement, the pairs that have been matched in the reinforcement phase are limited to the number of samples stored and do not always appear in the same mini-batch during the student training time. This means, we have to load the matching pair for every sample in the mini-batch that doubles the data load time and becomes an overhead for CPU-bound models. This overhead in the smallest models we consider is at most 30%. Even though much lower than the cost of knowledge distillation, it is still more than our desiderata would allow.

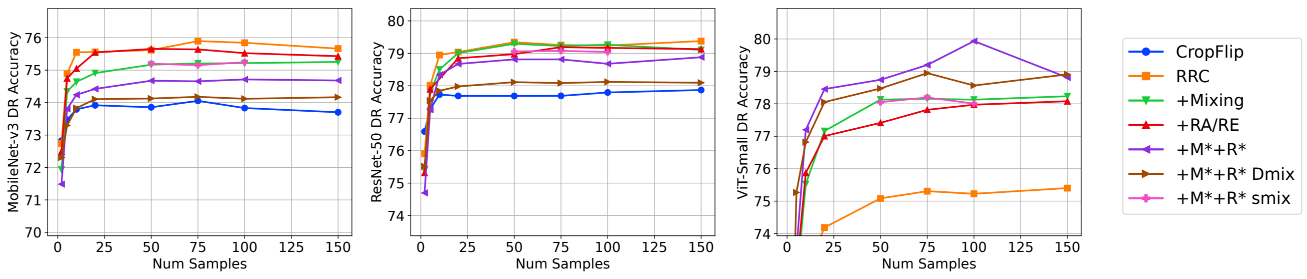
We consider an alternative where the pairing is done only with a library of selected samples from the training set. The library can be loaded in the memory once and reduce the additional cost incurred during the training. Fig. 12 shows the performance as we vary the library size. Even a relatively large library does not cover the accuracy drop caused by the reduced randomness in the mixing. The reason is that to reduce the cost, we can only have one augmentation per sample in the library which reduces the randomness from the mixing substantially and negatively affects knowledge distillation.

We also consider variations of mixing in Fig. 13. We consider two variations: Double-mix and Self-mix. In double-mix, for every augmented pair, we store two outputs with two sets of mixing coefficients. This means for every mini-batch we can load half the mini-batch along with a random pair for each sample, perform the stored augmentation on each and get two different mixed samples. As a result the overhead is zero. Second alternative, self-mix, mixes every image only by itself. As such, there



(a) MobileNetV3 (b) ResNet50 (c) ViT-Small

Figure 12: **Library of mixing pairs reduces wall-clock overhead for mixing methods but negatively impacts accuracy.** We plot the validation accuracy for models trained with ImageNet$^+$ as we vary the library size. The teacher is ConvNext-Base-IN22FT1K. See Appendix C.3 for details. (E=150)

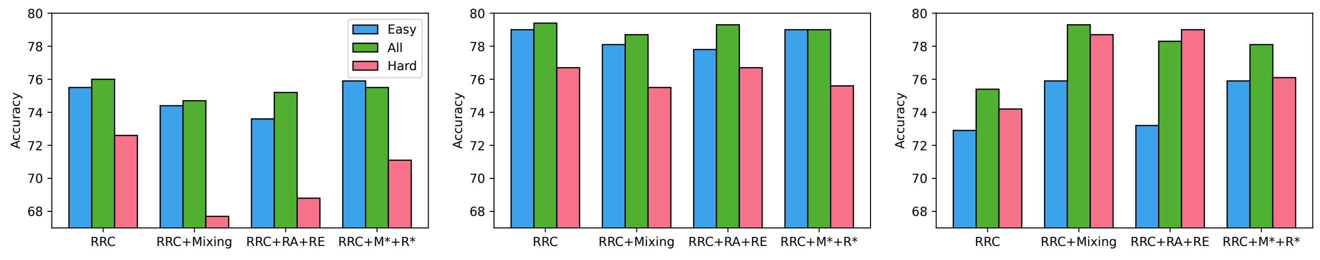


(a) Light-weight CNN (MobileNetV3) (b) Heavy-weight CNN (ResNet-50) (c) Transformer (ViT-Small)

Figure 13: **Alternative Mixing Augmentations.** Accuracy of three representative architectures trained on reinforcements of ImageNet. We use ConvNext-Base-IN22FT1K as the teacher and train for 150 epochs. The x-axis is the number of augmentations stored per original sample in the ImageNet training set. Augmentations used are Fixed Resize-RandomCrop and horizontal flip (CropFlip), Random-Resize-Crop and horizontal flip (*RRC*), MixUp and CutMix (Mixing), RandomAugment/RandomErase (RA/RE) and Mixing+RA/RE (M*+R*). Alterantive mixing augmentations: Double-mix (Dmix) and Self-mix (Smix).

is no data load time, but there is still an extra overhead of preprocessing the input twice. Fig. 13 shows that neither of the considered alternatives provide a better tradeoff compared with *RRC+RA/RE*. Therefore, we use *RRC+RA/RE* in our paper and call it ImageNet$^+$.

C.4. What is the best curriculum of reinforcements?



(a) Light-weight CNN (MobileNetV3) (b) Heavy-weight CNN (ResNet-50) (c) Transformer (ViT-Small)

Figure 14: **Tradeoff in augmentation difficulty can be further reduced with curriculums.** Using random samples of the optimal augmentation is the best ('All'). If we have access to one reinforcement dataset or we want to only keep a subset of data for efficiency, then it is better to use 'easy' curriculums for CNN based architectures and 'hard' curriculums for Transformers. Full table in Appendix C.5.

The one-time cost of reinforcing a dataset allows us to generate as much useful information as we need and store it for future use. An example is various metrics that can be used to devise learning curriculums that adapt to specific students. In this section we consider a set of initial curriculums we get for free with our dataset reinforcement strategy. Specifically, the output of the teacher on each sample also incorporates the confidence of the teacher on its prediction. We can use the confidence or the entropy of its predictions to make curriculums.

Given $\boldsymbol{p} \in \mathbb{R}^c$ the set of predicted probabilities of the teacher for c classes, we define confidence as $\max \boldsymbol{p}_j$. For every sample, we order its augmentations by the confidence value from 0 to #samples. During training, at each iteration we only sample from a range of augmentations with indices between $[a, b]$, where $0 \leq a, b <$ #samples. We devise curriculums by smoothly changing a, b during the training using a cosine function between specified values of initial and final values for a, b.

Fig. 14 shows the performance of various Easy, Hard, and All curriculums. Easy curriculums start from $[0, 10]$ (the 10% easiest samples), hard samples start from $[90, 100]$ (the 10% hardest samples), and All curriculums start from $[0, 100]$ (all the samples). We observe that the curriculum provides an alternative knob to control the difficulty of reinforcements that we can use adaptively during the training of the student. For example, the best performance of the light-weight CNN is with *RRC* combined with the All curriculum, but similar performance can be achieved with *RRC+RA/RE* combined with an Easy curriculum. Similarly, the transformer achieves its best performance with *RRC+M** *+R** combined with the All curriculum, while a similar performance can be achieved with *RRC+Mixing* and a Hard curriculum.

In Appendix C.6, we study various objectives for choosing most useful samples during the reinforcement process. We consider storing on the most informative samples according to a number of metrics such as entropy, loss, and clustering. We make similar observations to the behaviour of curriculums that the objectives that increase hardness benefit the transformer while the easy objectives benefit the light-weight CNN.

C.5. Additional details of curriculums

We study reinforcements on curriculums shown in Fig. 15. Table 16 provides the full results for the effect of dataset reinforcement curriculums. We summarized these results in Fig. 14 where we compared '*→all' curriculums that end with 'all' of the data. We observe that the beginning of the curriculum has much more impact on the generalization than the end of the curriculum. We observe that 'all→*' curriculums perform the best while 'hard→*' curriculums perform near optimal for ViT-Small and 'easy→*' performs best for MobileNetV3-Large. At the same time, we observe clearly that the hard and easy curriculums result in significantly worse generalization when used to train the opposite architecture, i.e., 'easy→*' for ViT-Small and 'hard→*' for MobileNetV3-Large. This result clearly demonstrates the tradeoff in the architecture independent generalization controlled by the difficulty of reinforcements.

Curriculum	**MobileNetV3-Large**				**ResNet50**				**ViT-Small**			
	RRC	+RA/RE	+Mixing	+M*+R*	RRC	+RA/RE	+Mixing	+M*+R*	RRC	+RA/RE	+Mixing	+M*+R*
`easy->all`	75.5	**75.9**	73.6	74.4	79.0	79.0	77.8	78.1	72.9	75.9	73.2	75.9
`easy->easy`	75.2	75.7	74.3	74.5	79.0	79.2	77.7	78.0	72.6	75.9	73.1	75.9
`easy->hard`	75.6	**75.8**	74.1	74.6	79.0	79.2	77.8	78.0	72.6	76.2	72.9	76.0
`all->all`	**76.0**	75.5	75.2	74.7	**79.4**	79.0	**79.3**	78.7	75.4	78.1	78.3	**79.3**
`all->easy`	75.7	75.5	75.2	74.6	**79.5**	79.2	**79.3**	78.9	75.2	78.4	78.2	**79.2**
`all->hard`	**75.8**	75.5	75.3	74.6	**79.4**	**79.3**	79.2	78.9	75.3	77.8	78.5	**79.3**
`hard->all`	72.6	71.1	68.8	67.7	76.7	75.6	76.7	75.5	74.2	76.1	79.0	78.7
`hard->easy`	72.5	71.3	68.7	67.7	76.7	75.9	77.0	75.3	74.2	76.2	78.8	78.6
`hard->hard`	72.2	71.4	68.9	67.7	76.8	75.7	76.9	75.7	73.9	76.2	79.0	78.6

Table 16: **The effect of curriculum.** We observe that the beginning of the curriculum has much more impact on the generalization than the end of the curriculum (accuracy within the groups of three rows is similar). Accuracies within 0.2% of the best accuracy in each column are highlighted.

C.6. Optimal augmentation sample selection

We discussed that augmentations used to reinforce the dataset are sampled from a pool of augmentation operations and that we apply the augmentations with a predetermined application probability. The setup of dataset reinforcement allows us to optimize for the most informative augmentation samples. For example, we can generate a large set of candidate augmentations and choose a subset with maximum or minimum values of ad-hoc metrics. We considered selecting samples according to metrics such as confidence, entropy, and loss. Given $\boldsymbol{p} \in \mathbb{R}^c$, the set of predicted probabilities of the teacher for c classes, we define confidence as $\max \boldsymbol{p}_j$, the entropy as $-\sum_j \boldsymbol{p}_j \log \boldsymbol{p}_j$, and the loss as $-\log \boldsymbol{p}_y$ where y is the ground-truth label.

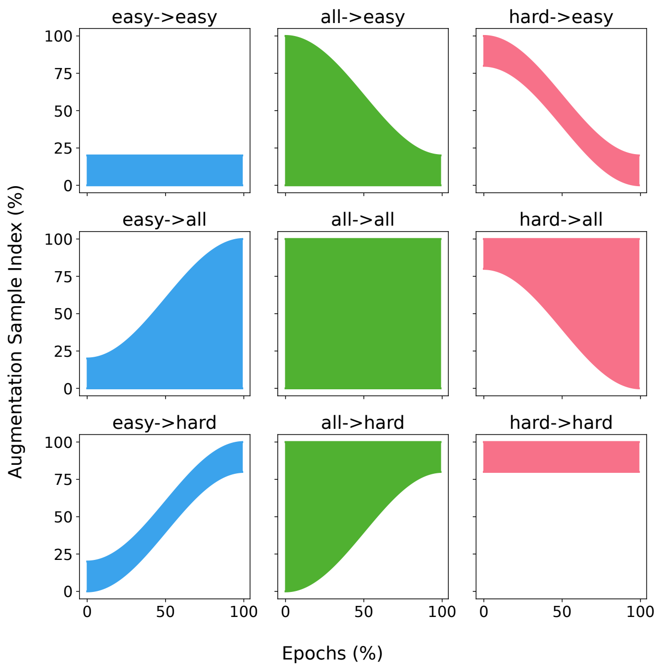
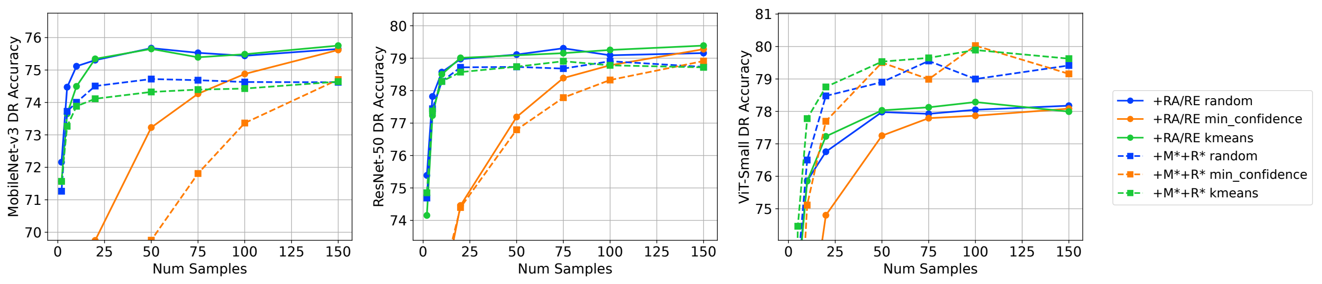


Figure 15: **Illustration of curriculums.** The x-axis shows the percentage of training epochs while the y-axis shows the index of augmentation samples in percentages as we order them from easy to hard by the confidence of the teacher. Highlighted regions show the subset of indices of reinforcements to uniformly draw from at each epoch.

To encourage diversity, we also considered selecting samples based on the clustering of the predicted probability vectors by performing KMeans on $\boldsymbol{p}$ vectors of the candidates and selecting one sample per cluster. Figure 16 shows the performance of a subset of sample selection methods we considered.

Generally we observe that max-entropy/min-confidence objectives demonstrate similar behaviors better than min-entropy/max-confidence. So we only show the min-confidence variant. We observe that overall random samples (blue lines) provides the best validation accuracy if used with the right augmentations (*RRC+RA/RE* for light-weight CNNs and *RRC+M** *+R** for transformers). Using min-confidence (orange) with *RRC+M** *+R** (dashed orange), leads to similar generalization on transformers while hurting the generalization on CNNs. This matches our observations with the complexity of augmentations and curriculums that transformers prefer difficult samples. We observe that diversified samples using KMeans clustering (green) provide similar behavior to random samples (blue) while for transformers provide more consistent improvements at varying number of samples (dashed green compared with dashed blue). We identify this potential for future work and investigate reinforced datasets with random samples in the rest of the paper. Note that the curriculums are a generalization of the objective-based metrics that are adaptive to the student (See Appendix C.4 and Appendix C.5).



(a) MobileNetV3 (b) ResNet50 (c) ViT-Small

Figure 16: **Optimal augmentation sample selection.** ImageNet^{+} accuracy for varying objectives and number of samples. The teacher is ConvNext (Base-IN22FT1K) (E=150)

D. Additional pretraining/finetuning/transfer learning results

In Tab. 17 we provide results for various combinations of pretraining and fine-tuning on reinforced/non-reinforced datasets. We observe that the best results are achieved when both pretraining and fine-tuning are done using reinforced datasets. We also observe that the improvement is significant compared to when only one of the pretraining/fine-tuning datasets is reinforced. The idea of training and fine-tuning on multiple reinforced datasets is unique to dataset reinforcement and would be challenging to replicate with standard data augmentations or knowledge distillation.

We train models for 100, 400, 1000 epochs on CIFAR-100, Food-101 and 1000, 4000, and 10000 epochs on Flowers-102 and report the best accuracy for each model. Models pretrained/fine-tuned on non-reinforced datasets tend to overfit at longer training while models trained on reinforced datasets benefit from longer training.

For future tasks and datasets, additional task-specific information could be considered as reinforcements. For example, an object detection dataset can be further reinforced using the teacher's uncertainty on bounding boxes, occlusion estimate, and border uncertainty. Multi-modal models such as CLIP are an immediate future work that can provide variety of additional training signal based on the relation to an anchor text.

Model	**Pretraining dataset**	**Fine-tuning dataset**					
		CIFAR-100	CIFAR-100$^+$	Flowers-102	Flowers-102$^+$	Food-101	Food-101$^+$
MobileNetV3-Large	None	80.2	83.6	68.8	87.5	85.1	88.2
	ImageNet	84.4	87.2	92.5	94.1	86.1	89.2
	ImageNet$^+$ (Ours)	86.0	**87.5**	93.7	**95.3**	86.6	**89.5**
ResNet-50	None	83.8	85.0	87.3	85.0	89.1	90.2
	ImageNet	88.4	89.5	93.6	94.9	90.0	91.8
	ImageNet$^+$ (Ours)	88.8	**89.8**	95.0	**96.3**	90.5	**92.1**
SwinTransformer-Tiny	None	35.0	82.2	78.3	72.5	89.6	90.9
	ImageNet	90.6	90.7	96.3	96.5	92.3	92.7
	ImageNet$^+$ (Ours)	90.9	**91.2**	96.6	**97.0**	**93.0**	**92.9**

Table 17: **Pretraining/Finetuning/Transfer learning for fine-grained object classification.**

E. Full table of calibration results

In Tab. 18 we provide the full results for Fig. 5. We see observe that validation ECE of ImageNet$^+$ pretrained models is lower than ImageNet pretrained models.

Model	**Method**	**Epochs**	**Train ECE**	**Val ECE**	**ECE gap**	**Train Error**	**Val Error**	**Error gap**
MobileNetV3-Large	ImageNet	300	0.1503	0.0727	0.0776	0.0934	0.2509	0.1575
	ImageNet$^+$	300	0.0339	0.0309	0.0030	0.1400	0.2298	0.0898
	ImageNet	1000	0.1489	0.0608	0.0881	0.0599	0.2491	0.1891
	ImageNet$^+$	1000	0.0312	0.0323	0.0011	0.1218	0.2206	0.0988
	KD	300	0.0303	0.0297	0.0006	0.1550	0.2358	0.0808
ResNet-50	ImageNet	300	0.1938	0.1513	0.0425	0.1239	0.2122	0.0883
	ImageNet$^+$	300	0.0263	0.0362	0.0098	0.1115	0.1944	0.0829
	ImageNet	1000	0.1887	0.1348	0.0539	0.0906	0.2036	0.1130
	ImageNet$^+$	1000	0.0241	0.0360	0.0119	0.0936	0.1830	0.0894
	KD	300	0.0250	0.0339	0.0089	0.1065	0.1846	0.0781
SwinTransformer-Tiny	ImageNet	300	0.1084	0.0663	0.0421	0.0734	0.1910	0.1176
	ImageNet$^+$	300	0.0201	0.0381	0.0180	0.0818	0.1698	0.0880
	ImageNet	1000	0.1042	0.0522	0.0519	0.0421	0.1905	0.1484
	ImageNet$^+$	1000	0.0195	0.0397	0.0203	0.0743	0.1621	0.0877
	KD	300	0.0206	0.0379	0.0173	0.0958	0.1701	0.0742

Table 18: Full calibration error and validation error for Fig. 5.

F. Cost of dataset reinforcement

In Appendix C.1, we observe that similar accuracy to knowledge distillation is reached with $\times 3$ fewer samples than the number of target epochs. This reduces the reinforcement cost. ImageNet$^+$ took 2080 mins to generate using 64xA100 GPUs which is highly parallelizable and similar to training ResNet-50 for 300 epochs on 8xA100 GPUs. The parallelization is another significant advantage to knowledge distillation because samples are reinforced independently while knowledge distillation requires following a trajectory on training samples. For CIFAR-100, Flowers-102, and Food-101, the reinforcement took 90, 40, and 120 minutes respectively. With pretrained teachers and extrapolating our ImageNet$^+$ observations, we can reinforce any new dataset and the cost is performing inference using the teacher on the dataset for approximately $\times 3$ fewer samples than the maximum intended training epochs. This is a one-time cost that is amortized over many uses.

We provide storage cost analysis for ImageNet$^+$ in Tab. 2. Note that for variants with mixing, the storage of *RRC+RA/RE* parameters doubles because each reinforcement consists of augmentations for a pair. The proposed ImageNet$^+$ variant, *RRC+RA/RE*, does not have that doubling cost. Also note that the storage can be further reduced using compression methods. For example, ImageNet$^+$ *RRC+RA/RE* with the compression from Python's Joblib with compression level 3 can be reduced to 55GBs instead of 61GBs. Even more compression is possible by reducing the number of stored logits for the teacher and more aggressive compression methods.

The storage cost for CIFAR-100$^+$, Flowers-102$^+$, and Food-101$^+$ uses the same set of formula given the number of samples that amounts to approximately 4.8, 1.0, and 7.3GBs in basic compressed form as in Tab. 2. We have not explored reducing the size of these datasets significantly as it is not a significant overhead for small datasets. For larger datasets such as ImageNet, the reinforcement overhead is much smaller relative to the original dataset size because the bulk of the dataset is taken by the inputs while our reinforcements only store the outputs.

We provide the breakdown of training time on MobileNetV3-Large, ResNet-50, and SwinTransformer-Tiny in Tab. 19. Except for mixing augmentations, reapplying all augmentations has zero overhead compared to standard training with the same augmentations. For mixing augmentations, our current implementation has approximately 30% time overhead because of the extra load time of mixing pairs stored with each reinforced sample. This overhead only translate to extra wall-clock for very small models where the bottleneck is on the CPU rather than GPU. We discuss efficient alternatives in Appendix C.3. Our balanced solution, *RRC+RA/RE*, does not use mixing and has zero overhead.

Model	**Dataset**	**Training Epochs**		
		150	**300**	**1000**
MobileNetV3-Large	ImageNet	1.00×	1.00×	1.00×
	ImageNet$^+$ (Ours)	1.13×	1.12×	1.12×
ResNet-50	ImageNet	1.00×	1.00×	1.00×
	ImageNet$^+$ (Ours)	1.04×	1.02×	0.97×
SwinTransformer-Tiny	ImageNet	1.00×	1.00×	1.00×
	ImageNet$^+$ (Ours)	0.99×	0.99×	0.99×

Table 19: **Training time for different models using ImageNet$^+$ is similar to ImageNet dataset**. Full results in Appendix A.

G. Hyperparameters and implementation details

We follow [42, 64] and use state-of-the-art recipes, including optimizers, hyperparameters, and learning. The details are provided in Tab. 20. Because of resource limitations, we train EfficientNet-B3/B4 with KD using batch size 512. Overall, we use the same hyperparameters on ImageNet and ImageNet$^+$ with the exception of the data augmentations that are removed from the training on ImageNet$^+$ based on our observations in Appendix C.2. For KD, we use the KL loss with temperature 1.0 (no mixing with the cross-entropy loss) and shrink the weight decay by $10\times$.

In Tab. 21 we provide hyperparameters for training with CVNets. For higher resolution and variable resolution training, we use the same metadata in ImageNet$^+$ to create a random crop then resize it to the target resolution instead of the base resolution of 224. In Tab. 22 we provide hyperparameters for training Detection/Segmentation models. In Tab. 23 we provide hyperparameters for transfer learning on CIFAR-100/Flowers-102/Food-101 datasets.

Model	**Training Method**	**Optimization Hyperparams**					**Data augmentation methods**				
		Optimizer	**Batch Size**	**LR**	**Warmup**	**Weight Decay**	**Label Smoothing**	**RandAugment**	**Random Erase** (p)	**MixUp** (α)	**CutMix** (α)
MobileNetV1	ImageNet	SGD+Mom=0.9	1024	0.8	3	$4.0e{-5}$	✓	✗	✗	✗	✗
	ImageNet$^+$	SGD+Mom=0.9	1024	0.8	3	$4.0e{-5}$	✓	✗	✗	✗	✗
	KD	SGD+Mom=0.9	1024	0.8	3	$4.0e{-6}$	✗	✗	✗	✗	✗
MobileNetV2	ImageNet	SGD+Mom=0.9	1024	0.4	3	$4.0e{-5}$	✓	✗	✗	✗	✗
	ImageNet$^+$	SGD+Mom=0.9	1024	0.4	3	$4.0e{-5}$	✓	✗	✗	✗	✗
	KD	SGD+Mom=0.9	1024	0.4	3	$4.0e{-6}$	✗	✗	✗	✗	✗
MobileNetV3	ImageNet	SGD+Mom=0.9	1024	0.4	3	$4.0e{-5}$	✓	✗	✗	✗	✗
	ImageNet$^+$	SGD+Mom=0.9	1024	0.4	3	$4.0e{-5}$	✓	✗	✗	✗	✗
	KD	SGD+Mom=0.9	1024	0.4	3	$4.0e{-6}$	✗	✗	✗	✗	✗
ResNet	ImageNet	SGD+Mom=0.9	1024	0.4	5	$1.0e{-4}$	✓	✓	✓(0.25)	✓(0.2)	✓(1.0)
	ImageNet$^+$	SGD+Mom=0.9	1024	0.4	5	$1.0e{-4}$	✓	✗	✗	✗	✗
	KD	SGD+Mom=0.9	1024	0.4	5	$1.0e{-5}$	✗	✓	✓(0.25)	✓(1.0)	✓(1.0)
EfficientNet-B2	ImageNet	SGD+Mom=0.9	1024	0.4	5	$4.0e{-5}$	✓	✓	✓(0.25)	✓(0.2)	✓(1.0)
	ImageNet$^+$	SGD+Mom=0.9	1024	0.4	5	$4.0e{-5}$	✓	✗	✗	✗	✗
	KD	SGD+Mom=0.9	1024	0.4	5	$4.0e{-6}$	✗	✓	✓(0.25)	✓(1.0)	✓(1.0)
EfficientNet-B3	ImageNet	SGD+Mom=0.9	1024	0.4	5	$4.0e{-5}$	✓	✓	✓(0.25)	✓(0.2)	✓(1.0)
	ImageNet$^+$	SGD+Mom=0.9	1024	0.4	5	$4.0e{-5}$	✓	✗	✗	✗	✗
	KD	SGD+Mom=0.9	512	0.2	5	$4.0e{-6}$	✗	✓	✓(0.25)	✓(1.0)	✓(1.0)
EfficientNet-B4	ImageNet	SGD+Mom=0.9	1024	0.4	5	$4.0e{-5}$	✓	✓	✓(0.25)	✓(0.2)	✓(1.0)
	ImageNet$^+$	SGD+Mom=0.9	1024	0.4	5	$4.0e{-5}$	✓	✗	✗	✗	✗
	KD	SGD+Mom=0.9	512	0.4	5	$4.0e{-6}$	✗	✓	✓(0.25)	✓(1.0)	✓(1.0)
ViT	ImageNet	AdamW (0.9, 0.999)	1024	0.001	5	0.05	✓	✓	✓(0.25)	✓(0.2)	✓(1.0)
	ImageNet$^+$	AdamW (0.9, 0.999)	1024	0.001	5	0.05	✓	✗	✗	✗	✗
	KD	AdamW (0.9, 0.999)	1024	0.001	5	0.005	✗	✓	✓(0.25)	✓(1.0)	✓(1.0)
SwinTransformer	ImageNet	AdamW (0.9, 0.999)	1024	0.001	5	0.05	✓	✓	✓(0.25)	✓(0.2)	✓(1.0)
	ImageNet$^+$	AdamW (0.9, 0.999)	1024	0.001	5	0.05	✓	✗	✗	✗	✗
	KD	AdamW (0.9, 0.999)	1024	0.001	5	0.005	✗	✓	✓(0.25)	✓(1.0)	✓(1.0)

Table 20: **Hyperparameters used for training different models.** We use cosine learning rate schedule to zero.

Model	Training Method	Optimization Hyperparams								Data augmentation methods
		Optimizer	Batch Size	LR	Warmup	Weight Decay	Mixed Precision	Resolution	Grad. Clip	
MobileNetV1	ImageNet	SGD+Mom=0.9	1024	0.8	3	4.0e−5	✓	224	✗	LS+RRC+HF
	ImageNet$^+$	SGD+Mom=0.9	1024	0.8	3	4.0e−5	✓	224	✗	✗
MobileNetV2	ImageNet	SGD+Mom=0.9	1024	0.4	3	4.0e−5	✓	224	✗	LS+RRC+HF
	ImageNet$^+$	SGD+Mom=0.9	1024	0.4	3	4.0e−5	✓	224	✗	✗
MobileNetV3	ImageNet	SGD+Mom=0.9	2048	0.4	3	4.0e−5	✓	224	✗	LS+RRC+HF
	ImageNet$^+$	SGD+Mom=0.9	2048	0.4	3	4.0e−5	✓	224	✗	✗
MobileNetViT	ImageNet	AdamW (0.9, 0.999)	1024	0.002	20	0.01	✓	VBS(160, 320, 256)	✗	LS+RRC+HF
	ImageNet$^+$	AdamW (0.9, 0.999)	1024	0.002	20	0.01	✓	VBS(160, 320, 256)	✗	✗
ResNet	ImageNet	SGD+Mom=0.9	1024	0.4	5	1.0e−4	✓	224	✗	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	SGD+Mom=0.9	1024	0.4	5	1.0e−4	✓	224	✗	✗
EfficientNet-B2	ImageNet	SGD+Mom=0.9	2048	0.8	3	4.0e−5	✓	VBS(144, 432, 288)	✗	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	SGD+Mom=0.9	2048	0.8	3	4.0e−5	✓	VBS(144, 432, 288)	✗	✗
EfficientNet-B3	ImageNet	SGD+Mom=0.9	2048	0.8	3	4.0e−5	✓	VBS(150, 450, 300)	✗	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	SGD+Mom=0.9	2048	0.8	3	4.0e−5	✓	VBS(150, 450, 300)	✗	✗
EfficientNet-B4	ImageNet	SGD+Mom=0.9	2048	0.8	3	4.0e−5	✓	VBS(190, 570, 380)	✗	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	SGD+Mom=0.9	2048	0.8	3	4.0e−5	✓	VBS(190, 570, 380)	✗	✗
ViT-Tiny	ImageNet	AdamW (0.9, 0.999)	2048	0.002	10	0.05	✓	224	✓(1.0)	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	AdamW (0.9, 0.999)	2048	0.002	10	0.05	✓	224	✓(1.0)	✗
ViT-Small/Base	ImageNet	AdamW (0.9, 0.999)	2048	0.002	10	0.2	✓	224	✓(1.0)	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	AdamW (0.9, 0.999)	2048	0.002	10	0.2	✓	224	✓(1.0)	✗
ViT-Base ↑384	ImageNet	AdamW (0.9, 0.999)	2048	0.002	20	0.2	✓	VBS(192, 576, 384)	✓(1.0)	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	AdamW (0.9, 0.999)	2048	0.002	20	0.2	✓	VBS(192, 576, 384)	✓(1.0)	✗
SwinTransformer	ImageNet	AdamW (0.9, 0.999)	1024	0.001	20	0.05	✓	224	✓(5.0)	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	AdamW (0.9, 0.999)	1024	0.001	20	0.05	✓	224	✓(5.0)	✗
SwinTransformer-Base ↑384	ImageNet	AdamW (0.9, 0.999)	1024	0.001	20	0.05	✓	VBS(192, 576, 384)	✓(5.0)	LS+RRC+HF+RA+RE+MU+CM
	ImageNet$^+$	AdamW (0.9, 0.999)	1024	0.001	20	0.05	✓	VBS(192, 576, 384)	✓(5.0)	✗

Table 21: **Hyperparameters used for training different models in CVNets.** LS: Label Smoothing with 0.1, RRC: Random-Resize-Crop, HF: Horizontal Flip, VBS(min-res, max-res, crop-size): Variable Batch Sampler with variable resolution. RA: RandAugment, RE: Random Erase with 0.25, MU: MixUp with alpha 0.2, CM: CutMix with alpha 0.1. We use cosine-learning rate schedule to 0.

Model	Training Method	Optimization Hyperparams										Data augmentation methods
		Optimizer	Epochs	Batch Size	LR	BackBone LR Mul.	Warmup iter.	Weight Decay	Mixed Precision	Resolution	Grad. Clip	
MobileNetV3-Large	Detection	SGD+Mom=0.9	36	64	multi-step-lr(0.1, [24, 33])	0.1	500	4.0e−5	✗	VBS(512, 1280, 1024)	✗	✗
	Segmentation	SGD+Mom=0.9	50	16	cosine-lr(0.02, 0.0001)	0.1	0	1.0e−4	✗	512	✗	RC+RSSR+RR+PD+RG
ResNet-50	Detection	SGD+Mom=0.9	100	64	multi-step-lr(0.1, [60, 84])	0.1	500	4.0e−5	✗	VBS(512, 1280, 1024)	✗	✗
	Segmentation	SGD+Mom=0.9	50	16	cosine-lr(0.02, 0.0001)	0.1	500	4.0e−5	✗	512	✗	RC+RSSR+RR+PD+RG
SwinTransformer-Tiny	Detection	SGD+Mom=0.9	100	64	multi-step-lr(6.0e−4, [60, 84])	1.0	500	0.05	✗	VBS(512, 1280, 1024)	✗	✗
	Segmentation	SGD+Mom=0.9	50	16	cosine-lr(6.0e−4, 1.0e−6)	0.1	500	0.05	✗	512	✗	RC+RSSR+RR+PD+RG

Table 22: **Hyperparameters of detection/segmentation using CVNets.** RC: Random Crop, RSSR: Random Short-Size Resize, RR: Random Rotate by maximum 10 degrees angle. VBS(min-res, max-res, crop-size): Variable Batch Sampler with variable resolution. PD: Photometric Distortion, RG: Random Gaussian noise

Model	Pretrained	Optimization Hyperparams					Data augmentation methods
		Optimizer	Batch Size	LR	Warmup	Weight Decay	
MobileNetV3-Large	✗	SGD+Mom=0.9	256	0.2	0	5.0e−4	✗
	✓	SGD+Mom=0.9	256	0.002	0	5.0e−4	✗
	✓+	SGD+Mom=0.9	256	0.002	0	5.0e−4	✗
ResNet-50	✗	SGD+Mom=0.9	256	0.2	0	5.0e−4	RA+MU+CM
	✓	SGD+Mom=0.9	256	0.002	0	5.0e−4	RA+MU+CM
	✓+	SGD+Mom=0.9	256	0.002	0	5.0e−4	✗
SwinTransformer-Tiny	✗	AdamW (0.9, 0.999)	256	0.0001	5	0.05	RA+MU+CM
	✓	AdamW (0.9, 0.999)	256	0.00001	5	0.05	RA+MU+CM
	✓+	AdamW (0.9, 0.999)	256	0.00001	5	0.05	✗

Table 23: **Hyperparameters used for CIFAR-100/Flowers-102/Food-101.** We use cosine learning rate schedule to zero. We resize the inputs for all datasets to 224 including CIFAR-100 where we pad the input by 16. We also use label smoothing.

H. CLIP, ViT, and Mixed Architecture Teachers

In this section, we evaluate the effectiveness of CLIP-pretrained models fine-tuned on ImageNet as teachers. We evaluate various ensembles teachers mixed with non-CILP pretrained teachers and a variety of ViT-based models. We provide the model names in Tab. 24. Table 25 shows the accuracy of various student models trained on reinforced datasets with our selection of ensembles. We observe 1) Ensembles are consistently better teachers 2) CLIP-pretrained teachers are at best on-par with the IG-ResNext ensemble 3) ViT-based teachers are not good teachers for CNN-based models, regardless of their training method.

Teacher Name	Timm name of Ensemble Member			
	1	2	3	4
CLIP	`vit_large_patch14_clip_224.openai_ft_in12k_in1k`	`vit_large_patch14_clip_224.openai_ft_in1k`	`vit_base_patch16_clip_224.openai_ft_in12k_in1k`	`vit_base_patch16_clip_224.openai_ft_in1k`
ViT	`vit_base_patch16_224`	`vit_base_patch8_224`	`vit_large_patch16_224`	`vit_small_patch32_224`
Mixed (RCVDx4).	`ig_resnext101_32x48d`	`convnext_xlarge_in22ft1k`	`volo_d5_224`	`deit3_huge_patch14_224`
Mixed (RCCVx4).	`ig_resnext101_32x48d`	`convnext_xlarge_in22ft1k`	`vit_large_patch14_clip_224.openai_ft_in1k`	`vit_base_patch16_224`

Table 24: CLIP, ViT, Mixed architecture teacher ensemble names.

Model	**Prev.**		**Mixed Archs**		**CLIP**			**ViT**	
	IN	**IN+**	**IN+-RCVDx4**	**IN+-RCCVx4**	**IN+-CLIPx1**	**IN+-CLIPx2**	**IN+-CLIPx4**	**IN+-ViTx1**	**IN+-ViTx4**
`MobileNetV3-Large`	74.7	**76.2**$_{+1.6}$	75.9$_{+1.2}$	75.9$_{+1.2}$	75.5$_{+0.8}$	75.5$_{+0.8}$	75.5$_{+0.8}$	74.3$_{-0.4}$	74.0$_{-0.6}$
`ResNet-50`	77.4	**79.6**$_{+2.3}$	**79.5**$_{+2.1}$	**79.4**$_{+2.0}$	79.2$_{+1.8}$	79.2$_{+1.8}$	79.3$_{+2.0}$	77.8$_{+0.4}$	78.0$_{+0.6}$
`Swin-Tiny`	79.9	**82.0**$_{+2.1}$	**81.9**$_{+2.0}$	**82.0**$_{+2.1}$	81.6$_{+1.7}$	81.6$_{+1.7}$	**81.8**$_{+1.9}$	80.0$_{+0.0}$	80.2$_{+0.3}$

(a) 150 epochs

Model	**Prev.**		**Mixed Archs**		**CLIP**			**ViT**	
	IN	**IN+**	**IN+-RCVDx4**	**IN+-RCCVx4**	**IN+-CLIPx1**	**IN+-CLIPx2**	**IN+-CLIPx4**	**IN+-ViTx1**	**IN+-ViTx4**
`MobileNetV3-Large`	74.9	**77.0**$_{+2.1}$	76.6$_{+1.7}$	76.7$_{+1.7}$	76.2$_{+1.3}$	76.3$_{+1.4}$	76.4$_{+1.5}$	75.1$_{+0.2}$	75.0$_{+0.1}$
`ResNet-50`	78.8	**80.6**$_{+1.8}$	**80.6**$_{+1.9}$	**80.4**$_{+1.7}$	80.0$_{+1.2}$	80.1$_{+1.3}$	80.3$_{+1.5}$	78.5$_{-0.3}$	78.6$_{-0.1}$
`Swin-Tiny`	80.9	**83.0**$_{+2.1}$	**82.9**$_{+2.0}$	**82.9**$_{+2.0}$	82.5$_{+1.6}$	82.6$_{+1.7}$	**82.9**$_{+2.0}$	80.7$_{-0.2}$	81.0$_{+0.1}$

(b) 300 epochs

Model	**Prev.**		**Mixed Archs**		**CLIP**			**ViT**	
	IN	**IN+**	**IN+-RCVDx4**	**IN+-RCCVx4**	**IN+-CLIPx1**	**IN+-CLIPx2**	**IN+-CLIPx4**	**IN+-ViTx1**	**IN+-ViTx4**
`MobileNetV3-Large`	75.1	**77.9**$_{+2.9}$	**77.7**$_{+2.6}$	77.4$_{+2.3}$	77.2$_{+2.1}$	77.0$_{+1.9}$	77.2$_{+2.1}$	76.0$_{+0.9}$	75.8$_{+0.7}$
`ResNet-50`	79.6	**81.7**$_{+2.1}$	81.4$_{+1.7}$	**81.5**$_{+1.8}$	81.1$_{+1.5}$	81.0$_{+1.4}$	81.1$_{+1.4}$	79.3$_{-0.3}$	79.6$_{-0.1}$
`Swin-Tiny`	80.9	**83.8**$_{+2.8}$	**83.7**$_{+2.8}$	**83.8**$_{+2.8}$	83.5$_{+2.5}$	**83.6**$_{+2.6}$	**83.7**$_{+2.7}$	81.3$_{+0.3}$	81.7$_{+0.8}$

(c) 1000 epochs

Table 25: **CLIP, ViT, Mixed architecture teachers.** Subscripts show the improvement on top of the ImageNet accuracy. We highlight the best accuracy on each row from our proposed datasets and any number that is within 0.2 of the best.