Supplementary Material

A. Speed Up Calculation

While Super Progressive Learning reduces the number of floating point calculations made by using a smaller resolution, Hard Augment adds an additional overhead for training which can be calculated by equations 9 and 11 respectively. Generally we find that we can reduce the number of steps by 1.7 times while reduce the number of floating point operations at each step by 1.4 times.

B. Additional Implementation Details

We have trained all our experiments in the single node setting with maximum 8 GPUs which restricted us to use 512 as the maximum batch size. Some of the SSL methods like SimCLR [6] and BYOL [22] have been shown to perform better with larger batch sizes which we could not replicate and restricted ourselves to the same training setting in all our experiments.

In some of the ablations and analysis experiments where we study the effect of a certain parameter different values can be used. For Linear evaluation we use freeze the encoder and re-initialize the last layer and train with batch size 2048 and learning rate 0.8 for 90 epochs using the LARS optimizer.

Our implementation is based on PyTorch Lightning [16] where we define Hard Augmentation and Super Progressive Learning Schedule as callbacks. We adapt the OneCycle Learning rate schedule implementation[3] for Extended Super Convergence. We will provide the code for our method upon publication. See the table for a full list of parameters Tab. 10.

C. Learning rate range finder

We have used the Learning rate range finder test proposed by [41] to find good values for minimum and maximum learning rates. This test increase learning rate exponentially and keeps track of on the training and validation loss. The minimum value and maximum value are determined by the point at which the validation loss starts to decrease and the point at which is starts to diverge. We calculate the validation loss at each step on a batch of 4096 validation samples in order to keep the test manageable.

In the test shown in Figure 4 we increase the learning rate from 1×10^{-3} to 1 in 200 steps. Both the training and validation losses plateau close to step 135 and learning rate .1 which we use in our experiments on Imagenette dataset.

[3] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html

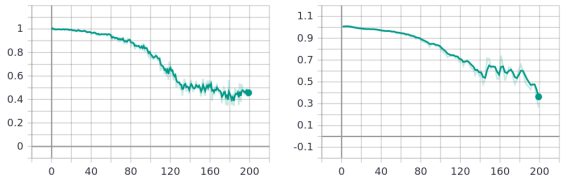

Figure 4. Learning rate finder plot for training (left) and validation (right) losses

D. Augmentation Resolution Relationship

In order to examine the relationship between resolution and augmentation magnitude for self-supervised learning we have conducted a series of experiments. We trained a BYOL [22] model on the Imagenette dataset using the ResNet18 [24] architecture with RandAugment [12] augmentation with the cosine annealing learning rate schedule where we change the augmentation strength m while we change the input image resolution. Each entry in the table is a separate experiment with fixed input resolution and augmentation magnitude. The results in Table 11 confirms that we can apply higher magnitude augmentations only on higher resolution images and the optimum augmentation magnitude increases with resolution. This confirms our intuition behind the relationship between resolution and augmentation magnitude in SSL and provides motivation for applying Super Progressive Learning and allows us to confirm the findings of [44] for SSL.

E. Progressive Augmentation Curriculum

However what values should be used as the minimum and maximum augmentation magnitude is an important question. Since we use the linearly scaled SimCLR [6] augmentations in our method we made a hyper-parameter search on these values in order to determine augmentation magnitudes empirically. We train our method for 320 epochs with Super Progressive learning schedule using 128 as the minimum resolution for various augmentation magnitudes and compare against the default augmentation magnitude of 5. Our experiment shows that a minimum value of 4 and maximum value of 6 perform better in our setting and outperform the fixed augmentation setting.

F. Future Work

This idea can be extended to other modalities where resolution is defined in different manners for example on sound with the sampling rate analogue which in a similar sense accelerates the training however than the amount of acceleration and cost benefit calculation will change. A similar case can also be made for tabular data where the most important features are processed first and later additional features are added however this is much more difficult to implement

	Imagenette		ImageNet	
Hyper-parameter	Baseline	Efficient	Baseline	Efficient
Number of classes	10	10	1000	1000
Batch size	128	128	512	512
Weight decay	5×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-4}
Learning rate	0.05	0.1	0.1	0.16
LR schedule	CA	ESC	CA	ESC
Warmup epochs	0	80	10	10
Momentum weight	0.9	0.85-0.95	0.9	0.85-0.95
Learning rate	0.05	0.05	0.05	0.16
Min aug. magnitude	5	4	5	4
Max aug. magnitude	5	6	5	6
Min view resolution	224	96	224	96
Number of positives	2	6	2	6
Selection resolution	N/A	64	N/A	64

Table 10. Hyper parameters used in various setups. CA: Cosan Annealing, ESC: Extended Super Convergence

	m=3	m=5	m=7	m=10	m=15
128	85.0	84.0	81.4	77.7	75.3
192	88.8	87.7	88.9	86.8	87.3
300	89.5	89.7	89.9	88.3	87.7

Table 11. Resolution and Augmentation Magnitude relationship shown by training a self-supervised learning method on the combination of resolution and magnitudes and measuring the online classification accuracy of a linear layer

m_{min}	m_{max}	Acc(%)
5	5	88.6
2.5	4	88.4
3	4	88.7
4	5	88.7
5	6	88.7
4	6	88.9

Table 12. Maximum and minimum augmentation magnitude when trained with Super Progressive learning

would probably require additional structure. Similarly training with a small vocabulary and than enlarging that can have a similar effect as well.