Supplementary Material for Contextual Classification Using Self-Supervised Autogenous Learning

July 15, 2020

Contents

A Clustering

Given a distance matrix $\mathcal{D}_c$ for a set of labels $\mathcal{Y}_c$, a split into k disjoint subsets are computed with the following clustering algorithm, based on the main principles of spectral clustering:

The clustering is split into two phases. The initialization phase populates all k clusters with at least one label first. In this phase, the next label is picked from $\mathcal{Y}_c$ with the largest distance to all already populated clusters according to $\mathcal{D}_c$. That label then differs from all other labels distributed so far, and it will populate a new cluster. When all clusters are populated with at least one label, the clustering phase begins, changing

the preference for the next label. The next label is chosen from the remaining labels as the one with the smallest average distance to any of the clusters. The label is assigned to that cluster of minimal average distance, if the cluster has not reached the predefined maximum size. Otherwise, the label is assigned to the next best cluster that is not full. If the label has the same distance to multiple clusters, it is assigned to a random one of those.
By limiting the size of the clusters, an even split of $\mathcal{Y}_c$ into subsets can be assured, which is not the case for spectral clustering.

B Networks

B.1 Tiny Imagenet

layer name	output size	Resnet18
Conv1	32×32	7×7, 64, stride 2
Conv2	16×16	3×3 maxpool, stride 2
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$
Conv3	8×8	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$
Conv4	4×4	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$
Conv5	2×2	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$
GAP	1×1	Global average pool
Output	1×1	FC-200, softmax

Table 1: Architecture for Tiny ImageNet. Downsampling is performed in the first layer of Conv3, Conv4 and Conv5. The residual blocks make use of the full pre-activation order (BN-ReLU-Conv), as proposed in [1].

B.2 CIFAR100

layer name	output size	Resnet50
Conv1	32×32	5×5, 64
Conv2	32×32	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$
Conv3	16×16	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$
Conv4	8×8	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$
Conv5	8×8	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$
GAP	1×1	Global average pool
Output	1×1	FC-100, softmax

Table 2: Resnet architecture for CIFAR100. Downsampling is performed in the first layer of Conv3 and Conv4. Residual blocks make use of the full pre-activation order (BN-ReLU-Conv), as proposed in [1].

layer name	output size	SE-WRN 16-8	WRN 28-10
Conv1	32×32	3×3, 16	
Conv2	32×32	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times2$	$\begin{bmatrix} 3\times3, 160 \\ 3\times3, 160 \end{bmatrix} \times4$
Conv3	16×16	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times2$	$\begin{bmatrix} 3\times3, 320 \\ 3\times3, 320 \end{bmatrix} \times4$
Conv4	8×8	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times2$	$\begin{bmatrix} 3\times3, 640 \\ 3\times3, 640 \end{bmatrix} \times4$
GAP	1×1	Global average pool	
Output	1×1	FC-100, softmax	

Table 3: Wide Residual Network architectures for CIFAR100. Downsampling is performed in the first layer of Conv3 and Conv4. Convolutional layers are succeeded by BatchNorm and a ReLU (conv-BN-ReLU). The Squeeze-and-Excitation version utilizes an SE block at the end of each residual block, before the shortcut is added (see [2])).

layer name	output size	DenseNet-BC 100-12	DenseNet-BC 190-40
Conv	32×32	3×3, 24	3×3, 80
Dense Block 1	32×32	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix}\times16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix}\times31$
Transition Block 1	16×16	1x1	
		2x2 average pool	
Dense Block 2	16×16	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix}\times16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix}\times31$
Transition Block 2	8×8	1x1	
		2x2 average pool	
Dense Block 3	8×8	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix}\times16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix}\times31$
GAP	1×1	Global average pool	
Output	1×1	FC-100, softmax	

Table 4: DenseNet architectures for CIFAR100. Convolutional layers are preceeded by BatchNorm and ReLU (BN-ReLU-conv). The number of filters increases with each conv block by $k = 12$ or 40, respectively. Each conv block within a dense block receives as input the concatenated output of all previous conv blocks of that same dense block. Compression of 0.5 is used. Therefore, each transition block halves the number of filters.

B.3 Imagenet

layer name	output size	Resnet50
Conv1	112×112	7×7, 64, stride 2
Conv2	56×56	3×3 maxpool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$
Conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$
Conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$
Conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$
GAP	1×1	Global average pool
Output	1×1	FC-1000, softmax

Table 5: Resnet-architecture for Imagenet. Downsampling is performed in the first layer of Conv3, Conv4 and Conv5. Convolutional layers use the original conv-BN-ReLU order.

C Auxiliary Architectures

The auxiliary networks consist of different combinations of conv-BN-ReLU blocks without any shortcut connections, Inception blocks, and one or more fully connected layers.

C.1 Tiny Imagenet

layer name	AuxNet1	AuxNet2
AuxConv1	[3×3, 128] ×2	-
AuxConv2	[3×3, 256] ×2	
GAP	Global average pool	
FC1	FC-256, ReLU	
Output	FC-200, softmax	

Table 6: Architecture of auxiliary networks for Tiny ImageNet. When attaching an auxiliary classifier to positions g_1 or g_2 of the main architecture, the AuxNet1 layout is used (Figure 4 in the main paper). The number of output units depends on the number of groups used in the SSAL objective.

C.2 CIFAR100

layer name	AuxNet1	AuxNet2	AuxNet3
AuxConv1	[5×5, 128, stride 2]	[5×5, 128]	[3×3, 128]
AuxConv2	[3×3, 128]		
AuxConv3	AuxInceptionE		
GAP	Global average pool		
Output	FC-100, softmax		

Table 7: Architecture of auxiliary networks on Resnet50 for CIFAR100. The number of output units for the last layer depends on the number of groups used in the SSAL objective. For our main result, AuxNet1 is attached on position g_2 (see Figure 4 on the paper), AuxNet2 on position g_3 and AuxNet3 on position g_4 of the main network. The layout of the AuxInceptionE block is shown in Figure 1c

.

layer name	AuxNet1	AuxNet2	AuxNet3
AuxConv1	$\begin{bmatrix} 5\times5,\ 128,\ \text{stride}\ 2 \\ 3\times3,\ 128 \end{bmatrix}$	$\begin{bmatrix} 5\times5,\ 128 \\ 3\times3,\ 128 \end{bmatrix}$	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}$
AuxConv2	AuxInceptionE	-	AuxInceptionE
GAP	Global average pool		
FC1	-	FC-768, ReLU	-
Output	FC-100, softmax		

Table 8: Architecture of auxiliary networks on WRN 16-8 for CIFAR100. The number of output units for the last layer depends on the number of groups used in the SSAL objective. The layout of the AuxInceptionE block is shown in Figure 1c

.

layer name	AuxNet1	AuxNet2	AuxNet3
AuxConv1	$\begin{bmatrix} 5\times5,\ 128,\ \text{stride}\ 2 \\ 3\times3,\ 128 \end{bmatrix}$	$\begin{bmatrix} 5\times5,\ 128 \\ 3\times3,\ 128 \end{bmatrix}$	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}$
AuxConv2	$\begin{bmatrix} 3\times3,\ 256 \end{bmatrix}\times2$		-
AuxConv3	$\begin{bmatrix} \text{AuxInceptionE} \end{bmatrix}\times2$	-	$\begin{bmatrix} \text{AuxInceptionE} \end{bmatrix}\times2$
GAP	Global average pool		
FC1	-	FC-768, ReLU	-
Output	FC-100, softmax		

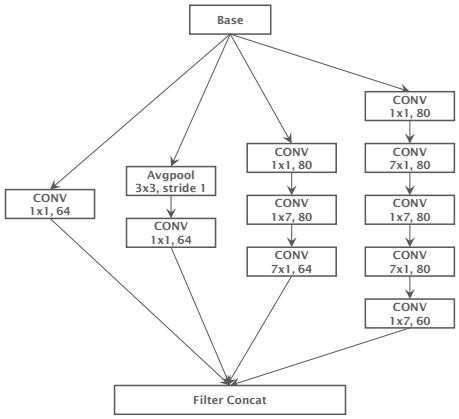
Table 9: Architecture of auxiliary networks on WRN 28-10 and DenseNet models for CIFAR100. The number of output units for the last layer depends on the number of groups used in the SSAL objective. For our main result, AuxNet1 is attached on position g_2 (see Figure 4 on the paper), AuxNet2 on position g_5 and AuxNet3 on position g_6 of the original network. The layout of the AuxInceptionE block is shown in Figure 1c

.

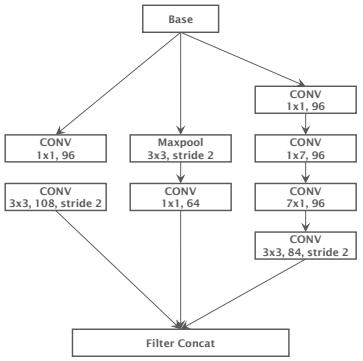
C.3 Imagenet

layer name	AuxNet1	AuxNet2	AuxNet3
AuxConv1	[3×3, 256, stride 2]	-	-
AuxConv2	AuxInceptionC		
AuxConv3	[AuxInceptionD AuxInceptionE]		-
AuxConv4	[AuxInceptionD AuxInceptionE*]		
GAP	Global average pool		
Output	FC-100, softmax		

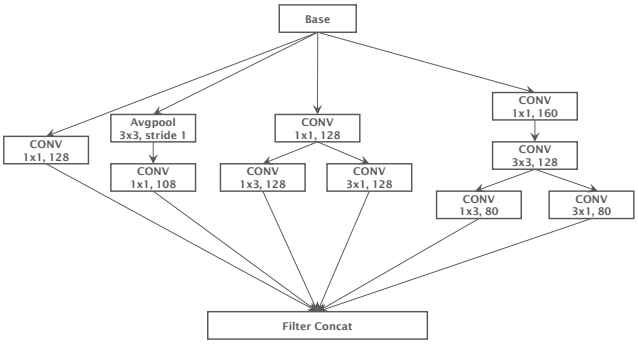
Table 10: Architecture of auxiliary networks on Resnet50 for Imagenet. The layout of the AuxInception blocks is shown in Figure 1. For our main result, AuxNet1 is attached on position g_2 (see Figure 4 of the paper), AuxNet2 on position g_3 and AuxNet3 on position g_4 of the original network.
*The final AuxInceptionE block has twice as many channels as the first AuxInceptionE found in AuxConv3.



(a) AuxInceptionC



(b) AuxInceptionD



(c) AuxInceptionE

Figure 1: Layout of AuxInception blocks, based on Inception blocks.

D CAM Visualizations

We show additional visualizations of CAM heatmaps together with SSAL groupings.

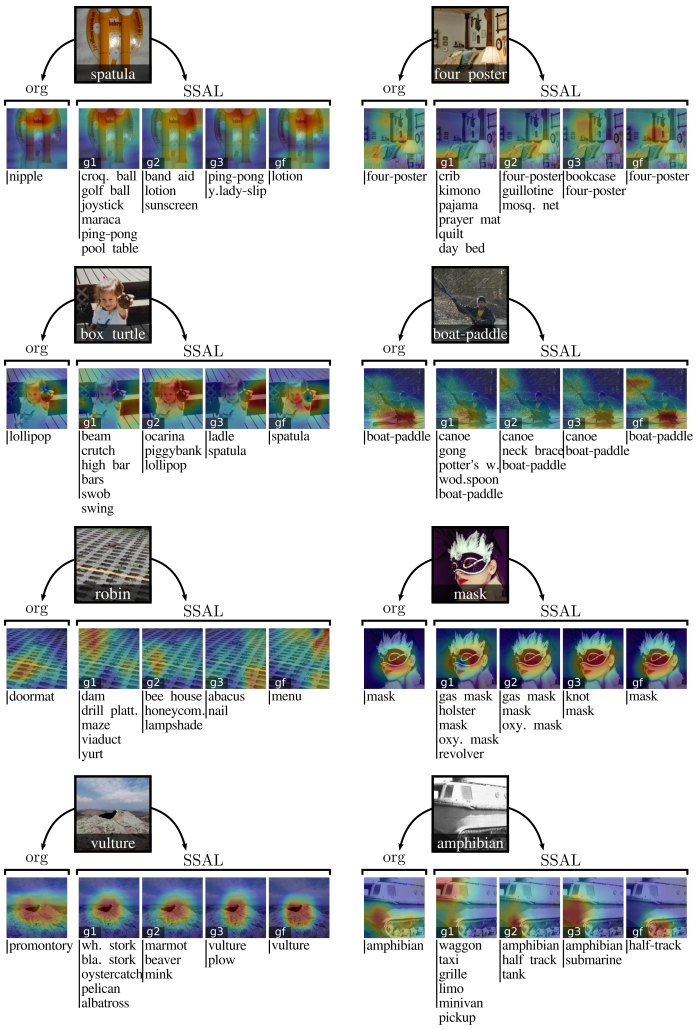


Figure 2: CAM samples with and without SSAL. The left column has incorrect predictions. The right column has correct predictions. The last row has mixed results.

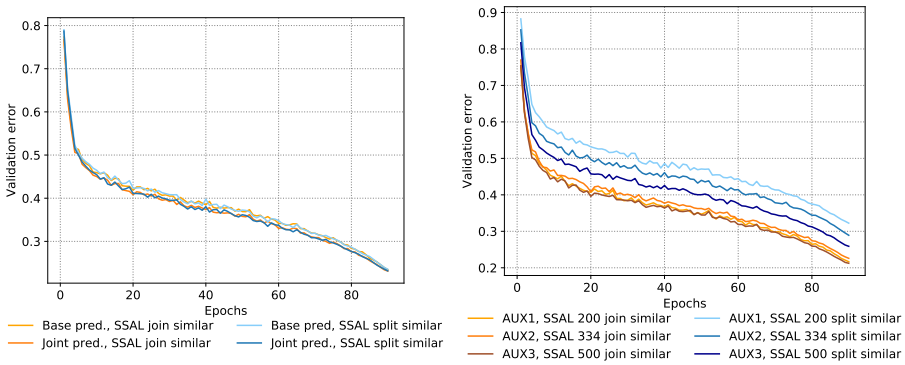
E Additional Experiments

E.1 Alternative Joint Predictions

WRN 28-10 is trained on CIFAR100 with one SSAL branch, forcing a single prediction that follows the operations done by a joint prediction. This allows the network to be trained as a regular network (i.e., with a single classification output as opposed to an MTL setup), while preserving all aspect of the SSAL architecture when used for predictions. Note that in this case, the grouping objective is **not** modeled in the loss; it is only indirectly expressed by the way the SSAL branch is wired to the main classifier output. With the right amount of weight decay, this setup reaches a performance that is on par with a similar SSAL model (trained using an MTL regime) with an explicit pointwise multiplication for prediction.

E.2 Training with different Grouping Criteria

Resnet-50 is trained on ImageNet with 3 SSAL branches of 200, 334 and 500 groups respectively. The ground-truth labels are divided into groups either by joining or splitting visually similar labels. The validation accuracy is observed throughout training (see Figure 3). The curves are close together for the base prediction error as well as for the joint prediction error, which indicates similar benefits from both criteria for the prediction. However, the validation error of the auxiliary predictions differ. When splitting similar labels into different groups, which creates subsets of diverse labels, the error in the auxiliary classifiers is higher. This meets the expectation, as it is intuitively more difficult to find features to represent each group. Also, contrary to groups of similar labels, the classes that are more often confused still need to be distinguished by the classifier.



(a) (b)

Figure 3: Validation error on a) base and joint prediction as well as b) the auxiliary predictions.

The combined loss is higher for split similar than for join similar, which meets the

expectation as joining similar labels creates an easier problem.

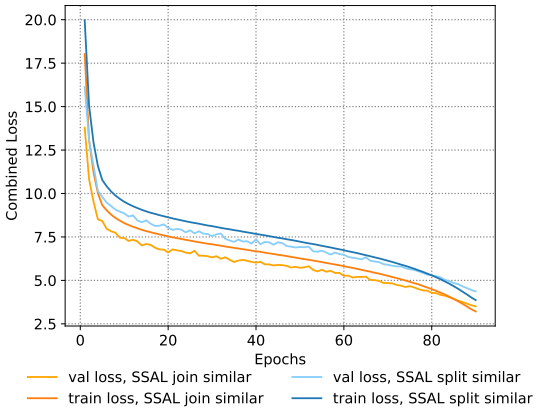
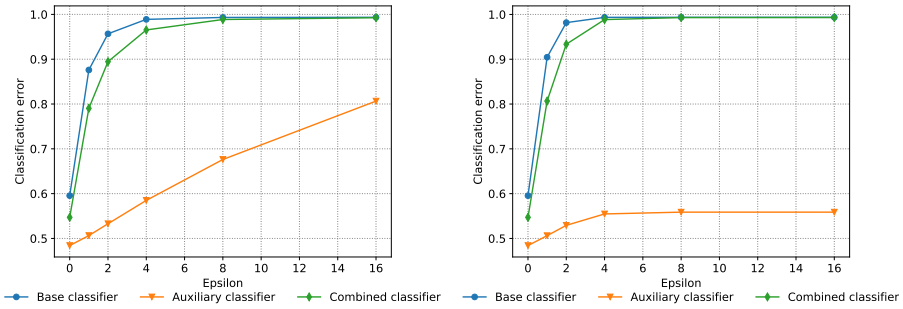


Figure 4: Combined validation and training loss throughout training.

E.3 Adversarial Attacks

Adversarial examples are generated from the Tiny Imagenet dataset using the Fast Gradient Sign Method (FGSM) [3] as well as the Basic Iterative Method (BIM) [4]. Attacks are aimed for the base network only, while auxiliary classifiers are not available to the attacker. The base network is a ResNet18 model (see Table 1) trained on Tiny Imagenet.

The effect of FGSM attacks is shown in Figure 5a. It is evaluated for different values for ε, which limits the alteration per pixel from the original image. While the auxiliary classifier is not affected as much by the attack against the base network as the base network itself, the combination of both still shows poor robustness.
For BIM (see Figure 5b), the auxiliary classifier is influenced even less. This is likely due to the multi-step creation of the adversarial samples, which makes them more effective towards the given base network, but less transferable to the auxiliary classifier. Once more, the auxiliary classifier barely decreases the amount of errors on the validation set when looking at the combined prediction. Further improvements can be gained by using multiple auxiliary classifiers, but overall, the use of the auxiliary classifiers against adversarial attacks is limited.



(a) FGSM (b) BIM

Figure 5: Effect of adversarial examples on the validation error. The auxiliary classifier classifies 40 groups of joint similar labels.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016. 2, 3

[2] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[3] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. 12

[4] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world." *Technical report, arXiv 1607.02533*, 2016. 12