Supplementary Material to *HyperSparse Neural Networks: Shifting Exploration to Exploitation through Adaptive Regularization*

Patrick Glandorf*, Timo Kaiser*, Bodo Rosenhahn
Institute for Information Processing (tnt)
L3S - Leibniz Universität Hannover, Germany
{glandorf, kaiser, rosenhahn}@tnt.uni-hannover.de

This document provides supplementary material for the paper *HyperSparse Neural Networks: Shifting Exploration to Exploitation through Adaptive Regularization*. At first, Sec. A gives detailed information about the implementation of our method. Subsequently, Sec. B presents more detailed results of the intersection of largest weights during training and the final pruning mask. The weight distribution after training with our introduced method shown in the main paper is analyzed for a wider set of configurations in Sec. C. Moreover, Sec. D and Sec. E elaborate the gradient and the compression behaviour during regularization presented in the main paper more into detail.

A. Detailed Experimental Setup

As described in [14], we evaluated our method on the datasets CIFAR-10/100 [9] and TinyImageNet [2] with the models ResNet-32 [6] and VGG-19 [13]. CIFAR-10 is a dataset for a classification task with $50\,000$ training and $10\,000$ validation samples on 32x32 color-images labeled with 10 classes. Respectively CIFAR-100 has 100 classes and the same amount of samples. The dataset TinyImageNet consists of $100\,000$ training and $10\,000$ validation samples with an image-size of 64x64, where samples are labeled with a set of 200 classes.

As done in [14], we train our models for 160 epochs on CIFAR-10/100 and 300 epochs on TinyImageNet using SGD-optimizer, with an initial learning rate of 0.1 and a batch size of 64. We decay the learning rate by factor 0.1 at epoch 2/4 and 3/4 of the total number of epochs. The weight decay is set to $1 \cdot 10^{-4}$. In our experiments all results are averaged over 5 runs.

In the original implementation of *SmartRatio* [14], weights in the final linear layer are pruned with a fixed pruning rate of 70%. Thus, too much weights remain when training on ResNet-32 with a pruning ratio of 99.8% on dataset CIFAR-100 and TinyImageNet. To this reason, we change the pruning ratio in the linear layer to 90% for this two training settings only. The methods *SNIP* [10], *GraSP* [16], *SmartRatio* [14], and *LTH* [5] suggest rules to obtain fixed masks. This mask is applied to the model weights before training. In contrast, *IMP* [7] iteratively trains a model to epoch T and prunes 20% of the remaining weights until the desired pruning rate is reached. After each iteration the weights and learning rate are reset to epoch k and retrained again to epoch T. To be comparable, we define $k = 20$ and $T = 160$ for CIFAR-10/100 as well as $k = 40$ and $T = 300$ for TinyImageNet. As described in [4], RigL performes better with a longer training duration. To this reason we extend the optimization time of the uniform distributed RigL-method by training for 360 epochs with a learning rate of 0.1, followed by the fine-tuning-step of 160 epochs on CIFAR-10/100 and 300 epochs on TinyImageNet. The fine-tuning step is equal to ART. All further hyperparameters of RigL are adopted from [4].

Our proposed method *ART*, described in Sec. 3.2 in the main paper, consists of three steps. In the first step we train our model to convergence for 60 epochs using a fix learning rate of 0.1. Subsequently we enable the used regularization term, with a small initialisation rate of $\lambda_{init} = 5 \cdot 10^{-6}$ and increasing factor of $\eta = 1.05$. To reduce noise in choosing the best pruned model, we average the accuracy $\psi(\nu(W_e) \odot W_e)$ over epoch $(e - 1,\ e,\ e + 1)$, where e describes the current epoch and ν denotes magnitude pruning that obtains a binary mask. The first two steps are used to obtain the weights and masks for fine-tuning. During fine-tuning, we use the training schedule described above as done in [14].

B. Mask intersection in Regularized Training

In this section we show further results of our experiments measuring the mask intersection over epoch e from Sec. 4.4 in the main paper. We measure the relative overlap between the weights with highest magnitude at epoch e and the final mask in different settings with different models, datasets, regularization losses, and pruning rates. Therefore, Tab. 1 shows the important keypoints of intersection at the end of

*These authors contributed equally to this work

pre-training (epoch 60) and one epoch before the final mask was found ($e = K - 1$). We observe that our regularization loss $\mathcal{L}_{\text{HS}}$ has a higher intersection in nearly all settings at epoch 60 and epoch $K-1$ compared to $\mathcal{L}_1$ and $\mathcal{L}_2$ loss. This indicates that our *HyperSparse* loss changes less parameter while reordering weights from remaining to pruned and vice versa.

In addition, Tab. 1 presents the total number of training epochs to obtain the final mask (including step 1 and step 2). It shows that our *HyperSparse* loss needs less epochs to terminate in nearly all settings. Since *ART* terminates, if the best pruned model outperforms the unpruned model at epoch e, we deduce that $\mathcal{L}_{\text{HS}}$ creates a well performing sparse network faster compared to $\mathcal{L}_1$ and $\mathcal{L}_2$ loss.

C. Weight Distribution

In this section, we show further experiments of the weight distribution per layer in the final mask, as evaluated in Sec. 4.4 in the main paper. We analyse the resulting masks for dataset CIFAR-10 and CIFAR-100, pruning-rate $\kappa \in \{90\%, 98\%, 99.5\%\}$ as well as for model ResNet-32 and VGG-19. Weight distributions obtained by the methods *IMP* [7], *SRatio* [14] and *ART* using *HyperSpase* loss are analyzed. All values are averaged over 5 runs.

In Fig. 1, we show the resulting weight distributions for ResNet-32. Note that the model is grouped in three residual blocks (RES), two downsampling blocks (DS) and a linear layer (LL). We observe that *ART* + $\mathcal{L}_{\text{HS}}$ and *IMP* have comparable distributions of weights. Both methods show a relative constant distribution in the residual layers, except the last one. This last layer has an decreasing number of weights, especially in the simpler task given in CIFAR-10. In comparison, *SRatio* uses a fixed keep-ratio in a quadratic decreasing manner and thus the weight distribution is not dependent on data. Since *ART* and *IMP* outperform *SRatio* by far in accuracy (Tab. 1 in the main paper), this handcrafted rule has adverse effect on performance. Moreover, we observe a relatively high number of weights in the downsampling layer for *ART* and *IMP*, which indicates that these layers are more important.

Further, we present the weight distribution for VGG-19 in Fig. 2. We observe that the layer around index 5 has more weights for *ART* and *IMP*. Nearly no weights remain in layer with index higher than 10, except the final linear layer. Considering the increasing sparsity, the weight distribution is shifted towards the earlier layers with low index. We deduce that in higher sparsity regimes the weight in earlier layer are more important in VGG-19. The handcrafted rule of *SRatio* shows a relatively flat weight distribution. Overall, the number of weights in the linear layer increases for CIFAR-100, due to the increasing number of classes compared to CIFAR-10 in ResNet-32 and for VGG-19.

D. HyperSparse Gradient Analysis

In this section, we analyze the gradient of our *HyperSparse* regularization loss with respect to the model weights $w \in W$. Assuming that important weights have large magnitudes, we show that *HyperSparse* subsides to no regularisation for important values and evolves to a strong penalization for unimportant values. This behaviour allows exploitation in the set of the important weights that remain after magnitude pruning. Furthermore, we show that our loss ensures a smooth transition in the gradient between unimportant and important weights, such that exploration in the set of unimportant weights is possible during training.

Gradient. Our loss evolves sparseness and adapts on the weight magnitude by utilizing the non-linearity of the *Hyper*bolic Tangent function

$$\tanh(x) = \frac{e^x - e^{-x}}{e^z + e^{-x}} \quad \in (-1, 1), \tag{1}$$

which is the reason for the name *HyperSparse*. The maximum of the derivative of $\tanh$ is 1 at $x = 0$ and strongly vanishes close to zero for large values:

$$\arg\max_x \frac{d\tanh(x)}{dx} = 0 \quad \text{and} \quad \lim_{x \to \pm\infty} \frac{d\tanh(x)}{dx} = 0\,. \tag{2}$$

In this paper, the Hyperbolic Tangent function of a magnitude $\tanh(|\cdot|)$ is denoted by $t(\cdot)$ for simplicity.

For the sake of completeness, we recapitulate the definition of *HyperSparse* from Eq. (2) in the main paper:

$$\begin{aligned} \mathcal{L}_{\text{HS}}(W) = & \frac{1}{A} \sum_{i=1}^{|W|} \left(|w_i| \sum_{j=1}^{|W|} t(s \cdot w_j) \right) - \sum_{i=1}^{|W|} |w_i| \\ & \text{with} \quad A := \sum_{w \in W} t(s \cdot w) \\ & \text{and} \quad \forall w \in W: \quad \frac{dA}{dw} = 0, \end{aligned} \tag{3}$$

and want to note again, that A denotes a pseudo-constant term that is considered to be a constant in the gradient computation, and s is a scaling factor discussed in the end of this section. In this section, sum notations as $\sum_{i=1}^{|W|}$ will be simplified by $\sum_{w_i}$ or by $\sum_w$ if w is unique. Furthermore, we will leave out declarations of set memberships like $w_i \in W$ and state that every w is in the set of model weights W. Also the scope of formulations is consistently defined as $\forall w \in W$.

With this notations and simplifications, the derivative of

Eq.(3) w.r.t. to a weight w_i can be defined as follows:

$$\begin{aligned}\frac{d\mathcal{L}_{HS}}{dw_i} =& \underbrace{\frac{d}{dw_i}\frac{|w_i|\sum_{w_j} t(s\cdot w_j)}{A}}_{\mathbf{I}} + \\ & \underbrace{\frac{d}{dw_i}\frac{\sum_{w_n\neq w_i}|w_n|\sum_{w_j} t(s\cdot w_j)}{A}}_{\mathbf{II}} - \text{sign}(w_i) \\ =& \text{sign}(w_i)\cdot \underbrace{\frac{|w_i|\cdot t'(s\cdot w_i) + \sum_{w_j} t(s\cdot w_j)}{A}}_{\mathbf{I}} + \\ & \text{sign}(w_i)\cdot \underbrace{\frac{\sum_{w_n\neq w_i}|w_n|\cdot t'(s\cdot w_i)}{A}}_{\mathbf{II}} - \text{sign}(w_i) \\ =& \text{sign}(w_i)\cdot\left[\frac{\sum_{w_j} t(s\cdot w_j)}{A=\sum_{w_j} t(s\cdot w_j)} + \right. \\ & \left. \frac{\sum_{w_n}|w_n|\cdot t'(s\cdot w_i)}{A=\sum_{w_j} t(s\cdot w_j)} - 1\right] \\ =& \text{sign}(w_i)\cdot\frac{t'(s\cdot w_i)\cdot\sum_{w_n}|w_n|}{\sum_{w_j} t(s\cdot w_j)} \quad .\end{aligned} \tag{4}$$

The gradient consists of a term that is depending on the weight distribution in W and the derivative $t' = \frac{dt}{dw_i}$ at the considered weights magnitude $|w_i|$ scaled with s. The behaviour of *HyperSparse* can be explained with the gradients for very small and very large magnitudes: For large magnitudes $|w_i| \gg 0$, the derivative in Eq. (4) collapses to

$$\left.\frac{d\mathcal{L}_{HS}}{dw_i}\right|_{|s\cdot w_i|\gg 0} \approx \text{sign}(w_i)\cdot 0 = 0 \tag{5}$$

which is effectively no regularisation. For very small values $w_i \approx 0$, the derivative

$$\left.\frac{d\mathcal{L}_{HS}}{dw_i}\right|_{|s\cdot w_i|\approx 0} \approx \text{sign}(w_i)\cdot\frac{\sum_{w_n}|w_n|}{\sum_{w_j} t(s\cdot w_j)} \tag{6}$$

is larger and increases, if the weights in W are clearly separated in two sets of important (large magnitude) and unimportant weights (low magnitude). The gradient of weights that are not assigned to one of those sets is between Eq. (5) and (6) and therefore allows an easier exploration of those weights during training.

Aligning with s. In the definition of *HyperSparse*, the scaling factor s aligns the loss with the actual weight distribution. The aim is that weights $|w| > |w_\kappa|$ are not or softly and $|w| < |w_\kappa|$ strongly penalized. A weight distribution does not need to be aligned with the derivative of the Hyperbolic Tangent function such that large weights are mapped close to 0 and small weights close to 1. To fix this, we align W by scaling w_κ with s so that it lies on the inflection point of the gradient. The desired scaling factor can be derived by

$$t'''(s\cdot|w_\kappa| \approx 0.6585) = 0 \tag{7}$$

and setting $s = \frac{0.6586}{|w_\kappa|}$. Large scaling factors s lead to rampant gradient distribution at weight w_κ towards weights of low magnitude. Examples can be found in Fig. 1 in the main paper.

			#Epochs to Final Mask			Intersection at $e = 60$			Intersection at $e = K-1$		
		κ	$\mathcal{L}_{\text{HS}}$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_{\text{HS}}$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_{\text{HS}}$	$\mathcal{L}_1$	$\mathcal{L}_2$
CIFAR-10	**ResNet-32**	90%	87	95	23	0.46	0.35	0.41	0.84	0.72	0.83
		98%	112	130	194	0.29	0.15	0.1	0.86	0.54	0.4
		99.5%	136	152	177	0.23	0.12	0.17	0.91	0.54	0.49
	VGG-19	90%	64	66	185	0.77	0.71	0.46	0.87	0.83	0.67
		98%	76	81	230	0.51	0.45	0.02	0.83	0.76	0.7
		99.5%	104	118	232	0.32	0.19	0.02	0.82	0.59	0.52
CIFAR-100	**VGG-19**	90%	68	71	211	0.66	0.6	0.13	0.85	0.81	0.67
		98%	96	106	238	0.39	0.28	0.02	0.83	0.64	0.66
		99.5%	116	121	226	0.27	0.16	0.01	0.85	0.62	0.48
	ResNet-32	90%	99	113	177	0.45	0.28	0.2	0.88	0.63	0.49
		98%	123	134	183	0.34	0.2	0.18	0.91	0.61	0.49
		99.5%	147	150	163	0.26	0.19	0.28	0.95	0.7	0.62

Table 1: Complementary key-points to the experiments about mask intersection in Sec. 4.4 in the main paper. The intersection indicates the relative overlap of weights with highest magnitude during training to remaining weights in the final mask obtained by *ART*. For simplicity, we only show the intersection at the end of pre-training ($e = 60$) and one epoch before the final mask was found ($e = K - 1$). In addition, this table shows the number of training epochs to the final mask. The information are demonstrated for $\mathcal{L}_1$, $\mathcal{L}_2$ and *HyperSparse* loss $\mathcal{L}_{HS}$, over different pruning rates κ, models and datasets.

E. Interpretable Compression

This chapter discusses the process of knowledge compression that compresses patterns from a pre-trained dense network into a sparse network that consists of the set of $1-\kappa$ highest weights, where κ denotes the desired pruning rate.

The first subsection presents the CIFAR-N [17] dataset that is used to analyze the compression behavior in Sec. 4.5. We show how it relates to CIFAR [9] and how we visualize the label distribution with the modern *CLIP* framework [12]. Then we elaborate the introduced metric *Compression Position* more in detail. For the sake of completeness, we lastly present and discuss figures and tables that show results for additional settings that could not be presented in the main paper due to lack of space.

CIFAR-N

To analyze human-like label errors and to provide real-world label noise for researchers, *Wei et al*. introduced the CIFAR-N [17] dataset that uses the CIFAR [9] training data $S = \{(x_n, y_n)\}_{n=1}^{N}$, but has different ground truth labels. Every sample was labeled by 3 different persons, inducing their subjective human bias, such that the dataset formulation can be defined as $S = \{(x_n, \{y_n^1, y_n^2, y_n^3\})\}_{n=1}^{N}$. They show, that single persons consistently induce an error rate between 10-20% (compared to original CIFAR). Moreover, they show that human-like label noise is harder to tackle in robust learning scenarios compared to synthetic label noise.

We use the multi-label y_i^m from CIFAR-N and the most likely correct label y_i from CIFAR-10 and derive a "hardness-score" h_i. For a sample (x_i, y_i), the score

$$h_i = \left|\left\{y_i^m \in \{y_i^1, y_i^2, y_i^3\} | y_i^m \neq y_i\right\}\right| \quad \in [0, 3] \tag{8}$$

describes, how often a sample was mislabeled in CIFAR-N and therefore relates to the difficulty. To illustrate the distribution of classes and labels, we map all images of CIFAR-10 to the latent space of the high performing diffusion model *CLIP* [12] that is using vision transformers [3] and is trained on large training data. After mapping the images to the *CLIP* latent space, we reduce the dimensions with the *t-SNE* [15] algorithm to two dimensions as shown in Fig. 3. The four sub-figures split the samples from CIFAR-10 according to their score h_i. It shows that all classes have samples with every score. Moreover, the variance of the samples per class grows with increasing hardness. As harder samples are more likely to have a larger distance to cluster centers, because they differ to the "easy" and unambiguous class templates, the increasing variance indicates that the *CLIP* latent space combined with *t-SNE* is a good tool to visualize a human-like sample distribution.

According to the well known and often discussed effect that samples with easy patterns and unambiguous labels are memorized first [1, 8, 11], samples with a higher hardness-score should be compressed later in the training process. We use the hardness-score to evaluate if this effect is also present in the process of compressing patterns from a pre-trained dense neural network into a dense sub-network using our method.

Compression Position (CP)

The next section formally defines the evaluation metric *Compression Position* as described in Sec. 4.5 in the main paper. Measurement of classification capabilities of neural networks $f(W, x)$ is usually performed by the accuracy metric

$$\psi(S, f, W) = \frac{\left|\left\{(x, y) \in S \mid f(W, x) = y\right\}\right|}{|S|}. \tag{9}$$

The accuracy of a specific class can be obtained by calculating ψ for a subset $S_c \subset S$ with only samples of a specific class $y = c$. To answer the question *"Which classes are represented first in a neural network?"*, one can measure the class accuracy after every training epoch e and plot them. To reduce the complex plot into a single metric, the area-under-curve (AUC) could be obtained for every specific class. Drawbacks from the AUC mesurements are, that the absolute values of AUC are not comparable between different settings (*i.e.*, datasets, models, ...). For example, large and complex data will lead to lower AUCs. Moreover, the class specific accuracy metric is not satisfying for the question *"Which classes are compressed first into the higher magnitude weights?"* that is addressed in this paper. We noticed, that the class accuracy of sparse networks underlie high noise rates and are therefore hard to interpret.

To tackle the drawbacks and generate a suitable metric for our work, we introduce the *Compression Position* (CP) metric that is basically a sample based accuracy over time. It aims to quantify the relative position in time between epoch e_S and e_E, where a sample x is compressed from the dense weights W into the weights with high magnitude $\nu(W)$, so that the sparse neural network is able to predict the correct ground truth label $f(\nu(W) \odot W, x) = y$.

First, we redefine Eq. (9) into a individual sample based accuracy for the pruned model that is defined as

$$\psi_{\text{I}}(x, y, f, \mathcal{W}) = \frac{\left|\left\{W_e \in \mathcal{W} \mid f(\nu(W_e) \odot W_e, x) = y\right\}\right|}{e_E - e_S} \tag{10}$$

and $\mathcal{W} = \{W_e\}_{e=e_S}^{e_E}$ denotes a set of weight sets during the training between epoch e_S and e_E. The CP metric of x is the normalized position of $\psi_{\text{I}}(x, y, f, \mathcal{W})$ in a sorted list of the sample accuracy $\Psi = \text{sort}\left(\left\{\psi_{\text{I}}(x_n, y_n, f, \mathcal{W})\right\}_{n=1}^{N}\right)$ in descending order, such that

$$\text{CP}(x, y, f, \mathcal{W}, S) : \ \psi_{\text{I}}(x, y, f, \mathcal{W}) \overset{!}{=} \Psi_{\text{CP}(x,y,f,\mathcal{W}) \cdot |\Phi|} \tag{11}$$

holds. The CP metric indicates the temporal position when a sample is compressed into the sparse weights $\nu(W) \odot W$, because CP increases if the corresponding sample is classified correct early and continuously in the training process.

Compression Behaviour

We present the main impressions of our investigations about the compression behaviour on class level in Tab. 3 and on sample level in Fig. 4 in the main paper. For the sake of completeness and to strengthen the claims, we report more detailed results in Tab. 2 and Fig. 4 and 5.

The order of compression Ψ_{sort} for a dense, two low sparsity, and three high sparsity networks is visualized in Fig. 4 and 5. Every sub-figure shows a consecutive set of 2500

samples from Ψ_{sort} and gives an intuition, which patterns are compressed into the sparse network in the beginning, middle phase and end of training. First, we observe that the diversity of classes in the first 2500 samples decreases with increasing sparsity. Second, it shows that the intra-class variance increases over time. The first observation suggests that the highest weights do not make any decisions at the beginning, or only between a few classes. In the same way that only a few classes are compressed at the beginning, the remaining classes are compressed in isolation at the end (see Fig. 5c). This is important for magnitude pruning based methods and high sparsity rates: If the highest weights have no capabilities in classification for all classes after dense training, perhaps the basic assumption that highest weights encode most important decision rules is wrong. Interestingly, our experiments consistently show, that the class *deer* tends to be compressed first and moreover, *deer* is the center in the *t-SNE* mapped latent space of *CLIP*. It seems like *deer* is the general prototype of the dataset and therefore we call the effect of preferring one class in the first compression stage *The deer bias*. The second observation reveals the main commonality between dense training and compression through regularization. Derived from the human ability to reproduce simple patterns faster, dense and sparse networks learn the general patterns first during compression and encode the high frequency samples later.

The Tab. 2 quantifies the results discussed before. It shows the compression rate for every class in CIFAR-10, subdivided by the hardness score introduced earlier. The dense networks compression rate for every class is more or less uniform-distributed. This promotes the first observation that all classes are encoded into the weights at the same time in dense networks. With increasing pruning rate κ, the classes are successively compressed into the high weights during regularization. The second observation is confirmed by dividing the classes according to their human label errors. The samples with higher label error are consistently compressed later into the high weights.

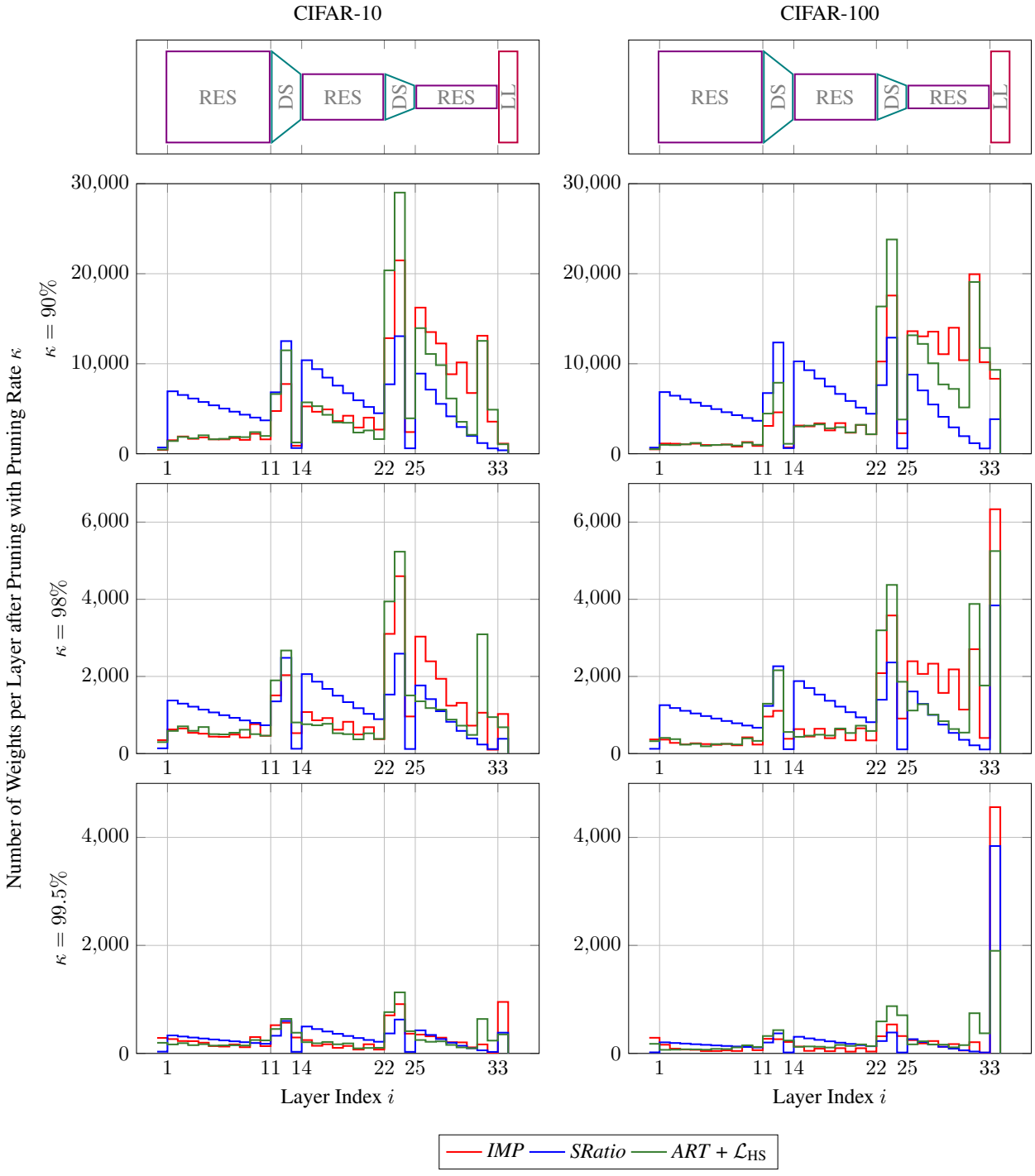


Figure 1: Distribution of weights per layer in **ResNet-32** after pruning. We visualize the results in the left column for CIFAR-10 and right column for CIFAR-100 as well as for pruning rates $\kappa \in \{90\%, 98\%, 99.5\%\}$ in each row. The layer index describes the execution order which means that higher indices are calculated later in inference. All results are averaged over 5 runs. We group the model in residual blocks (RES), downsampling blocks (DS) and the linear layer (LL). Our Method *ART* + $\mathcal{L}_{\text{HS}}$ distributes weights comparable to *IMP* [7], but has more weights in the downsampling layers. The method *SRatio* [14] has a quadratic decreasing keep-ratio. We observe the linear layer in CIFAR-100 deserves more weights compared to CIFAR-10, due to the bigger number of classes.

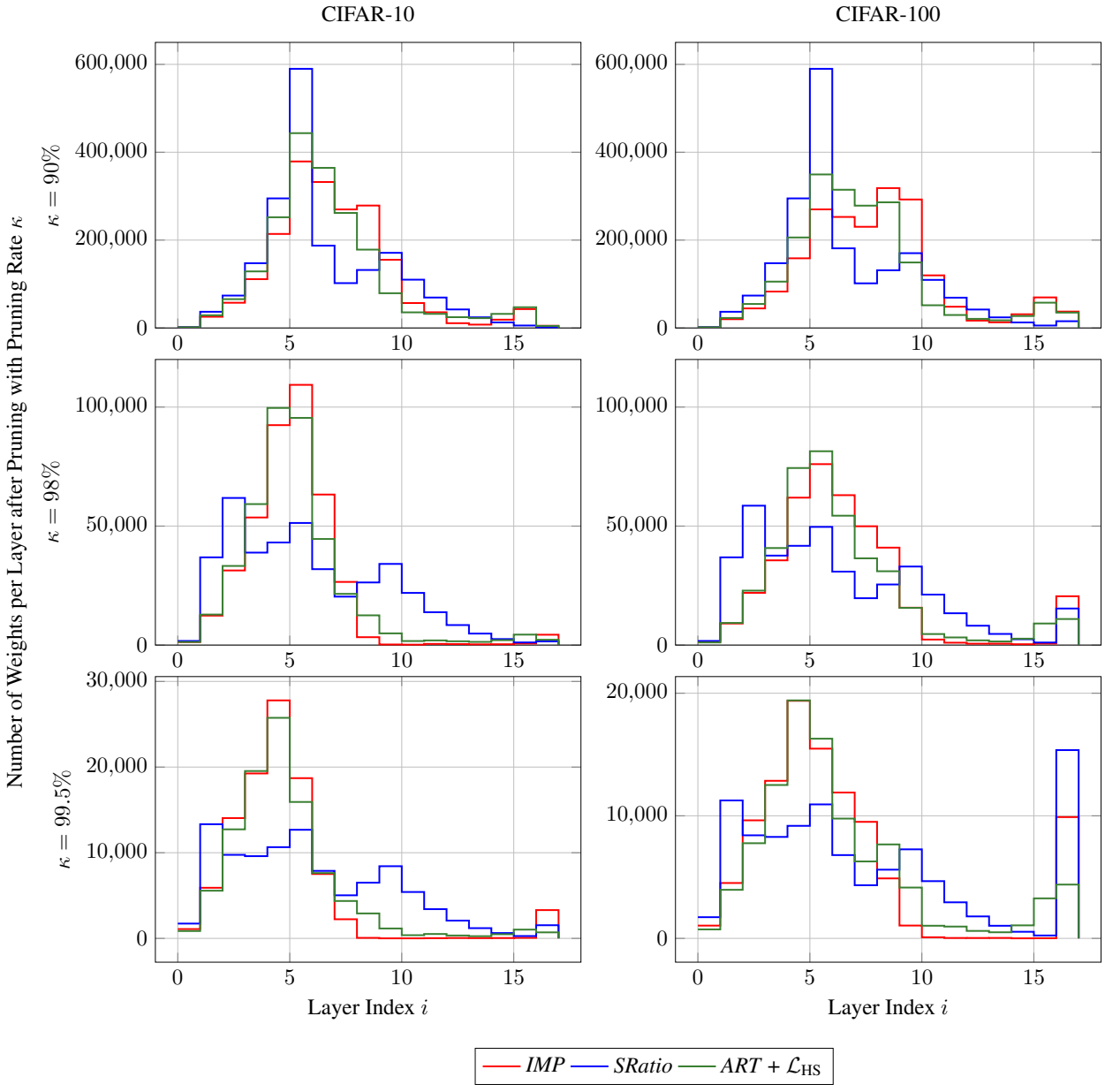


Figure 2: Distribution of weights per layer in **VGG-19** after pruning. We visualize the results in the left column for CIFAR-10 and right column for CIFAR-100 as well as for pruning rates $\kappa \in \{90\%, 98\%, 99.5\%\}$ in each row. The layer index describes the execution order which means that higher indices are calculated later in inference. All results are averaged over 5 runs. In [14], VGG-19 is constructed using multiple convolutional layers, which irregularly increase in the number of parameter over index i. The model ends with a linear layer. Our Method *ART* + $\mathcal{L}_{\text{HS}}$ distributes weights comparable to *IMP* [7], while SRatio [14] uses more weights in layers with higher index. We observe the linear layer in CIFAR-100 deserves more weights compared to CIFAR-10, due to the bigger number of classes.

CP		Human Label Errors	CIFAR-10 Class									
			airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Sparsity Level (κ)	Dense Model (0%)	0	0.452	0.295	0.554	0.639	0.482	0.551	0.425	0.429	0.357	0.400
		1	0.547	0.329	0.632	0.722	0.548	0.628	0.482	0.493	0.439	0.479
		2	0.676	0.389	0.741	0.808	0.653	0.699	0.572	0.710	0.592	0.588
		3	0.783	0.542	0.823	0.862	0.760	0.826	0.665	0.769	0.720	0.695
	Low Sparsity (90%)	0	0.580	0.612	0.499	0.494	0.166	0.460	0.257	0.524	0.470	0.567
		1	0.671	0.640	0.573	0.601	0.217	0.540	0.303	0.570	0.522	0.613
		2	0.790	0.666	0.678	0.721	0.292	0.625	0.376	0.737	0.631	0.676
		3	0.841	0.764	0.772	0.796	0.392	0.754	0.461	0.808	0.762	0.710
	Low Sparsity (95%)	0	0.739	0.685	0.604	0.696	0.187	0.486	0.400	0.173	0.223	0.543
		1	0.796	0.709	0.665	0.763	0.217	0.538	0.439	0.212	0.253	0.578
		2	0.868	0.733	0.745	0.836	0.280	0.581	0.517	0.383	0.363	0.625
		3	0.912	0.808	0.822	0.880	0.366	0.701	0.574	0.468	0.494	0.655
	Low Sparsity (98%)	0	0.742	0.635	0.579	0.640	0.045	0.504	0.349	0.471	0.313	0.452
		1	0.798	0.666	0.643	0.721	0.053	0.563	0.395	0.523	0.350	0.492
		2	0.870	0.696	0.727	0.800	0.066	0.623	0.479	0.700	0.452	0.546
		3	0.906	0.797	0.797	0.851	0.079	0.753	0.547	0.767	0.577	0.583
	High Sparsity (99%)	0	0.752	0.651	0.578	0.532	0.108	0.500	0.487	0.261	0.289	0.551
		1	0.812	0.678	0.643	0.614	0.135	0.554	0.527	0.315	0.327	0.597
		2	0.889	0.711	0.728	0.703	0.175	0.616	0.598	0.500	0.428	0.657
		3	0.919	0.820	0.800	0.775	0.234	0.736	0.647	0.593	0.539	0.709
	High Sparsity (99.5%)	0	0.761	0.691	0.472	0.441	0.049	0.571	0.364	0.758	0.246	0.429
		1	0.817	0.720	0.525	0.495	0.055	0.615	0.401	0.787	0.279	0.466
		2	0.885	0.748	0.586	0.568	0.067	0.665	0.469	0.881	0.373	0.513
		3	0.911	0.854	0.652	0.623	0.079	0.758	0.522	0.911	0.465	0.534
	High Sparsity (99.8%)	0	0.798	0.770	0.296	0.387	0.056	0.618	0.206	0.824	0.308	0.622
		1	0.835	0.787	0.317	0.411	0.061	0.647	0.214	0.835	0.331	0.647
		2	0.882	0.807	0.342	0.439	0.069	0.678	0.241	0.893	0.399	0.677
		3	0.902	0.863	0.363	0.469	0.081	0.741	0.257	0.929	0.469	0.699

Table 2: *Compression Position* (see Sec. E) for dense NNs (during pre-training) and κ pruned NNs (during regularization) for all CIFAR-10 classes. Samples of a class are split into 4 subsets according to the number of human label errors in CIFAR-N to indicate the difficulty. In sparse networks, different classes are compressed at different times and difficult samples are compressed later. These results are supplemental to Tab. 3 in the main paper.

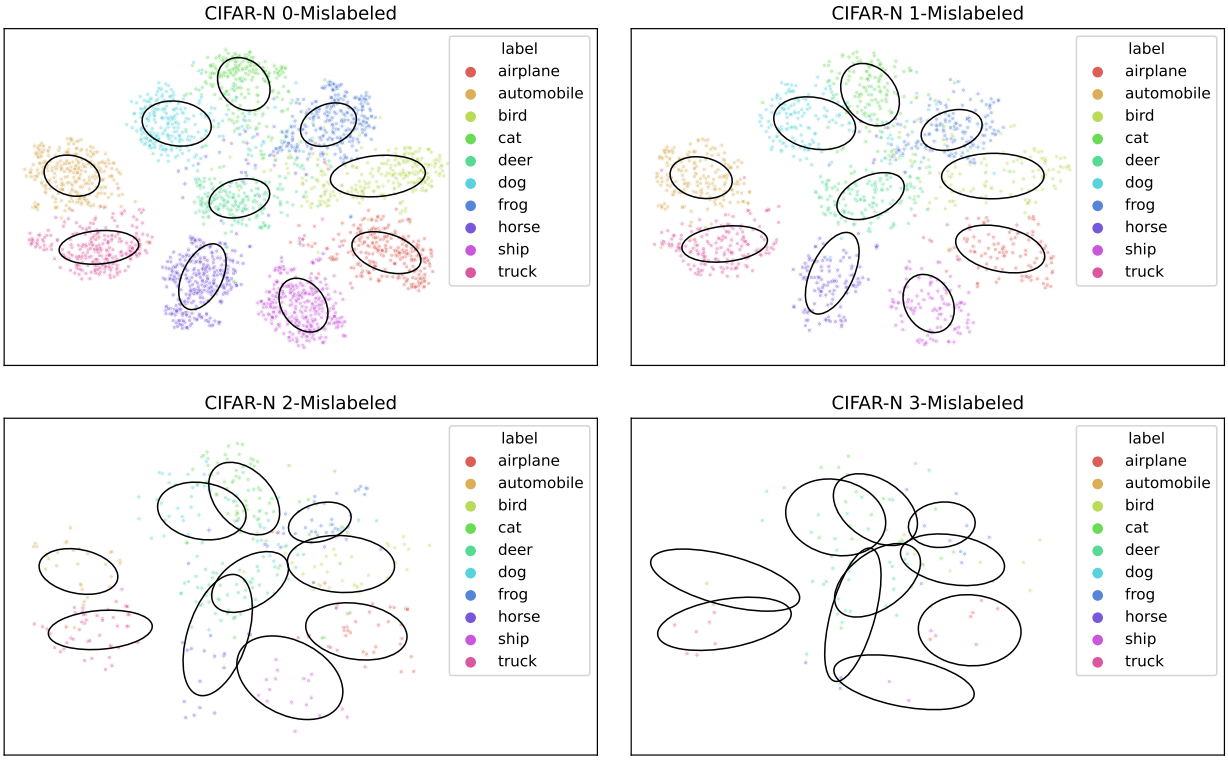
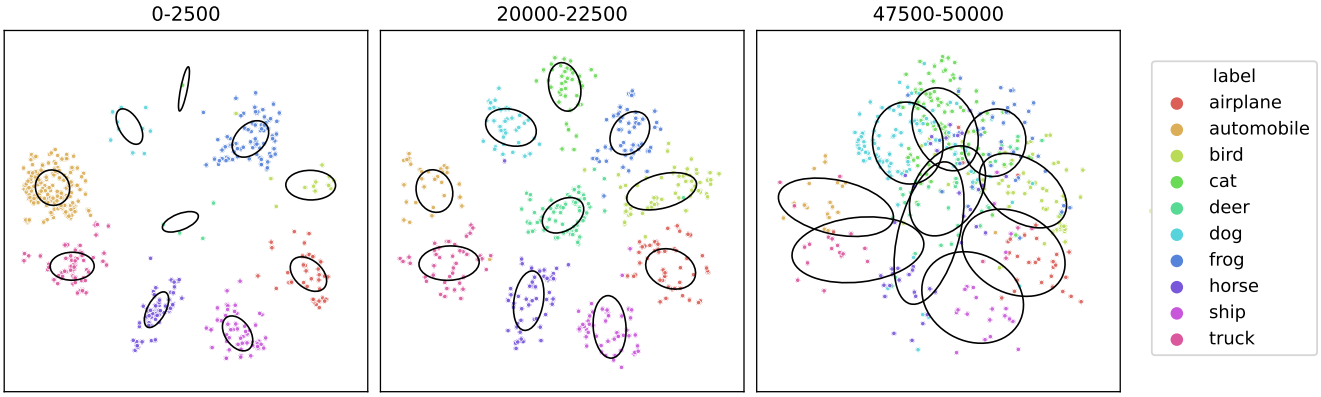
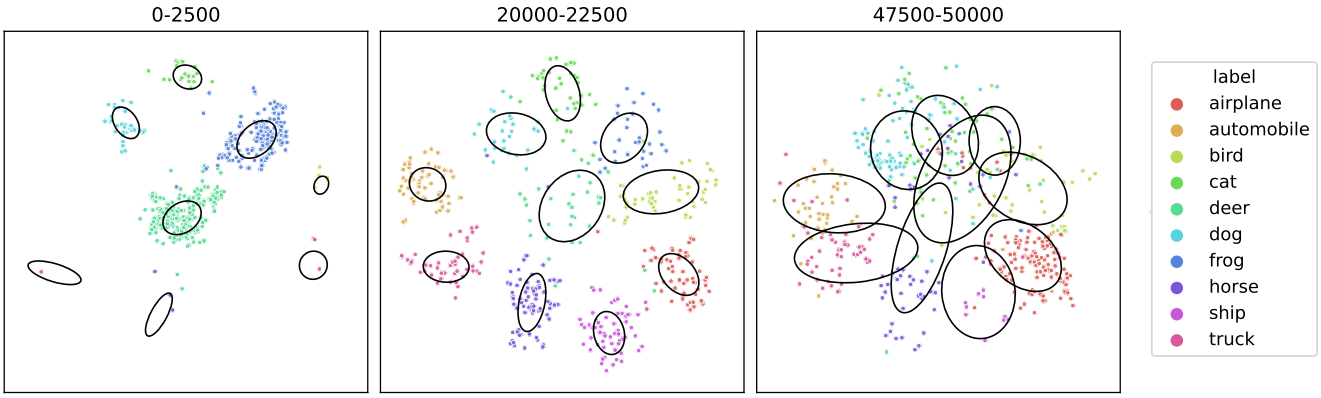


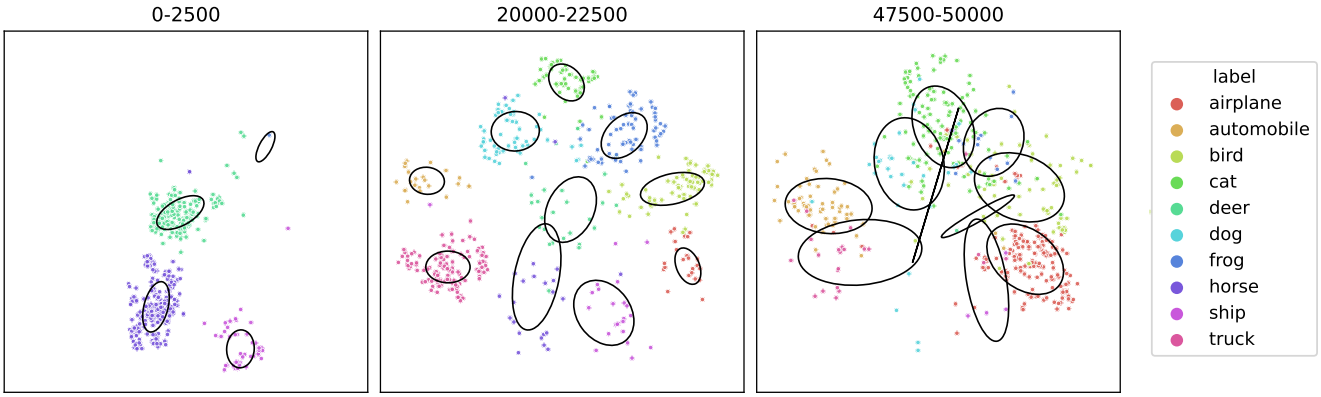
Figure 3: CIFAR-10N samples in the *CLIP* latent space, mapped by *t-SNE* into two dimensions. The datset is split into four subsets deduced by the hardness score explained in Sec. E. The term "h-Mislabeled" explains that h of 3 persons mislabeled the corresponding sample. Ellipses indicate the double standard deviation of a class in the *t-SNE* space. It shows that samples with higher hardness score h lead to a larger standard deviation and suggest that *CLIP* in combination with *t-SNE* is a suitable visualization tool to show visualize human-like recognition behaviour.



(a) Dense Model ($\kappa = 0\%$)

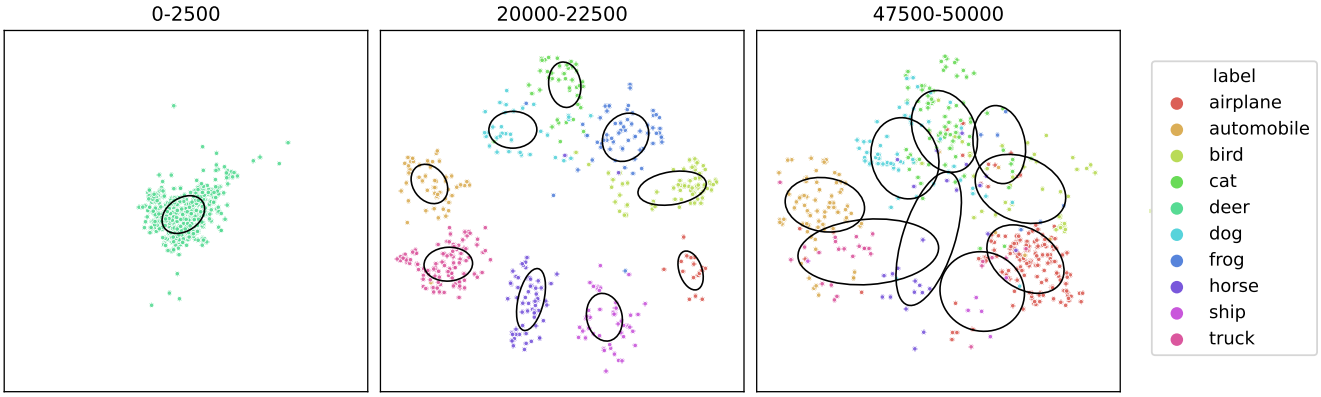


(b) Low Sparsity ($\kappa = 90\%$)

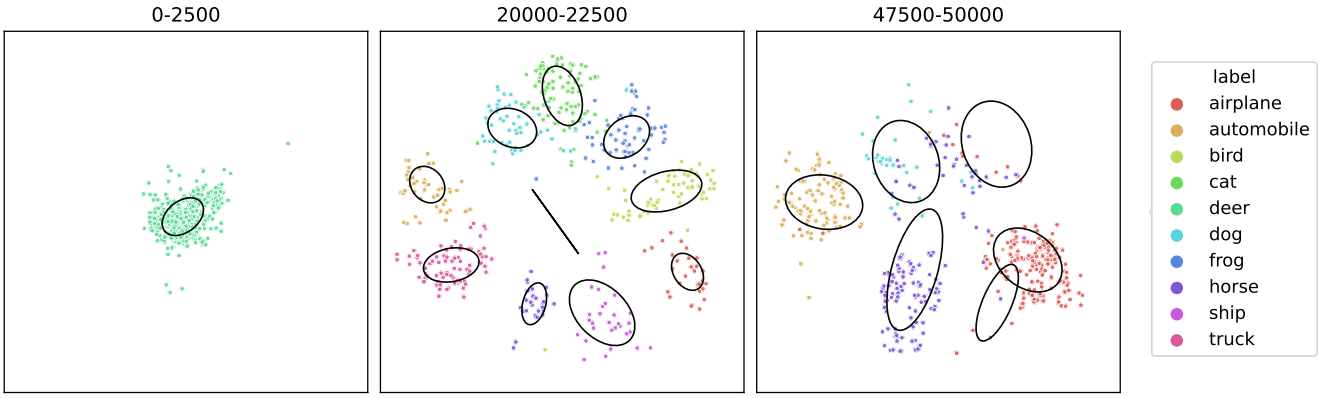


(c) Low Sparsity ($\kappa = 95\%$)

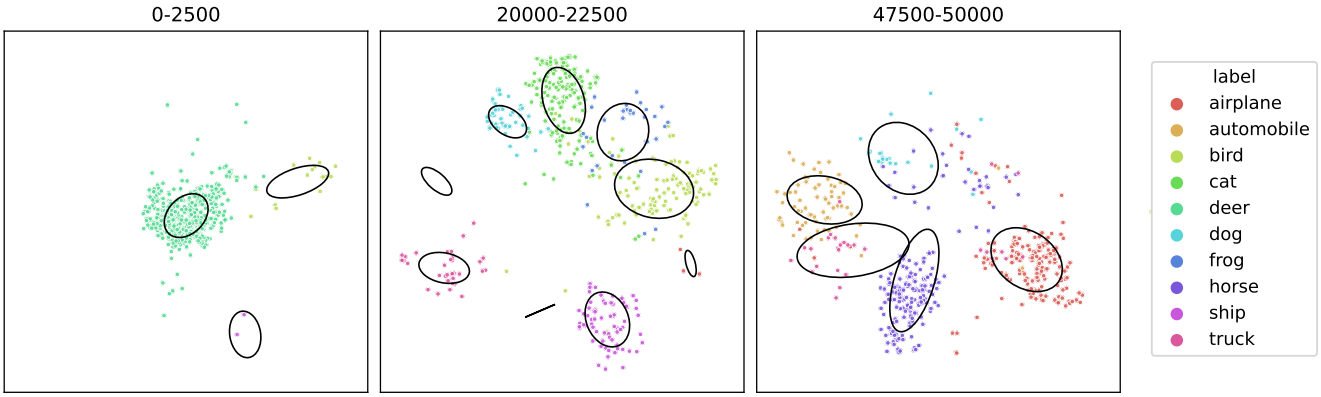
Figure 4: First 5%, 40%-45%, and 95%-100% CIFAR-10 samples that are compressed into the remaining highest weights after pruning with $\kappa \in \{0\%, 90\%, 95\%\}$ deduced by the CP-metric. While dense networks learn samples approximately uniform-distributed over classes, the highest weights compress decision rules only for a subset of classes in the early learning stage. Note that we sampled by factor 10 for visualization purposes and ellipses represent the double standard deviation of cluster centers.



(a) Low Sparsity ($\kappa = 98\%$)



(b) High Sparsity ($\kappa = 99.5\%$)



(c) High Sparsity ($\kappa = 99.8\%$)

Figure 5: First 5%, 40%-45%, and 95%-100% CIFAR-10 samples that are compressed into the remaining highest weights after pruning with $\kappa \in \{98\%, 99.5\%, 99.8\%\}$ deduced by the CP-metric. While dense networks learn samples approximately uniform-distributed over classes, the highest weights compress decision rules only for a subset of classes in the early learning stage. Note that we sampled by factor 10 for visualization purposes and ellipses represent the double standard deviation of cluster centers.

References

[1] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning (ICML)*, 2017. 4

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 1

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 4

[4] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning (ICML)*, 2020. 1

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[7] Karolina Gintare Jonathan Frankle, Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020. 1, 2, 6, 7

[8] Timo Kaiser, Lukas Ehmann, Christoph Reinders, and Bodo Rosenhahn. Blind knowledge distillation for robust image classification. In *arXiv:2211.11355*, 2022. 4

[9] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 1, 3, 4

[10] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[11] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 4

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. 3, 4

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1

[14] Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D Lee. Sanity-checking pruning methods: Random tickets can win the jackpot. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 6, 7

[15] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research (JMLR)*, 2008. 4

[16] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations (ICLR)*, 2019. 1

[17] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations (ICLR)*, 2022. 3, 4