A. Omitted Proof

A.1. Proof for Proposition 1

Proof. We first analyze the performance of f_{reg}. The prediction from this classifier is $\hat{Y}^e = \mathsf{sgn}(\boldsymbol{w}_{\mathsf{spu}}^{\mathsf{T}} Z_{\mathsf{sp}}^e)$, thus

$$\begin{aligned} \boldsymbol{w}_{\mathsf{spu}}^{\mathsf{T}} Z_{\mathsf{sp}}^e &= \frac{1}{\sqrt{D}} \sum_{i=1}^{D} Z_{\mathsf{sp},i}^e = \sqrt{D}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e\Big], \\ \hat{Y}^e &= \mathsf{sgn}(\boldsymbol{w}_{\mathsf{spu}}^{\mathsf{T}} Z_{\mathsf{sp}}^e) = \mathsf{sgn}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e\Big]. \end{aligned} \tag{2}$$

We then analyze the error for any environment:

$$\begin{aligned} \mathsf{Err}^e &= \frac{1}{2}\Big[1 - \mathbb{E}^e\big[\hat{Y}^e Y^e\big]\Big], \\ \mathbb{E}^e\big[\hat{Y}^e Y^e\big] &= \mathbb{E}^e\Big[\mathsf{sgn}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e\Big] Y^e\Big] = \sum_{y\in\{-1,1\}} \mathbb{P}[Y^e = y]\mathbb{E}^e\Big[\mathsf{sgn}\Big(\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e\Big)|Y^e = y\Big] y, \end{aligned} \tag{3}$$

$$\begin{aligned} \mathbb{E}^e\Big[\mathsf{sgn}\Big(\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e\Big)|Y^e = 1\Big] &= \mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e > 0|Y^e = 1\Big] - \mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e \le 0|Y^e = 1\Big] \\ &= 2\mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e > 0|Y^e = 1\Big] - 1. \end{aligned} \tag{4}$$

Observe that $\mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e < 0|Y^e = 1\Big] = \mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e > 0|Y^e = -1\Big]$. Using this observation and plugging equation 4 into equation 3 we get

$$\mathsf{Err}^e = \mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e \le 0|Y^e = 1\Big]. \tag{5}$$

Let us now bound Err^e. Define $\bar{Z}_{\mathsf{sp}}^e = \frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e$. Since

$$\mathbb{E}^e\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e|Y^e = 1\Big] = 2p^e - 1, \tag{6}$$

we have

$$\begin{aligned} \mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e \le 0|Y^e = 1\Big] &= \mathbb{P}\Big[\bar{Z}_{\mathsf{sp}} \le 0|Y^e = 1\Big] \\ &= \mathbb{P}\Big[\bar{Z}_{\mathsf{sp}}^e - \mathbb{E}[\bar{Z}_{\mathsf{sp}}^e] \le -\mathbb{E}[\bar{Z}_{\mathsf{sp}}^e]|Y^e = 1\Big] \le \mathbb{P}\Big[|\bar{Z}_{\mathsf{sp}}^e - \mathbb{E}[\bar{Z}_{\mathsf{sp}}^e]| \ge \mathbb{E}[\bar{Z}_{\mathsf{sp}}^e]|Y^e = 1\Big] \\ &\le 2e^{-2(2p^e-1)^2 D} \le 2e^{-2c^2 D}. \end{aligned} \tag{7}$$

In the test environment, since Z^e and Y^e are independent and $p^e = 0.5$. As a result, the error in test environment for the regular classifier is $\mathsf{Err}^e = \mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e \le 0|Y^e = 1\Big] = \mathbb{P}\Big[\frac{1}{D}\sum_{i=1}^{D} Z_{\mathsf{sp},i}^e \le 0\Big] = 0.5$.

Now let us consider the optimal sparse max-margin classifier. For $d = 2$, the max-margin classifier for the above data distribution is simply $w_{\mathsf{inv}} = 1$ and $\boldsymbol{w}_{\mathsf{sp}} = \mathbf{0}$. Since $Z_{\mathsf{inv}}^e = Y^e$, in both train and test, the sparse classifier has a perfect accuracy in both train and test environments.

Next, we compare the margins. The margin for f_{reg} is $Y^e \boldsymbol{w}_{\mathsf{sp}}^{\mathsf{T}} Z_{\mathsf{sp}} \ge \frac{c\sqrt{D}}{2}$ with a probability at least $1 - \delta$ (the proof follows directly from Hoeffding's inequality and we refer to the Appendix A of Nagarajan et al. (2020)). In comparison, f_{sparse}^d that assigns weights to invariant and spurious parts as follows $w_{\mathsf{inv}} = 1$ and $\boldsymbol{w}_{\mathsf{sp}} = \mathbf{0}$ achieves a margin of $Y^e w_{\mathsf{inv}} Z_{\mathsf{inv}} = 1$.

□

B. More Discussion

One possible algorithm we do not explore in this work is the IRM games (Ahuja et al., 2020). This approach sees the IRM formulation from a game theory perspective across different environments and design a corresponding algorithm. The algorithm has one network per environment thus the number of parameters scale in number of environments. In order to make comparisons apple to apple, we keep all the methods to have the same parameter complexity. IRM games will have more parameters so we do not compare with it. Other methods including extra parameters (*e.g.*, auxiliary neural networks) are also not considered due to analogous reason and we shall explore them in future work.

We notice there is a deep connection between Parascandolo et al. (2020) and our work. Our MRM explicitly learn a subnetwork architecture and only update the corresponding subset of weights, while Parascandolo et al. (2020) also restricts its optimization within a subset of weights by "and mask" algorithm, which only updates a parameter when gradients across domains are consensus to each other. However, this method may need a appropriate large number of domains to work (see details in their paper) and hence is not among the studied algorithms in this paper. In the future work, we intend to explore the relationship between the structure of our digit module and the subset found by their "and mask" under more domains.

Similar to our method, Dropout (Srivastava et al., 2014) also only updates part of the network during training. We list some of the main difference here: Dropout aims to prevent overfitting by simply not updating whole model parameters towards one single function, while we pursue to identify one particular functional module architecture within full network; Dropout only randomly zero out the updating gradients, while we intentionally pick particular functional part of the model in an end-to-end way; what's more, Dropout is not activated during testing inference time, while our subnetwork is kept for all following stages.

In Hooker et al. (2019; 2020), the authors propose that model compression will hurt the accuracy on underrepresented groups with negligible impacts on overall accuracy, which seems contradictory to our results. Here we state about the difference in the settings and claim that their works are actually consistent in spirit to ours. First of all, their work do not target OOD problems or invariant prediction, but focus on long tail underrepresented subgroups in IID situations. In our context "bias" means a spurious / shortcut way of inference and we aim to zero out the spurious part in the model parameters, while their settings don't contain a spurious feature that will bias the model prediction, hence most of the parameters are rightful and shouldn't be got rid of. As a result, it's natural for model compression to hurt in their cases. What's more, our method utilize a much more careful subnetwork selection method where we aim to maximize the in-domain performance when searching structures, ensuring that we do not wipe off the useful and rightful part of parameters.

In Bengio et al. (2019) and its following analysis (Priol et al., 2020), the authors claim that a better knowledge of true causal relationship will bring a faster speed of transferring learning. While on the other hand, from Figure 1 we can see a good subnetwork leads to faster convergence and better performance. Although we do not target a *causally* promising algorithm in this work (our main goal is to get rid of spurious correlation for OOD problems), we claim our approach is connected with causal discovery (Bengio et al., 2019) stated above, in the sense of capturing causal relationship hidden in data better (please note that invariant prediction can be seen as a higher level of transfer learning in an OOD sense).

C. More on Experiments

C.1. Dataset details

FULLCOLOREDMNIST. We use ten colors taken from Ahmed et al. (2021) for all the data. Their RGB values are: [0, 100, 0], [188, 143, 143], [255, 0, 0], [255, 215, 0], [0, 255, 0], [65, 105, 225], [0, 225, 225], [0, 0, 255], [255, 20, 147], [160, 160, 160]. Each image is of size $3 \times 32 \times 32$ and the whole dataset contains 60000 images. The difference between Ahmed et al. (2021) and ours are significant: their whole data is deemed to come from "majority group" and "minority group", where the images from "majority group" are colored with previous mentioned ten colors and those from "minority group" are colored with *other fifty different colors*. In contrast, all of our images are colored with these ten colors. We set a bias relationship to connect each digit and each color one by one and define bias coefficient to be the ratio of bias data that follows the relationship, as described in Section 3.1. Besides, their data are pooled and integrated as one domain before presented to algorithms. Our dataset is also different with Arjovsky et al. (2019), where they use label noise intentionally to make the correlation between label and color to be higher than the one between label and digit. This makes ERM severely biased towards color and thus fails in test domain. On the contrary, we do not impose any sort of label noise and leave label as the ground truth digit information. What's more, Arjovsky et al. (2019) use binary classification and two colors, which is a much simpler setting. Our setting is also different from Nam et al. (2020), which is not multi environment and only has

Table 4. Generalization performance on FULLCOLOREDMNIST with oracle validation.

METHODS	TRAIN ACCURACY	TEST ACCURACY
ERM	98.10 ± 0.10	58.04 ± 1.95
MRM	98.90 ± 0.05	**73.21** ± 0.58
IRM	98.17 ± 0.12	59.55 ± 1.90
MODIRM	98.67 ± 0.20	**70.35** ± 3.22
REX	98.83 ± 0.09	76.17 ± 1.53
MODREX	99.28 ± 0.05	**82.13** ± 0.82
DRO	98.94 ± 0.11	78.56 ± 1.42
MODDRO	99.38 ± 0.06	**85.67** ± 0.51
UNBIAS	99.05 ± 0.04	97.86 ± 0.20

Table 5. Generalization performance on COLOREDOBJECT with oracle validation.

METHODS	TRAIN ACCURACY	TEST ACCURACY
ERM	87.58 ± 2.42	44.39 ± 2.44
MRM	94.00 ± 0.56	**55.03** ± 1.76
IRM	87.63 ± 2.32	44.49 ± 2.15
MODIRM	92.95 ± 0.41	**52.55** ± 0.65
REX	89.28 ± 1.35	46.07 ± 2.59
MODREX	82.65 ± 1.85	**55.56** ± 3.16
DRO	91.84 ± 2.17	53.20 ± 1.15
MODDRO	92.32 ± 1.37	**55.46** ± 1.43
UNBIAS	92.32 ± 1.80	72.77 ± 3.48

one domain.

COLOREDOBJECT. We imitate Ahmed et al. (2021) to take ten objects as invariant features and put them on ten different color backgrounds. We use the ten colors mentioned above, and take the following ten objects: boat, airplane, truck, dog, zebra, horse, bird, train, bus, motorcycle. Each image is of size $3 \times 64 \times 64$ and the whole dataset contains 10000 images. The difference between ours and Ahmed et al. (2021) is similar to FULLCOLOREDMNIST: their settings contain a minority group which has many other backgrounds than the mentioned ten color backgrounds.

SCENEOBJECT. We imitate Ahmed et al. (2021) to take ten objects as invariant features and put them on ten different scenery backgrounds (beach, canyon, building facade, staircase, desert sand, crevasse, bamboo forest, broadleaf, ball pit and kasbah). We use the same ten object classes mentioned above. Each image is of size $3 \times 64 \times 64$ and the whole dataset contains 10000 images. The difference between ours and Ahmed et al. (2021) is similar to FULLCOLOREDMNIST: their settings contain a minority group which has many other backgrounds than the mentioned ten scenery backgrounds.

C.2. Experimental details

For the results in the main text, we take a commonly used policy to report the last step accuracy. We also apply the another evaluation method (the "oracle validation" in Gulrajani & Lopez-Paz (2020)) to report accuracy, and provide corresponding results in Table 4, 5 and 6. These results are consistent to those in main text, showing the validity of our conclusion. For both method, we take a similar approach to Gulrajani & Lopez-Paz (2020), and the difference is that we take a finite search set for hyperparameters instead of sampling of a human defined distribution. For all datasets, we search the regularization coefficient of IRM and REx in $\{1e-1, 5e-1, 1, 1e1, 1e2, 1e3, 1e4\}$, the step when the regularization is added into training in $\{0, 1000, 2000\}$. Furthermore, we also search a binary option about whether to scale down the whole loss term by the regularization coefficient as in Arjovsky et al. (2019). For DRO we search the group proportion step size η_q (notation taken

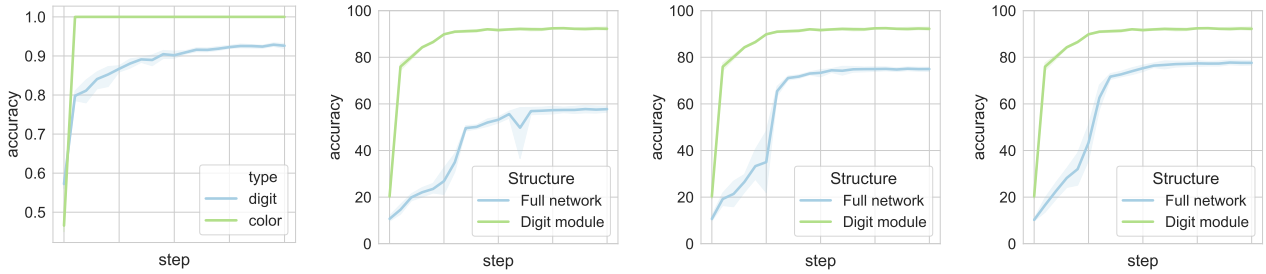
Table 6. Generalization performance on SCENEOBJECT with oracle validation.

METHODS	TRAIN ACCURACY	TEST ACCURACY
ERM	95.04 ± 2.13	37.44 ± 1.15
MRM	98.83 ± 0.26	**39.49** ± 0.15
IRM	92.66 ± 0.24	37.54 ± 0.94
MODIRM	94.63 ± 0.54	**40.04** ± 1.90
REX	92.73 ± 1.61	39.39 ± 0.96
MODREX	96.72 ± 0.53	**40.79** ± 1.62
DRO	94.53 ± 2.17	36.64 ± 1.35
MODDRO	93.52 ± 5.90	**40.99** ± 1.70
UNBIAS	85.47 ± 2.37	56.91 ± 1.31

from Sagawa et al. (2019)) in $\{1e-4, 1e-3, 1e-2, 1e-1, 1\}$. For the subnetwork structure learning, we follow Csordás et al. (2020) to use Adam optimizer and search the logit learning rate among $\{1e-2, 1e-1, 1\}$ and the sparsity coefficient among $\{1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3\}$. For Figure 4(a), the x-axis is in log scale and ranges from $1e-5$ to $1e-3$ for sparse cases.

For FULLCOLOREDMNIST, all experiments are measured by computing mean and standard deviation across five trials with random seeds. For optimization we use SGD + momentum (0.9) with $1e-4$ weight decay, where the initial learning rate is $1e-1$ and is decayed every 600 steps for all algorithms. We take the batch size to be 128. The training process longs for 2000 steps (and thus N_1 is also 2000). N_2 is set to 2000, and it can be seen from Table 7 that the importance of this hyperparameter is very limited. We use a simple ConvNet with three convolutional layers with feature map dimensions of 64, 128 and 256, each followed by a ReLU nonlinear and a batch normalization layer. The fourth layer is a fully connected layer. For COLOREDOBJECT, all experiments are measured by computing mean and standard deviation a cross three trials with random seeds. The learning rate is $1e-1$ and decays every 1200 steps. The training process longs for 2000 steps. Others are kept the same with previous dataset. We use Wide ResNet 28-2 architecture (Zagoruyko & Komodakis, 2016). To adapt to 64×64 image size, we replace the average pooling layer with window size 8 to one with size 16. For SCENEOBJECT, we use a learning rate of $5e-2$ which decays every 1200 steps. Other unmentioned settings are kept consistent as above. We use Tesla V100 GPU to perform the experiments.

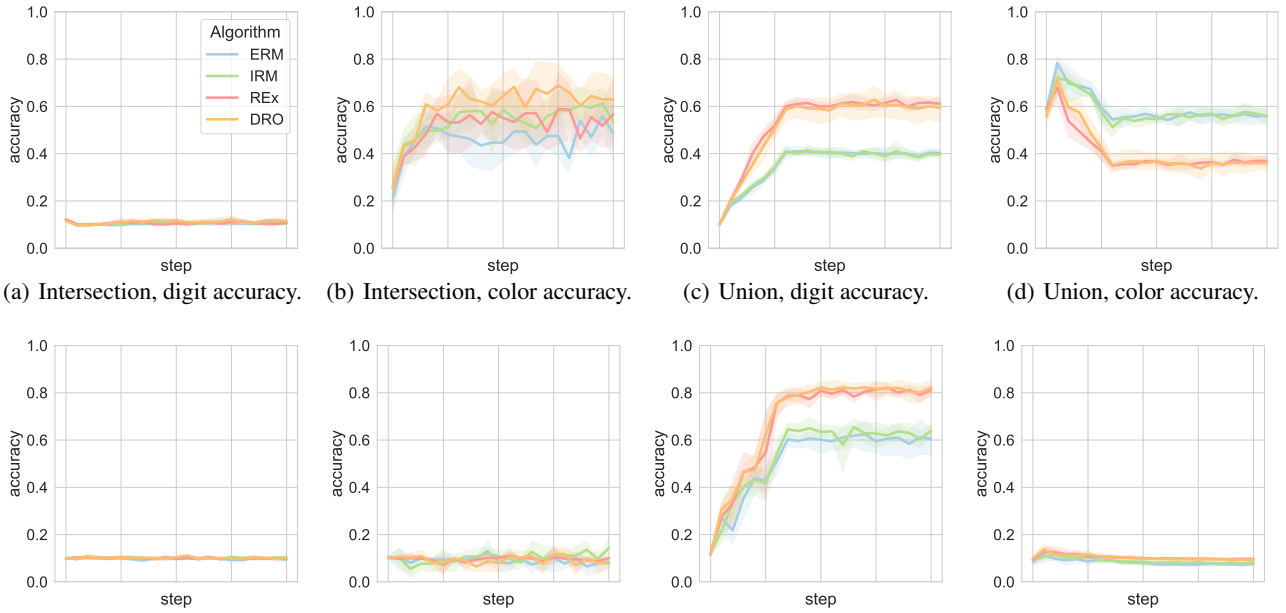
C.3. More empirical results



(a) Convergence speed for ERM. (b) Accuracy of module for IRM. (c) Accuracy of module for REx. (d) Accuracy of module for DRO.

Figure 7. (a) Convergence speed comparison for modules of ERM. (b)(c)(d) Analogy of Figure 2(b) for other algorithms.

We affirm that color is more fitted to neural network prior by plotting the digit accuracy of digit module and color accuracy of color module w.r.t. the training process in Figure 7(a). The two modules are both learned given the same ERM trained model at every step. The result shows that the training of color module converges much faster than the digit one, and thus showing that neural networks have a natural favor for color (texture) information than digit (shape) information as shown in Zeiler & Fergus (2014); Ritter et al. (2017); Brendel & Bethge (2019); Shi et al. (2020). We also plot the behaviors of digit module learning of three OOD algorithms in Figure 7(b) 7(c) and 7(d). These three plots show a very similar results



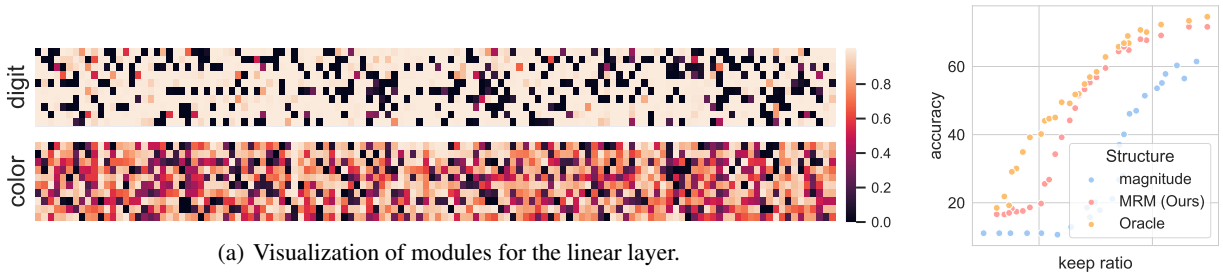
(a) Intersection, digit accuracy. (b) Intersection, color accuracy. (c) Union, digit accuracy. (d) Union, color accuracy.

(e) Complement of intersection, digit accuracy. (f) Complement of intersection, color accuracy. (g) (Complement of color module) ∪ (digit module), digit accuracy. (h) (Complement of color module) ∪ (digit module), color accuracy.

Figure 8. The digit / color accuracy for some logical operation results of learned digit and color module.

to Figure 2(b). We omit the behavior of color module of these algorithms since it's very similar to that of ERM shown in Figure 7(a).

We play with the obtained digit and color module with several basic logical operations in Figure 8. The intersection of digit and color module is important as its complement behaves trivially in Figure 8(e) and 8(f). However, the intersection module alone can not express predictiveness (Figure 8(a)) for digit identification. We show that it needs to be combined with other part to work in Figure 8(g) and 8(h). We also additionally visualize the two modules for the linear layer in Figure9(a) and get similar results to that in Figure 3.



(a) Visualization of modules for the linear layer.

Figure 9. Compensatory experimental results.

For completeness we also compare with magnitude pruning, although it's not for out-of-distribution generalization and its structure search process is data independent like us. The results for ERM on FULLCOLOREDMNIST are in Figure 9(b). We can see our methods surpass this widely used pruning approach (*e.g.*, by the original lottery ticket hypothesis paper and most prevailing approaches (Han et al., 2015; Zhu & Gupta, 2017)) in all level of sparsity. Please see more related discussion in Section C.4. Furthermore, we do ablation about the hyperparameters of subnetwork learning in the stage 2 of MRM algorithm in Table 7 to show the importance of proper level of "good" sparsity (instead of random sparsity, whose importance is restricted as discussed in main text). In Table 7 we also show results with different values of N_2 in MRM algorithm. Although N_2 we adopt is not the optimal setting, we simply use it since it's enough to achieve an approximate

Table 7. Ablation for hyperparameters of subnetwork learning. All numbers are out-distribution generalization accuracy.

LR \ α	$1e-3$	$1e-4$	$1e-5$	$1e-6$	$1e-7$
$1e-2$	50.45 ± 2.70	68.56 ± 0.39	67.46 ± 0.42	64.13 ± 1.05	63.25 ± 1.12
$1e-1$	36.95 ± 2.25	67.83 ± 0.81	72.98 ± 0.58	71.91 ± 0.85	71.30 ± 0.55
1	21.76 ±1.29	50.19 ± 2.33	65.54 ± 2.16	66.31 ± 2.56	65.51 ± 3.31

N_2	ACCURACY
1000	71.46 ± 1.89
2000	72.98 ± 0.58
3000	73.39 ± 0.84
5000	73.70 ± 0.59

Table 8. Ablation for the third stage of MRM.

METHODS	W/ STAGE 3	W/O STAGE 3
MRM	72.98 ± 0.58	62.99 ± 1.96
MODIRM	70.86 ± 2.12	58.87 ± 2.40
MODREX	82.06 ± 0.73	77.48 ± 2.30
MODDRO	85.53 ± 0.61	80.47 ± 1.87

convergence of stage 2. We further do ablation about MRM for with or without stage 3 in Table 8, showing the importance of the last stage and the validity of our functional lottery ticket hypothesis.

C.4. Towards a different pruning method

In this paper we propose an OOD algorithm coined as MRM. Here we point out that MRM can actually be seen as a different pruning method and can be applied to IID tasks as well. We do not state this explicitly in the main text since it's not closely related to our OOD research category. We slightly modify MRM to serve as a pruning algorithm in this way: we also use three stages training paradigm as MRM in main text, but specifically for stage 2, we jointly train the subnetwork structure and model parameters, and let the algorithm break the looping once the specified sparsity level is reached. We shall not claim this method to be a novel pruning method as we notice that it's similar (though stille different) to Louizos et al. (2018), which for unknown reasons is not taken into baseline consideration by recent pruning works. We now show related experimental results in the sense of pruning settings (which is IID generalization) in Table 9. The codes and baseline results are based on Wang et al. (2020). We also keep the hyperparameters setting to be consistent with Wang et al. (2020), and take the modular learning rate 0.01 and modular sparsity regularization 0.0001. Our method outperforms other baseline methods, especially for extremely sparse 98% pruning ratio cases. We think of this as an insightful advantage of our approach.

Table 9. Our pruning method on CIFAR10 with ResNet32 across different pruning ratios.

METHODS \ RATIO	90%	95%	98%
FULL NETWORK	94.23	–	–
OBD (LECUN ET AL., 1989)	94.17	93.29	90.32
MLPRUNE (ZENG & URTASUN, 2018)	94.21	93.02	90.31
LT (FRANKLE & CARBIN, 2018)	92.31	91.06	88.78
LT REWIND (FRANKLE ET AL., 2020)	93.97	92.46	89.18
DSR (MOSTAFA & WANG, 2019)	92.97	91.61	88.46
SET (MOCANU ET AL., 2018)	92.30	90.76	88.29
DEEP-R (MOCANU ET AL., 2018)	91.62	89.84	86.45
SNIP (LEE ET AL., 2018)	92.59	91.01	87.51
GRASP (WANG ET AL., 2020)	92.38	91.39	88.81
OURS	94.19	93.36	92.80