# Supplementary Material

## Contents

1	Introduction	1
2	Preliminaries	3
	2.1 Problem Formulation	3
	2.2 Pruning Targets and Settings	3
	2.3 Subnetwork Discovery in Pre-trained SSL	3
3	Method	4
	3.1 Algorithm	4
	3.2 Obtaining and Adjusting the Initial Subnetwork	4
	3.3 PARP-Progressive (PARP-P)	5
4	Experiments and Analysis	5
	4.1 Comparing PARP, OMP, and IMP on LSR, H2L, and CSR.	5
	4.2 How Important is the Initial Subnetwork (Step 1) in PARP?	7
	4.3 Are Pruning Masks Transferrable across Spoken Languages?	8
	4.4 Discovering a Single Subnetwork for 10 Spoken Languages	8
	4.5 Does PARP work on Pre-trained BERT/XLNet?	8
	4.6 Implications	9
5	Related Work	10
6	Conclusions	10
7	NeurIPS Paper Checklist	20
8	Model Details	21
	8.1 Model and Pruning Configurations	21
	8.2 Finetuning Hyper-Parameters	21
	8.3 PARP Hyper-Parameters	21
	8.4 Implementation	21
9	Experimental Setup for LSR, H2L, and CSR	22
10	How important is the IMP rewinding starting point?	22
11	OMP Masks Overlap in H2L and CSR	23

11.1 OMP Masks Overlap in H2L	23
11.2 OMP Masks Overlap in CSR	27
12 xlsr Cross-Lingual Mask Transfer	31
<b>13</b> Details of Task Transfer Results on Pre-trained BERT	32
14 Full H2L and CSR Pruning Results	34
15 wav2vec2 + PARP with Random Seeds and LM Decoding	38
16 wav2vec2 Cross-Task Mask Transfer on SUPERB	39
<b>17</b> Does Observation <b>1</b> generalize across Pre-Training Objectives?	40
<b>18 Pruned Weights Localization Across Layers</b>	41
<b>19</b> Experimental Limitations	53

## 7 NeurIPS Paper Checklist

- 1. For all authors...
  - (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Through experiments results are in Section 4. For example, our claim that PARP outperforms LTH is visible in Figure 3.
  - (b) Did you describe the limitations of your work? [Yes] Refer to Section 6 and Appendix 19.
  - (c) Did you discuss any potential negative societal impacts of your work? [No] We mention in Section 6 on the broader impact of this research work. Since this work is on pruning existing speech SSL models for low-resource spoken languages, we do not see its potential negative societal impacts. However, we welcome reviewers and AC to raise such concerns, and we will include corresponding statements.
  - (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
- 2. If you are including theoretical results...
  - (a) Did you state the full set of assumptions of all theoretical results? [N/A]
  - (b) Did you include complete proofs of all theoretical results? [N/A]
- 3. If you ran experiments...
  - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Due to its simplicity, PARP only adds a few lines of code to. Data and pre-trained models are all publicly available. These details are in the Appendix and in our project webpage: https://people.csail.mit.edu/clai24/parp/]
  - (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We follow [6, 29] for the model configurations and fine-tuning hyper-parameters. These details are in Appendix [8].
  - (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Due to the computational expense and scale of our experiments, we were not able to extensively re-run. We do note that our re-created baselines match the numbers reported in prior work [6, 29].
  - (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We briefly mention the compute needed in the footnote in Page 2, and more details are in the Appendix [8.4].
- 4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
  - (a) If your work uses existing assets, did you cite the creators? [Yes] Our work (code and pre-trained models) are based on [6, 29].
  - (b) Did you mention the license of the assets? [N/A]
  - (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
  - (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] No, we used published datasets and to the best of our knowledge, none of them have consent-related issues.
  - (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We used published datasets and, to the best of our knowledge, all of them have been reviewed carefully by the authors/community.
- 5. If you used crowdsourcing or conducted research with human subjects...
  - (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
  - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
  - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

## 8 Model Details

Model and pruning configurations for wav2vec2-base, wav2vec2-large, and xlsr can be found in Section 8.1. Fintuning hyper-parameters are generally the same as in 6, and we detailed them in Section 8.2. PARP's hyper-parameter is detailed in Section 8.3. More details on system implementations is in Section 8.4.

#### 8.1 Model and Pruning Configurations

wav2vec 2.0 consists of three modules: a 7-layer CNN feature encoder for pre-processing raw speech waveforms, a quantization layer for discretizating, and a BERT for learning contextualized representations. Given that the feature encoder is fixed and the quantization layer is discarded during finetuning, we focus on pruning the BERT module in wav2vec 2.0 and XLSR-53. We also do not prune the positional embedding layer nor the layer normalization layers within BERT. This setup is consistent with BERT-Ticket [19]. wav2vec 2.0 BASE (wav2vec2-base) is based on BERT-BASE, which has 12 transformer blocks, hidden dimension 768, 12 self-attention heads, and 95M parameters. wav2vec 2.0 LARGE (denote as wav2vec2-large) is based on BERT-LARGE, which has 24 transformer blocks, hidden dimension 768, 16 self-attention heads, and 315M parameters. XLSR-53 (denoted as xlsr) shares the same architecture as wav2vec2-large. We took wav2vec2-base and wav2vec2-large that were pre-trained on Librispeech 960h. wav2vec2-base, wav2vec2-large, and xlsr are pre-trained with the contrastive predictive coding objective.

**More on Pruning Configuration.** There are 3 components in wav2vec2/xlsr that we did not prune out: (1) CNN feature extractor, (2) layer norm running statistics, and (3) positional embedding/task-specific linear layer. For (1), it is due to the CNN feature extractor being fixed during finetuning by default, and the majority of the model parameters lie in the BERT module in wav2vec2/xlsr. For (2)(3), we simply follow the setup described in BERT-Ticket [19]. These 3 decisions is why in left of Figure [4], PARP (black line) attains ~50% PER at 100% sparsity. In fact, while re-producing BERT-Ticket [19], we were surprised that BERT's layer norm statistics plus its final linear layer achieve non trivial loss/accuracy (e.g. BERT's MLM at 0% sparsity is ~60% accuracy while at 100% sparsity is ~15% accuracy.).

#### 8.2 Finetuning Hyper-Parameters

wav2vec2 is finetuned for 20k steps on the 10h split, 15k steps on the 1h split, and 12k steps on the 10min split. xlsr is finetuned for 12k steps for each spoken languages. In the default setup in [6], wav2vec2 except the final linear layer is freezed for 10k steps, however, we observe doing so on the pruned models may lead to training instability. Therefore, we do not include this trick in our fine-tuning setups. The learning rate ramps up linearly for first 10% of the steps, remains the same for 40% of the steps, and decay exponentially for 50% of the steps. The waveform encoder output is randomly masked according to [6]. For LSR, the validation set is the dev-other subset from Librispeech.

#### 8.3 PARP Hyper-Parameters

PARP introduces an additional pruning frequency hyper-parameter, n in Algorithm Table []. As long as n is a sensible small number (e.g. 5-50 out of 10k+ steps), the final pruned models should have similar performance. We heuristically set n = 5 for pruning XLSR on all spoken language splits; we set n = 50 for wav2vec2-base on 10min/1h, n = 5 for wav2vec2-base on 10h, n = 5 for wav2vec2-large on 10min, n = 2 for wav2vec2-large on 1h, and n = 1 for wav2vec2-large on 10h.

#### 8.4 Implementation

All experiments are based on the Fairseq repository<sup>7</sup> and Wav2letter++ decoding<sup>8</sup>. We took publicly available pre-trained wav2vec2-base, wav2vec2-large, and xlsr<sup>9</sup>. The pruning code is based on

```
https://github.com/pytorch/fairseq
```

```
<sup>8</sup>https://github.com/flashlight/wav2letter
```

<sup>&</sup>lt;sup>9</sup>Pre-trained models available at <a href="https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/README.md">https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/README.md</a>

PyTorch's pruning module <sup>10</sup> For each experiment, we fine-tune the model on either 2 or 4 GPUs in parallel, and unlike the standard wav2vec 2.0 fine-tuning setup, we do not include a LM for validation during fine-tuning. Given that not all of our GPUs support FP16, our fine-tuning setup is on FP32. For fair comparison, we imposed a reasonable computational budget for all pruning methods used in this study<sup>[11]</sup>

## 9 Experimental Setup for LSR, H2L, and CSR

For LSR, we finetune pre-trained wav2vec2-base and wav2vec2-large on the 10h/1h/10min splits from Librispeech and Libri-light, as this is the *de facto* setup for studying speech representation learning [6]. For H2L, we replicate the setting described in [94, 29], where pre-trained wav2vec2-base is finetuned on 10 spoken languages (1 hour each) from CommonVoice: *Spanish (es), French (fr), Italian (it), Kyrgyz (ky), Dutch (nl), Russian (ru), Swedish (sv-SE), Turkish(tr), Tatar (tt), and Mandarin (zh-TW)*. For CSR, we replicate the setting in [29], where pre-trained xlsr is finetuned on the same 10 languages as in H2L. Studying LSR can inform us the effect of amount of finetuning supervision (10min~10h) and pre-trained model scales (base v.s. large) on pruning; on the other hand, comparing CSR and H2L could yield insights on the effect of mono-lingual versus cross-lingual pre-training on pruning.

**Evaluation Criteria.** Word Error Rate (WER) is reported for LSR; Phone Error Rate (PER) is reported for H2L and CSR<sup>12</sup> Earlier work on pruning sequence to sequence tasks, such as ASR [12] or Machine Translation [123] [41], showed that pruned models do not match or outperform the full model, albeit with "minimal degradation". Moreover, to isolate the effects of different pruning methods, we **do not** include any external LM nor any means of self-training [118] during training or decoding. To provide an unbiased grounding and accurate reflection of the pruned models, we thus report relative gains of our proposed method over OMP/IMP/MP1, in addition to their raw WER/PERs.

#### **10** How important is the IMP rewinding starting point?

We also examined the effectiveness of IMP rewinding [40, 93] for pruning speech SSL, where instead of re-starting each IMP pruning iteration all the way back from pre-trained SSL initializations, the iteration starts at some points during the downstream ASR finetuning. For example, in figure 9 IMP with 10% rewinding (dark red line) means that each pruning iteration starts at 10% into the ASR downstream finetuning; We find that rewinding has minimal effect for pruning speech SSL, which aligns with the results in NLP [19]. Curiously, we observe the effect diminishes when the pre-training model size is scaled up from base to large.



Figure 9: IMP on wav2vec2-base and wav2vec2-large with different rewinding starting point within the downstream ASR finetuning. Its effect diminishes when pruning wav2vec2-large.

<sup>&</sup>lt;sup>10</sup>https://pytorch.org/tutorials/intermediate/pruning\_tutorial.html

<sup>&</sup>lt;sup>11</sup>Each finetuning run is capped at a total of 100 V100 hours. For example, OMP requires 2 finetunings, so we will run it for at most a total of 50 hours on across 4 V100s.

<sup>&</sup>lt;sup>12</sup>WER/PER (lower the better) is standard criteria for ASR. This is opposite to previous work on pruning CV or NLP models, where accuracy or BLEU scores (higher the better) was reported.

## 11 OMP Masks Overlap in H2L and CSR

We provide the rest of Figure 2 at other sparsities to support Observation 1. For readability, we re-state it again:

For any sparsity, any amount of finetuning supervision, any pre-training model scale, and any downstream spoken languages, the non-zero ASR pruning masks obtained from taskagnostic subnetwork discovery has high IOUs with those obtained from task-aware subnetwork discovery.

In addition to IOU, we also provide the overlap percentage between masks<sup>13</sup> We divide this section into OMP masks overlap over spoken language pairs on finetuned wav2vec2-base in H2L (Section 11.1) and overlaps on finetuned xlsr in CSR (Section 11.2).

#### 11.1 OMP Masks Overlap in H2L

**H2L** OMP masks overlap procedure. Each set of experiments require  $10 \times 10$  rounds of xlsr finetunings because there are 10 downstream spoken languages ASR. The experimental procedure is:

- 1. Finetune wav2vec2-base for a source spoken language ASR.
- 2. Prune the finetuned model and obtain an OMP mask for each spoken language ASR.
- 3. Calculate IOU/mask overlap over all pairs of spoken language masks at each sparsity.







Figure 11: OMP pruning masks IOUs and overlap percentages on finetuned wav2vec2 at 20% sparsity.

<sup>&</sup>lt;sup>13</sup>Instead of taking the Union in the denominator as in IOU, simply take the full number of parameters.

	Lang	uage	OMP N	1ask I	OU at	30% S	parsit	y in w	av2ve	c 2.0	-1.00	Langua	ige Of	4P Ma	sk Ove	erlap F	ercen	tage a	it 30%	Spar	sity in	wav2ve	ec 2.0
es	1	0.94	0.932	0.93	0.926	0.924	0.942	0.934	0.928	0.922		es	1	0.981	0.979	0.978	0.977	0.976	0.982	0.98	0.978	0.976	
fr	0.94	1	0.933	0.932	0.928	0.926	0.945	0.937	0.93	0.925	- 8.85	fr	0.981	1	0.979	0.979	0.978	0.977	0.983	0.981	0.978	0.977	
it	0.932	0.933	1	0.924	0.92	0.919	0.934	0.928	0.922	0.917		it S	0.979	0.979	1	0.976	0.975	0.975	0.979	0.978	0.976	0.974	- 0.38
iasks	0.93	0.932	0.924	1	0.92	0.92	0.934	0.932	0.927	0.917	- 0.90	~	0.978	0.979	0.976	1	0.975	0.975	0.98	0.979	0.977	0.974	
MP m	0.926	0.928	0.92	0.92	1	0.915	0.931	0.924	0.919	0.914	- 5.85	MP T	0.977	0.978	0.975	0.975	1	0.973	0.979	0.976	0.975	0.973	- 036
ge Ol	0.924	0.926	0.919	0.92	0.915	1	0.927	0.923	0.918	0.912		0	0.976	0.977	0.975	0.975	0.973	1	0.977	0.976	0.974	0.972	
m .	0.942	0.945	0.934	0.934	0.931	0.927	1	0.939	0.932	0.927	- 0.90	Ū	0.982	0.983	0.979	0.98	0.979	0.977	1	0.981	0.979	0.977	
	0.934	0.937	0.928	0.932	0.924	0.923	0.939	1	0.929	0.921	- 8.75	t Lan	0.98	0.981	0.978	0.979	0.976	0.976	0.981	1	0.978	0.975	- 004
Target	0.928	0.93	0.922	0.927	0.919	0.918	0.932	0.929	1	0.916		e e	0.978	0.978	0.976	0.977	0.975	0.974	0.979	0.978	1	0.974	
	0.922	0.925	0.917	0.917	0.914	0.912	0.927	0.921	0.916	1	- 0.79		0.976	0.977	0.974	0.974	0.973	0.972	0.977	0.975	0.974	1	- 0.92
MPL	0.95	0.955	0.94	0.941	0.936	0.933	0.961	0.947	0.939	0.934	- 885	MPI	0.985	0.986	0.982	0.982	0.98	0.979	0.988	0.984	0.981	0.98	
RP	0.177	0.177	0.177	0.177	0.177	0.177	0.177	0.177	0.177	0.177		RP	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	
	es	fr	it So	urce L	angua	ge ON	sv_se 1P ma:	sks	tt	zh_TW	0.60		es	fr	it Soi	<sup>ky</sup> urce L	ni angua	ge OM	sv_se 1P ma:	sks	tt	zh_TW	530









Figure 14: OMP pruning masks IOUs and overlap percentages on finetuned wav2vec2 at 50% sparsity.

	Lang	uage	OMP N	lask l	OU at	60% S	parsit	y in w	av2ve	c 2.0	La	ngua	ige ON	4P Ma	sk Ove	erlap F	ercen	tage a	it 60%	Spar	sity in	wav2ve	ec 2.0
es	1	0.98	0.977	0.977	0.975	0.975	0.981	0.978	0.976	0.974		es	1	0.988	0.986	0.986	0.985	0.985	0.988	0.987	0.985	0.984	
fr	0.98	1	0.978	0.977	0.976	0.975	0.982	0.979	0.977	0.975	- 6.85	fr	0.988	1	0.986	0.986	0.985	0.985	0.989	0.987	0.986	0.985	
	0.977	0.978	1	0.974	0.973	0.973	0.978	0.976	0.974	0.972	V	it	0.986	0.986	1	0.984	0.984	0.983	0.987	0.985	0.984	0.983	
	0.977	0.977	0.974	1	0.973	0.973	0.978	0.977	0.976	0.972		ky	0.986	0.986	0.984	1	0.984	0.984	0.987	0.986	0.985	0.983	
OMP m	0.975	0.976	0.973	0.973	1	0.971	0.977	0.974	0.973	0.971	AP	n	0.985	0.985	0.984	0.984	1	0.983	0.986	0.984	0.983	0.982	-1
ge Ol	0.975	0.975	0.973	0.973	0.971	1	0.976	0.974	0.973	0.971	e O	ru	0.985	0.985	0.983	0.984	0.983	1	0.985	0.984	0.983	0.982	
sv_se	0.981	0.982	0.978	0.978	0.977	0.976	1	0.98	0.978	0.976	0	sv_SE	0.988	0.989	0.987	0.987	0.986	0.985	1	0.988	0.986	0.985	
<b>e</b>	0.978	0.979	0.976	0.977	0.974	0.974	0.98	1	0.976	0.974	-175 LI	tr	0.987	0.987	0.985	0.986	0.984	0.984	0.988	1	0.986	0.984	- 1
larget	0.976	0.977	0.974	0.976	0.973	0.973	0.978	0.976	1	0.972	arget	tt	0.985	0.986	0.984	0.985	0.983	0.983	0.986	0.986	1	0.983	
	0.974	0.975	0.972	0.972	0.971	0.971	0.976	0.974	0.972	1	to	zh_TW	0.984	0.985	0.983	0.983	0.982	0.982	0.985	0.984	0.983	1	- 1
MPL	0.984	0.985	0.98	0.98	0.979	0.978	0.987	0.983	0.98	0.978	- 845	MPI	0.99	0.991	0.988	0.988	0.987	0.987	0.992	0.989	0.988	0.987	
RP	0.429	0.429	0.429	0.429	0.429	0.429	0.429	0.429	0.429	0.429		RP	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	
	es	fr	it So	<sup>ky</sup> urce L	ni angua	ge ON	<sup>sv_SE</sup> 1P ma:	sks	tt	zh_TW	0.60		es	fr	it Soi	<sup>ky</sup> urce L	angua	ru ige OM	sv_se 1P ma:	sks	tt	zh_TW	









Figure 17: OMP pruning masks IOUs and overlap percentages on finetuned wav2vec2 at 80% sparsity.

	Lang	uage						·			1.00 L	angua	ige Of					-				wav2ve	ec 2
es	1	0.996	0.995	0.995	0.995	0.995	0.996	0.996	0.995	0.995		es	1	0.996	0.996	0.996	0.995	0.995	0.997	0.996	0.996	0.995	
fr	0.996	1	0.996	0.996	0.995	0.995	0.996	0.996	0.995	0.995	- 835	fr	0.996	1	0.996	0.996	0.996	0.996	0.997	0.996	0.996	0.996	
it	0.995	0.996	1	0.995	0.995	0.995	0.996	0.995	0.995	0.995	- cao 🗸		0.996	0.996	1	0.995	0.995	0.995	0.996	0.996	0.995	0.995	
ky ky	0.995	0.996	0.995	1	0.995	0.995	0.996	0.995	0.995	0.995	mask	ky	0.996	0.996	0.995	1	0.995	0.995	0.996	0.996	0.996	0.995	
l nl	0.995	0.995	0.995	0.995	1	0.994	0.995	0.995	0.995	0.994	- III OMP	nl -	0.995	0.996	0.995	0.995	1	0.995	0.996	0.995	0.995	0.995	
	0.995	0.995	0.995	0.995	0.994	1	0.995	0.995	0.995	0.994	de O	ru -	0.995	0.996	0.995	0.995	0.995	1	0.996	0.995	0.995	0.995	
sv_SE	0.996	0.996	0.996	0.996	0.995	0.995	1	0.996	0.996	0.995	guad	sv SE	0.997	0.997	0.996	0.996	0.996	0.996	1	0.996	0.996	0.996	
j tr	0.996	0.996	0.995	0.995	0.995	0.995	0.996	1	0.995	0.995		tr	0.996	0.996	0.996	0.996	0.995	0.995	0.996	1	0.996	0.995	
n tt	0.995	0.995	0.995	0.995	0.995	0.995	0.996	0.995	1	0.994	arde .	h tt	0.996	0.996	0.995	0.996	0.995	0.995	0.996	0.996	1	0.995	
zh_TW	0.995	0.995	0.995	0.995	0.994	0.994	0.995	0.995	0.994	1	- 0.00 FC	zh_TW	0.995	0.996	0.995	0.995	0.995	0.995	0.996	0.995	0.995	1	
MPL	0.997	0.997	0.996	0.996	0.996	0.996	0.997	0.997	0.996	0.996	- 845	MPI	0.997	0.997	0.996	0.997	0.996	0.996	0.998	0.997	0.996	0.996	
RP	0.818	0.818	0.818	0.818	0.818	0.818	0.818	0.818	0.818	0.818		RP	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	
	es	fr	it So	<sup>ky</sup> urce L	ni angua	ru ige ON	sv_se 1P ma:	sks	tt	zh_TW	0.60		es	fr	it So	<sup>ky</sup> urce L	ni angua	ru ige ON	<sup>sv_SE</sup> 1P ma:	tr sks	tt	zh_TW	

Figure 18: OMP pruning masks IOUs and overlap percentages on finetuned wav2vec2 at 90% sparsity.

#### 11.2 OMP Masks Overlap in CSR

**CSR** OMP masks overlap procedure. Each set of experiments require  $10 \times 10$  rounds of xlsr finetunings because there are 10 downstream spoken languages ASR. The experimental procedure is:

- 1. Finetune xlsr for a source spoken language ASR.
- 2. Prune the finetuned model and obtain an OMP mask for each spoken language ASR.
- 3. Calculate IOU/mask overlap over all pairs of spoken language masks at each sparsity.







Figure 20: OMP pruning masks IOUs and overlap percentages on finetuned xlsr at 20% sparsity.







Figure 22: OMP pruning masks IOUs and overlap percentages on finetuned xlsr at 40% sparsity.



Figure 23: OMP pruning masks IOUs and overlap percentages on finetuned xlsr at 50% sparsity.

	La	nguag	e OMF	9 Mask	lou a	at 60%	Spar	sity in	XLSR-	53	Lar	ιgι	iage (	OMP N	1ask C	verlap	Perce	entage	e at 60	)% Sp	arsity	in XLSR	-53	-10
es	1	0.959	0.956	0.955	0.96	0.954	0.952	0.959	0.957	0.959		es ·	1	0.975	0.973	0.972	0.975	0.972	0.97	0.975	0.974	0.975		
fr	0.959	1	0.953	0.953	0.958	0.952	0.95	0.957	0.955	0.956	- 2.95	fr	0.975	1	0.971	0.971	0.974	0.97	0.969	0.974	0.972	0.973		
	0.956	0.953	1	0.95	0.954	0.949	0.947	0.954	0.952	0.953	-120 V	it	0.973	0.971	1	0.969	0.972	0.969	0.967	0.972	0.97	0.971		- 0
masks	0.955	0.953	0.95	1	0.954	0.95	0.947	0.955	0.953	0.953	×	ky	0.972	0.971	0.969	1	0.972	0.969	0.967	0.972	0.971	0.971		
UMP II	0.96	0.958	0.954	0.954	1	0.953	0.951	0.958	0.956	0.957		nl÷	0.975	0.974	0.972	0.972	1	0.971	0.97	0.974	0.973	0.974		
ge Ol	0.954	0.952	0.949	0.95	0.953	1	0.946	0.953	0.951	0.952	<u>Ψ</u>	nu -	0.972	0.97	0.969	0.969	0.971	1	0.967	0.971	0.97	0.97		
sv_se	0.952	0.95	0.947	0.947	0.951	0.946	1	0.951	0.949	0.95	oen 5v_s	se	0.97	0.969	0.967	0.967	0.97	0.967	1	0.97	0.968	0.969		
e t	0.959	0.957	0.954	0.955	0.958	0.953	0.951	1	0.957	0.957		tr	0.975	0.974	0.972	0.972	0.974	0.971	0.97	1	0.973	0.974		
Target	0.957	0.955	0.952	0.953	0.956	0.951	0.949	0.957	1	0.955	2	tt	0.974	0.972	0.97	0.971	0.973	0.97	0.968	0.973	1	0.972		
	0.959	0.956	0.953	0.953	0.957	0.952	0.95	0.957	0.955	1	-0.00 PD zh_T	w	0.975	0.973	0.971	0.971	0.974	0.97	0.969	0.974	0.972	1		
MPL	0.971	0.967	0.962	0.962	0.968	0.961	0.958	0.968	0.965	0.969	- 885 M	PI -	0.982	0.98	0.977	0.977	0.981	0.976	0.974	0.981	0.979	0.981		I
RP	0.429	0.429	0.429	0.429	0.429	0.429	0.429	0.429	0.429	0.429		RΡ	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52		I
	es	fr	it So	urce L	angua	ge ON	<sup>sv_SE</sup> 1P ma:	sks	ŧ	zh_TW	6.60	1	es	fr	it Soi	urce L	ni angua	ge OM	sv_se 1P ma:	sks	tt	zh_TW		









Figure 26: OMP pruning masks IOUs and overlap percentages on finetuned xlsr at 80% sparsity.

	La	nguag	e OMF	9 Masł	i IOU a	at 90%	Spar	sity in	XLSR	53	Lang	luage	OMP N	1ask C	verla	o Perc	entage	e at 90	)% Sp	arsity	in XLSI	R-53	3
es -	1	0.991	0.99	0.99	0.991	0.99	0.989	0.991	0.991	0.991	es	- 1	0.992	0.991	0.991	0.992	0.991	0.99	0.992	0.991	0.992		
fr	0.991	1	0.99	0.99	0.991	0.989	0.989	0.99	0.99	0.99	-05 fr	0.992	1	0.991	0.991	0.991	0.99	0.99	0.991	0.991	0.991		
it	0.99	0.99	1	0.989	0.99	0.989	0.988	0.99	0.989	0.99	it V	0.991	0.991	1	0.99	0.991	0.99	0.989	0.991	0.99	0.991		- 0
masks <sup>ky</sup>	0.99	0.99	0.989	1	0.99	0.989	0.988	0.99	0.99	0.99	w system	0.991	0.991	0.99	1	0.991	0.99	0.989	0.991	0.991	0.991		
ul ni-	0.991	0.991	0.99	0.99	1	0.99	0.989	0.991	0.99	0.991	-u dw ni OWb u	0.992	0.991	0.991	0.991	1	0.991	0.99	0.992	0.991	0.992		- 6
	0.99	0.989	0.989	0.989	0.99	1	0.988	0.99	0.989	0.989	u de Ol	0.991	0.99	0.99	0.99	0.991	1	0.989	0.991	0.99	0.99		
ebenb sv_se	0.989	0.989	0.988	0.988	0.989	0.988	1	0.989	0.989	0.989	n sv_SE	0.99	0.99	0.989	0.989	0.99	0.989	1	0.99	0.99	0.99		
	0.991	0.99	0.99	0.99	0.991	0.99	0.989	1	0.99	0.991	t Lan	0.992	0.991	0.991	0.991	0.992	0.991	0.99	1	0.991	0.992		
n get	0.991	0.99	0.989	0.99	0.99	0.989	0.989	0.99	1	0.99	n Ige	0.991	0.991	0.99	0.991	0.991	0.99	0.99	0.991	1	0.991		
	0.991	0.99	0.99	0.99	0.991	0.989	0.989	0.991	0.99	1	-∞≫ <u>P</u> zh_TW	0.992	0.991	0.991	0.991	0.992	0.99	0.99	0.992	0.991	1		
MPL	0.994	0.993	0.992	0.992	0.993	0.991	0.991	0.993	0.992	0.993	- eas MPI	0.994	0.993	0.993	0.993	0.994	0.992	0.992	0.994	0.993	0.994		
RP	0.818	0.818	0.818	0.818	0.818	0.818	0.818	0.818	0.818	0.818	RP	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82		
	es	fr	it So	<sup>ky</sup> urce L	angua	ge ON	sv_se 1P ma:	sks	ŧ	zh_TW	0.60	es	fr	it So	<sup>ky</sup> urce L	angua	ge ON	sv_se 1P ma:	sks	tt	zh_TW		_

Figure 27: OMP pruning masks IOUs and overlap percentages on finetuned xlsr at 90% sparsity.

## 12 xlsr Cross-Lingual Mask Transfer

**Cross-lingual mask transfer procedure.** Each set of experiments require  $10 \times 10 \times 2$  rounds of xlsr finetunings because there are 10 downstream spoken languages ASR, and we finetune for each spoken language ASR twice (the first one for retrieving mask, and second one for mask transfer). The experimental procedure is:

- 1. Finetune xlsr/wav2vec2 for a source spoken language ASR.
- 2. Prune the finetuned model and obtain an OMP mask for each spoken language ASR.
- 3. Apply the OMP mask at xlsr pre-trained initializations and finetune for a target spoken language ASR with PARP.

Figure 28 is the result, and it has the same cross-lingual mask transfer setup as that in Section 4.3 and Figure 7, except the pre-trained model is xlsr instead of wav2vec2.

			-,	Langa	<u>age i i</u>			opurs			
(%	es (80%) -	0	0.073	-0.497	-0.474	-0.613	-1.465	-0.083	-0.762	0.023	-1.925
(Sparsity	fr (80%) -	-0.688	0	-0.115	-0.366	-0.55	-1.503	-0.3	-0.765	-0.037	-1.584
	it (80%) -	-0.749	-0.626	0	-0.301	-0.269	-1.377	0.427	-1.218	0.03	-2.153
language	ky (80%) -	3.079	-0.099	-0.315	0	-0.518	-1.827	-0.267	-1.974	-0.688	-1.618
source	nl (80%) -	2.107	-0.565	-0.472	-0.798	0	-1.465	-0.686	-0.67	-0.29	-2.037
trom S	ru (80%) -	0.741	0.147	-0.335	-0.877	-0.357	0	-0.181	-1.298	-0.227	-1.514
0 0	sv_SE (80%) -	1.439	-0.295	-0.351	-0.615	-0.591	-1.256	0	-1.473	-0.224	-1.812
	tr (80%) -	0.476	-0.565	-0.06	-1.178	-0.597	-1.358	-0.616	0	-0.537	-2.403
Subnetworks	tt (80%) -	1.808	-0.381	-0.428	-1.42	-0.54	-1.492	-0.744	-1.69	0	-2.015
sut ĭ	h_TW (80%) -	-0.549	-0.737	-0.08	-0.464	-0.338	-1.053	-0.23	-1.199	-0.134	0
		es	fr	it Finetur	he subr	network	տ s on Ta	<sup>sv_SE</sup> raet lar	nguage	ť	zh_TW

Transferrability of Language Masks at 80% Sparsity in XLSR-53 with PARP

- 4

3

- 2

- 1

- 0

Figure 28: Cross-lingual mask transfer for xlsr. Cross-lingual mask transfer with PARP has minimal PER degradation (darker the better).

## 13 Details of Task Transfer Results on Pre-trained BERT

**Cross-task mask transfer procedure.** Each set of experiments require  $9 \times 9 \times 2$  rounds of finetunings because there are 9 subtasks in GLUE, and we finetune for each subtask twice (the first one for retrieving mask, and second one for mask transfer). We first note that our cross-task transfer experimental designs are closely knitted to NLP probing work's experimental setup [113, 36], i.e. pretrained BERT/XLNet on 9 subtasks in GLUE. The experimental procedure is:

- 1. Finetune BERT/XLNet for a source task in GLUE.
- 2. Prune the finetuned model and obtain an IMP mask for each task.
- 3. Apply the IMP mask at BERT/XLNet pre-trained initializations and finetune for a target task in GLUE with PARP.

Figure 29 is the IMP mask overlap for pre-trained BERT on the 9 natural language tasks in GLUE. Figure 30 is the cross-task transfer result. For all the GLUE tasks, PARP can achieve better results compared to BERT-Ticket (cross-task subnetwork regular finetuning) [19]. For the tasks with poor transferability in BERT-Ticket [19], like CoLA and STS-B, PARP still achieves good transfer scores.

		11-11	MUSK	100 01	OLOL		C / O / O	Spursh	Ly III DI	_1\1	- 1.00
(	CoLA -	1	0.9943	0.9849	0.9737	0.9949	0.9876	0.9938	0.9953	0.9722	
М	1RPC -	0.9943	1	0.9851	0.9738	0.9957	0.9878	0.9946	0.996	0.9724	- 0.95
	QNLI -	0.9849	0.9851	1	0.9709	0.9853	0.9816	0.985	0.9854	0.9697	- 0.90
⊢	QQP -	0.9737	0.9738	0.9709	1	0.974	0.9718	0.9738	0.974	0.9632	- 0.85
GLUE IMP	RTE -	0.9949	0.9957	0.9853	0.974	1	0.9881	0.9952	0.9972	0.9725	- 0.80
Ļ	ST-2 -	0.9876	0.9878	0.9816	0.9718	0.9881	1	0.9876	0.9882	0.9706	- 0.75
	TS-B -	0.9938	0.9946	0.985	0.9738	0.9952	0.9876	1	0.9954	0.9723	- 0.70
V	WNLI -	0.9953	0.996	0.9854	0.974	0.9972	0.9882	0.9954	1	0.9725	- 0.65
,	MNLI -	0.9722	0.9724	0.9697	0.9632	0.9725	0.9706	0.9723	0.9725	1	
		CoLA	MRPC	QNLI	<sub>oop</sub> Source (	GLUE IM	<sub>sst-2</sub> P masks	STS-В	WNLI	MŃLI	- 0.60

IMP Mask IOU of GLUE tasks at 70% Sparsity in BERT

Figure 29: IOUs over all GLUE tasks' IMP pruning masks on finetuned BERT at 70% sparsity. Notice the high overlap rates, which aligns with Observation 1.



Transferrability of GLUE Masks at 70% Sparsity in BERT



Figure 30: Results for subnetwork transfer experiment (take subnetwork found by IMP at task A and finetune it for task B). Top: the transfer results in BERT-Ticket [19]. Bottom: transfer with PARP finetuning instead. Each row is a source task A, and each column is a target task B. All numbers are subtracted by the scores of same-task transfer (task A = task B, and the darker the better).

## 14 Full H2L and CSR Pruning Results

We provide the full set of H2L and CSR pruning (refer to Section 4.1 and Section 4.4 for experimental description). Below are the rest of Figure 4 to other spoken languages from CommonVoice: *Spanish* (es), French (fr), Italian (it), Kyrgyz (ky), Dutch (nl), Russian (ru), Swedish (sv-SE), Turkish(tr), Tatar (tt), and Mandarin (zh-TW)



Figure 31: Comparison of pruning techniques on H2L & CSR with 1h of Spanish (*es*) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + *es*). (Center) Pruning CSR (xlsr + *es*). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on *es*.



Figure 32: Comparison of pruning techniques on H2L & CSR with 1h of French (fr) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + fr). (Center) Pruning CSR (xlsr + fr). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on fr.



Figure 33: Comparison of pruning techniques on H2L & CSR with 1h of Italian (*it*) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + *it*). (Center) Pruning CSR (xlsr + *it*). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on *it*.



Figure 34: Comparison of pruning techniques on H2L & CSR with 1h of Kyrgyz (ky) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + ky). (Center) Pruning CSR (xlsr + ky). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on ky.



Figure 35: Comparison of pruning techniques on H2L & CSR with 1h of Dutch (nl) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + nl). (Center) Pruning CSR (xlsr + nl). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on nl.



Figure 36: Comparison of pruning techniques on H2L & CSR with 1h of Russian (ru) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + ru). (Center) Pruning CSR (xlsr + ru). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on ru.



Figure 37: Comparison of pruning techniques on H2L & CSR with 1h of Swedish (sv-SE) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + sv-SE). (Center) Pruning CSR (xlsr + sv-SE). (**Right**) Pruning jointly-finetuned wav2vec2-base and xlsr on sv-SE.



Figure 38: Comparison of pruning techniques on H2L & CSR with 1h of Turkish (tr) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + tr). (Center) Pruning CSR (xlsr + tr). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on tr.



Figure 39: Comparison of pruning techniques on H2L & CSR with 1h of Tatar (tt) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + tt). (Center) Pruning CSR (xlsr + tt). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on tt.



Figure 40: Comparison of pruning techniques on H2L & CSR with 1h of Mandarin (*zh-TW*) ASR finetuning. (Left) Pruning H2L (wav2vec2-base + *zh-TW*). (Center) Pruning CSR (xlsr + *zh-TW*). (Right) Pruning jointly-finetuned wav2vec2-base and xlsr on *zh-TW*.

#### 15 wav2vec2 + PARP with Random Seeds and LM Decoding

We re-iterate the two reasons why we did not including LM decoding in our main results. First, we isolate the effect of pruning on ASR. Note that the standard LM (either 4-gram/transformer) used in the wav2vec series are also trained on Librispeech (text-corpus) [6]. [4]. Therefore, the LMs can easily recover errors made by the acoustic model. Secondly, note that the 4-gram/transformer LM decoding hyper-parameters are carefully searched via Bayesian optimization<sup>14</sup> in the wav2vec series. Such optimization procedure would be quite expensive to run for **just one** model, let alone thousands of pruned models produced in this work.

We provide two sets of results to validate our claim that applying PARP on wav2vec2 reduces the downstream ASR:

- The first result is the impact of random seeds. We finetune wav2vec2-base with 10min data at 10% sparsity with PARP at 8 additional seeds. Table 5 is the result with Viterbi decoding without LM. We can see that at different seed values, pruned wav2vec2-base all converged to similar WERs, which is ~10% WER reductions compared to a full wav2vec2-base.
- The second result is pruned wav2vec2-base with the official 4-gram/transformer LM decoding. The pruned wav2vec-base is finetuned on the 10min Librispeech split and pruned at 10% sparsity with PARP. Since we do not have the compute resource to replicate 1500 beam 4-gram decoding and 500 beam transformer-LM decoding used in the original paper [6], this experiment is based on a more moderate beam size. Similar to [6], decoding hyper-parameters are searched via Ax on the dev-other Librispeech subset over 128 trials. As shown in Table [6], the performance gain over the full wav2vec2-base reduces with LM decoding, but we still observe a performance improvement at 10% sparsity with PARP.

Table 5: Pruning wav2vec-base	with PARP at different trainnig seeds	. Setting is on Librispeech
10min without LM decoding.		

Method	seed	test-clean/test-other
Full wav2vec2-base	2447	49.3/53.2
wav2vec2-base + 10% PARP	2447 0 1 2 3 5	38.04/44.33 37.01/43.02 37.82/43.66 37.59/43.55 37.57/43.29 37.48/44.10
	6 7 8	37.87/43.55 37.65/43.53 38.22/43.91

Table 6: Decode pruned wav2vec-base with official 4-gram/transformer LMs. Setting is on Librispeech 10min.

Method	decoding algorithm	beam size	test-clean/test-other
Full wav2vec2-base	viterbi (no LM) 4-gram LM	5	49.3/53.2 27.82/32.02
	transformer LM	5	27.16/32.68
	viterbi (no LM)		37.69/43.66
wav2vec2-base + 10% PARP averaged	4-gram LM	5	25.17/32.13
	transformer LM	5	25.45/32.46

<sup>&</sup>lt;sup>14</sup>https://github.com/facebook/Ax

## 16 wav2vec2 Cross-Task Mask Transfer on SUPERB

We extend experiments in Section 4.3 to downstream tasks other than ASR, i.e. extend the transferability of pruning masks across speech tasks. We selected three drastically different target tasks from SUPERB [120]: Phone Recognition with 10h Librispeech data (in PER), Automatic Speaker Verification on VoxCeleb (in EER), and Slot Filling on audio SNIPS (in slot type  $F_1$ /slot value CER). PER/EER/CER are lower the better, and  $F_1$  is higher the better. The experiment procedure <sup>15</sup> is as follows:

- 1. Finetune wav2vec2 for a source task in SUPERB.
- 2. Prune the finetuned model and obtain an OMP mask for each task.
- 3. Apply the OMP mask at wav2vec2 pre-trained initializations and finetune for a target task in SUPERB with PARP.

Table 7 is the wav2vec-base cross-task transfer result in SUPERB. We did learning rate grid search over  $\{1.0 \times 10^{-3}, 1.0 \times 10^{-4}, 1.0 \times 10^{-5}, 2.0 \times 10^{-5}, 3.0 \times 10^{-5}, 1.0 \times 10^{-6}, 1.0 \times 10^{-7}\}$ , and presented the best number. Note that different from SUPERB's default setup, we make the upstream wav2vec2 jointly finetunable for PARP. Therefore, the hyper-parameters for each task finetuning are not optimized, and the results here have to be taken with a grain of salt.

Source task	Target task 1: Phone Recog (in PER)	Target task 2: Speaker Verification (in EER)	Target task 3: Slot Filling (in slot type $F_1$ /slot value CER)
10h Librispeech ASR	0.0567	0.1230	0.7635/0.4432
1h Librispeech ASR	0.0567	0.1316	0.7563/0.4470
10min Librispeech ASR	0.0576	0.1399	0.7452/0.4596
10h Phone Recog	0.0471	0.1392	0.7575/0.4468
1h Phone Recog	0.0483	0.1138	0.7508/0.4537
10min Phone Recog	0.0535	0.1224	0.7519/0.4596
Intent Classification	0.0617	0.1165	0.7490/0.4621
Slot Filling	0.0601	0.1097	0.7708/0.4327
Keyword Spotting	0.0656	0.1303	0.7490/0.4661
Speaker Verification	0.0790	0.1131	0.7497/0.4654
Speaker ID	0.0677	0.1271	0.7581/0.4559
Speaker Diarization	0.0756	0.1104	0.7449/0.4623

Table 7: Cross-task mask transfer for wav2vec-base at 50% sparsity.

We first see that indeed the more similar source and target tasks are, the performance are better. For instance, source subnetwork obtained from speaker related task perform better than those obtained from ASR/keyword spotting on speaker verification. For another, source subnetwork obtained from ASR/phone recognition perform better than those obtained from speaker related task on phone recognition. We do note that the numbers are not off by too much, and the differences could be potentially reduced via hyper-parameter tuning. This pilot study also suggests that subnetworks transferability depends on task similarity. Lastly, this experiment does not contradict our main setting, as we were primarily interested in cross-lingual transferability of subnetworks in Section 4.3.

<sup>&</sup>lt;sup>15</sup>All experiments are run with SUPERB's toolkit https://github.com/s3pr1/s3pr1.

## 17 Does Observation 1 generalize across Pre-Training Objectives?

Observation 1 states that:

For any sparsity, any amount of finetuning supervision, any pre-training model scale, and any downstream spoken languages, the non-zero ASR pruning masks obtained from taskagnostic subnetwork discovery has high IOUs with those obtained from task-aware subnetwork discovery.

We provide analysis on whether Observation 1 holds *across* pre-training objectives, i.e. does pruning masks from wav2vec2 have high similarity with those from hubert [55]? The setup follows that of Section 16 and is based on the downstream tasks in SUPERB<sup>16</sup>.

- 1. Finetune wav2vec2 for all tasks in SUPERB.
- 2. Prune the finetuned models and obtain an OMP mask for each task.
- 3. Finetune hubert for all tasks in SUPERB
- 4. Prune the finetuned models and obtain an OMP mask for each task.
- 5. For each task in SUPERB and at a fixed sparsity, calculate the mask IOU between wav2vec2 and hubert.

Table 8 is the mask IOUs at 50% sparsity between wav2vec-base and hubert-base on tasks in SUPERB. The table indicates that while Observation 1 holds separately for wav2vec2 (contrastive pre-training) and hubert (mask-predict pre-training), it does not generalize across pre-training method give the close to random mask IOUs (c.f. last row of Table 8). Therefore,

*Observation* **1** *holds true conditioned on the same speech SSL pre-training objective.* 

target task	mask IOU between wav2vec-base and hubert-base
10h Librispeech ASR 1h Librispeech ASR	0.3472
10min Librispeech ASR	0.3473
10h Phone Recog 1h Phone Recog 10min Phone Recog	0.3473 0.3473 0.3473
Intent Classification Slot Filling	0.3473 0.3472
Keyword Spotting	0.3473
Speaker Verification Speaker ID Speaker Diarization	0.3473 0.3473 0.3472
Random Pruning	0.3473

Table 8: Mask IOU between wav2vec-base and hubert-base at 50% sparsity.

This finding is perhaps not so surprising, see prior work on similarity analysis between contextualized speech [24] and word [113] representations. They suggest that different pre-trained models' contextualized representations have low similarities, e.g. BERT v.s. XLNet. We stress that this does not invalidate PARP. As long as Observation [] holds, PARP's step 2 should make learnable adjustments to the initial mask given the high overlaps between pruning masks.

<sup>&</sup>lt;sup>16</sup>For this set of experiments, we used the same optimization method (Adam with constant  $1.0 \times 10^{-5}$  learning rate) for finetuning wav2vec-base and hubert-base.

#### 18 Pruned Weights Localization Across Layers

The wav2vec series [6, 4, 55, 53] is known to have more valuable contextualized representations towards the middle of the network for downstream ASR. We examine whether previous observations holds true for pruning, that weights in middle layers are pruned less. To understand such a phenomenon, we calculated the distributions of the pruned weights/neurons across each layer, and an example is shown in Table 9.

Table 9: wav2vec-base finetuned for Spanish (H2L setting) pruned at 50% sparsity with OMP.

layer		1		2	3	4		5	6	7	8	9	10	11	12
sparsity (%	6)   5	3.52	52	2.45	49.24	47.90	4	46.51	46.84	45.97	45.58	45.96	47.96	52.54	65.53

Table shows that indeed bottom and higher layers of wav2vec2-base are pruned more, while the middle layers are pruned less. We observe similar pruned weight distributions across spoken languages (10 languages) and sparsities (10%, 20%, 30%, ..., 90%). See the rest of the sparsity distribution in the Figures below. This analysis suggests that regardless of spoken languages, intermediate layers' neurons are more valuable than lower and higher-level layers, manifested by the layer's sparsity ratio.

						es_bert_	0.1_mask						- 100
Sparsity %	10.44	10.07	9.3317	9.1451	9.0708	9.3758	9.3869	9.4126	9.22	9.1605	9.9586	15.429	- 50
01	ò	i	ź	3	4	5 wav2vec2	6 BERT Layer	7	8	9	10	11	- 0

Figure 41: Sparsity over layers for wav2vec-base finetuned for Spanish es at 10% sparsity.





Figure 42: Sparsity over layers for wav2vec-base finetuned for Spanish es at 20% sparsity.





Figure 44: Sparsity over layers for wav2vec-base finetuned for Spanish es at 40% sparsity.



Figure 45: Sparsity over layers for wav2vec-base finetuned for Spanish es at 50% sparsity.



Figure 46: Sparsity over layers for wav2vec-base finetuned for Spanish es at 60% sparsity.

					es_bert_(	0.7_mask					
.388	72.905	69.835	68.448	66.828	66.894	65.475	64.932	65.712	68.781	73.916	82.887
ό	i	2	3	4	5	6	7	8	9	10	11
	0.388	8.388 72.905 0 1	0 1 2	0.388 72.905 69.835 68.448	0 1 2 3 4	.388 72.905 69.835 68.448 66.828 66.894 0 1 2 3 4 5	0.388 72.905 69.835 68.448 66.828 66.894 65.475 0 1 2 3 4 5 6		388 72.905 69.835 68.448 66.828 66.894 65.475 64.932 65.712   0 1 2 3 4 5 6 7 8	388 72.905 69.835 68.448 66.828 66.894 65.475 64.932 65.712 68.781   0 1 2 3 4 5 6 7 8 9	388 72.905 69.835 68.448 66.828 66.894 65.475 64.932 65.712 68.781 73.916   0 1 2 3 4 5 6 7 8 9 10

Figure 47: Sparsity over layers for wav2vec-base finetuned for Spanish es at 70% sparsity.



Figure 48: Sparsity over layers for wav2vec-base finetuned for Spanish es at 80% sparsity.



Figure 49: Sparsity over layers for wav2vec-base finetuned for Spanish es at 90% sparsity.



Figure 50: Sparsity over layers for wav2vec-base finetuned for French fr at 10% sparsity.



Figure 51: Sparsity over layers for wav2vec-base finetuned for French fr at 20% sparsity.



Figure 52: Sparsity over layers for wav2vec-base finetuned for French fr at 30% sparsity.



Figure 53: Sparsity over layers for wav2vec-base finetuned for French fr at 40% sparsity.



Figure 54: Sparsity over layers for wav2vec-base finetuned for French fr at 50% sparsity.

						.6_mask					
	62.825	59.51							58.253	63.334	74.783
ó	i	ź	3	4	5	6	7	8	9	10	11
	.679 0	.679 62.825 0 1	.679 62.825 59.51 0 1 2	.679 62.825 59.51 58.073 0 1 2 3	0 1 2 3 4	0 1 2 3 4 5	0 1 2 3 4 5 6	0 1 2 3 4 5 6 7	679 62 825 59.51 58.073 56.521 56.731 55.566 55.074 55.651 0 1 2 3 4 5 6 7 8 wav2vec2 BERT Layer	0 1 2 3 4 5 6 7 8 9	

Figure 55: Sparsity over layers for wav2vec-base finetuned for French fr at 60% sparsity.



Figure 56: Sparsity over layers for wav2vec-base finetuned for French fr at 70% sparsity.



Figure 57: Sparsity over layers for wav2vec-base finetuned for French fr at 80% sparsity.



Figure 58: Sparsity over layers for wav2vec-base finetuned for French fr at 90% sparsity.



Figure 59: Sparsity over layers for wav2vec-base finetuned for Italian it at 10% sparsity.



Figure 60: Sparsity over layers for wav2vec-base finetuned for Italian it at 20% sparsity.



Figure 61: Sparsity over layers for wav2vec-base finetuned for Italian it at 30% sparsity.



Figure 62: Sparsity over layers for wav2vec-base finetuned for Italian it at 40% sparsity.

						it_bert_0	).5_mask						- 10
Sparsity %				47.895	46.512	46.848	45.964	45.577	45.954	47.96		65.521	- 50
3	ó	i	2	3	4	5	6	ź	8	9	10	'n	- 0
						wav2vec2	BERT Layer	r					

Figure 63: Sparsity over layers for wav2vec-base finetuned for Italian it at 50% sparsity.



Figure 64: Sparsity over layers for wav2vec-base finetuned for Italian it at 60% sparsity.



Figure 65: Sparsity over layers for wav2vec-base finetuned for Italian it at 70% sparsity.



Figure 66: Sparsity over layers for wav2vec-base finetuned for Italian it at 80% sparsity.



Figure 67: Sparsity over layers for wav2vec-base finetuned for Italian it at 90% sparsity.



Figure 68: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 10% sparsity.



Figure 69: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 20% sparsity.



Figure 70: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 30% sparsity.

						ky_bert_(	0.4_mask						- 100
Sparsity %	42.974	41.854	39.015	37.914	36.821	37.214	36.599	36.353	36.549	37.911	41.683		- 50
স	Ó	i	ź	3	4	5	6	7	8	9	10	11	- 0
wav2vec2 BERT Layer													

Figure 71: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 40% sparsity.



Figure 72: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 50% sparsity.



Figure 73: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 60% sparsity.



Figure 74: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 70% sparsity.



Figure 75: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 80% sparsity.



Figure 76: Sparsity over layers for wav2vec-base finetuned for Kyrgyz ky at 90% sparsity.



Figure 77: Sparsity over layers for wav2vec-base finetuned for Dutch *nl* at 10% sparsity.



Figure 78: Sparsity over layers for wav2vec-base finetuned for Dutch nl at 20% sparsity.

						nl_bert_(	.3_mask						- 100
Sparsity %	32.131	31.149	28.901	28.108	27.388	27.805	27.458	27.322	27.377	28.13	30.906	43.326	- 50
5	Ó	i	ź	3	4	5 wav2vec2	6 BERT Layer	7	8	9	10	11	- 0

Figure 79: Sparsity over layers for wav2vec-base finetuned for Dutch *nl* at 30% sparsity.



Figure 80: Sparsity over layers for wav2vec-base finetuned for Dutch *nl* at 40% sparsity.







Figure 82: Sparsity over layers for wav2vec-base finetuned for Dutch nl at 60% sparsity.



Figure 83: Sparsity over layers for wav2vec-base finetuned for Dutch nl at 70% sparsity.



Figure 84: Sparsity over layers for wav2vec-base finetuned for Dutch nl at 80% sparsity.



Figure 85: Sparsity over layers for wav2vec-base finetuned for Dutch *nl* at 90% sparsity.

						ru_bert_	0.1_mask						- 100
Sparsity %	10.461	10.079	9.3561	9.1535	9.0737	9.3645	9.3608	9.3959	9.1946	9.178	9.9775	15.405	- 50
0,	Ó	i	2	3	4	5 wav2vec2	6 BERT Layer	7	8	9	10	11	- 0

Figure 86: Sparsity over layers for wav2vec-base finetuned for Russian ru at 10% sparsity.

						ru_bert_	0.2_mask						- 100
Sparsity %	21.221	20.487	19.006	18.524	18.148	18.56	18.424	18.41	18.307	18.557	20.311	30.045	- 50
٥.	ό	i	ź	3	4	5 way2yec2	6 BERT Layer	7	8	9	10	11	- 0

Figure 87: Sparsity over layers for wav2vec-base finetuned for Russian ru at 20% sparsity.

						ru_bert_	0.3_mask						- 100
Sparsity %	32.131	31.154	28.901	28.11	27.38	27.805	27.456	27.327	27.387	28.132	30.891	43.327	- 50
01	Ó	i	ź	3	4	5 wav2vec2	6 BERT Layer	7	8	9	10	11	- 0

Figure 88: Sparsity over layers for wav2vec-base finetuned for Russian ru at 30% sparsity.



Figure 89: Sparsity over layers for wav2vec-base finetuned for Russian ru at 40% sparsity.



Figure 90: Sparsity over layers for wav2vec-base finetuned for Russian ru at 50% sparsity.



Figure 91: Sparsity over layers for wav2vec-base finetuned for Russian ru at 60% sparsity.



Figure 92: Sparsity over layers for wav2vec-base finetuned for Russian ru at 70% sparsity.



Figure 93: Sparsity over layers for wav2vec-base finetuned for Russian ru at 80% sparsity.



Figure 94: Sparsity over layers for wav2vec-base finetuned for Russian ru at 90% sparsity.

						sv_SE_bert	_0.1_mask						- 100
Sparsity %	10.437	10.062	9.3298	9.1494	9.0698	9.3822	9.3808	9.4172	9.2101	9.1633	9.9618	15.437	- 50
S	Ó	i	2	3	4	5 wav2vec2	6 BERT Layer	7	8	9	10	11	- 0

Figure 95: Sparsity over layers for wav2vec-base finetuned for Swedish *sv-SE* at 10% sparsity.



Figure 96: Sparsity over layers for wav2vec-base finetuned for Swedish sv-SE at 20% sparsity.



Figure 97: Sparsity over layers for wav2vec-base finetuned for Swedish sv-SE at 30% sparsity.



Figure 98: Sparsity over layers for wav2vec-base finetuned for Swedish sv-SE at 40% sparsity.



Figure 99: Sparsity over layers for wav2vec-base finetuned for Swedish *sv-SE* at 50% sparsity.



Figure 100: Sparsity over layers for wav2vec-base finetuned for Swedish sv-SE at 60% sparsity.



Figure 101: Sparsity over layers for wav2vec-base finetuned for Swedish sv-SE at 70% sparsity.



Figure 102: Sparsity over layers for wav2vec-base finetuned for Swedish sv-SE at 80% sparsity.

					fr_bert_0	.9_mask					
91.613	91.758	90.399	89.763	88.646	88.259	86.662	86.315	87.39	90.193	93.179	95.824
6	i	2	3	4	5	6	7	8	9	10	11

Figure 103: Sparsity over layers for wav2vec-base finetuned for Swedish sv-SE at 90% sparsity.



Figure 104: Sparsity over layers for wav2vec-base finetuned for Turkish tr at 10% sparsity.







Figure 106: Sparsity over layers for wav2vec-base finetuned for Turkish tr at 30% sparsity.



Figure 107: Sparsity over layers for wav2vec-base finetuned for Turkish tr at 40% sparsity.



Figure 108: Sparsity over layers for wav2vec-base finetuned for Turkish tr at 50% sparsity.



Figure 109: Sparsity over layers for wav2vec-base finetuned for Turkish tr at 60% sparsity.



Figure 110: Sparsity over layers for wav2vec-base finetuned for Turkish tr at 70% sparsity.

						tr_bert_0	.8_mask					
% -	82.698	82.575	80.148	79.018	77.502	77.397	75.771	75.248	76.223	79.48	84.015	89.923
	Ó	i	2	3	4	5	6	7	8	9	10	'n
						wav2vec2	BERT Layer					

Figure 111: Sparsity over layers for wav2vec-base finetuned for Turkish tr at 80% sparsity.



Figure 112: Sparsity over layers for wav2vec-base finetuned for Turkish tr at 90% sparsity.



Figure 113: Sparsity over layers for wav2vec-base finetuned for Tatar *tt* at 10% sparsity.



Figure 114: Sparsity over layers for wav2vec-base finetuned for Tatar *tt* at 20% sparsity.



Figure 115: Sparsity over layers for wav2vec-base finetuned for Tatar *tt* at 30% sparsity.



Figure 116: Sparsity over layers for wav2vec-base finetuned for Tatar *tt* at 40% sparsity.



Figure 117: Sparsity over layers for wav2vec-base finetuned for Tatar tt at 50% sparsity.



Figure 118: Sparsity over layers for wav2vec-base finetuned for Tatar *tt* at 60% sparsity.

				tt_bert_0	.7_mask					
2.905	69.837	68.452	66.835	66.901	65.474	64.929	65.713	68.775	73.911	82.876
i	2	3	4	5	6	7	8	9	10	11
1	-	5		wav2v	ec2	ec2 BERT Laver	ec2 BERT Laver	ec2 BERT Laver	ec2 BERT Laver	

Figure 119: Sparsity over layers for wav2vec-base finetuned for Tatar *tt* at 70% sparsity.



Figure 120: Sparsity over layers for wav2vec-base finetuned for Tatar tt at 80% sparsity.





						zh_⊤W_ber	t_0.1_mask						- 100
parsity %	10.474	10.089	9.3543	9.163	9.0791	9.3721	9.3652	9.3998	9.2128	9.1735	9.9835	15.334	- 50
01	ò	i	ź	3	4	5 wav2vec2	6 BERT Layer	7	8	9	10	11	- 0

Figure 122: Sparsity over layers for wav2vec-base finetuned for Mandarin *zh-TW* at 10% sparsity.



Figure 123: Sparsity over layers for wav2vec-base finetuned for Mandarin *zh-TW* at 20% sparsity.



Figure 124: Sparsity over layers for wav2vec-base finetuned for Mandarin *zh-TW* at 30% sparsity.



Figure 125: Sparsity over layers for wav2vec-base finetuned for Mandarin *zh-TW* at 40% sparsity.



Figure 126: Sparsity over layers for wav2vec-base finetuned for Mandarin zh-TW at 50% sparsity.

4 62.83	3 59.525							58.257	63.331	74.727
i	ź	3	4	5	6	7	8	9	10	ú
	i	1 2	1 2 3	1 2 3 4	1 2 3 4 5	1 2 3 4 5 6	1 2 3 4 5 6 7 wav2vec2 BERT Layer	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8 9	

Figure 127: Sparsity over layers for wav2vec-base finetuned for Mandarin *zh-TW* at 60% sparsity.



Figure 128: Sparsity over layers for wav2vec-base finetuned for Mandarin *zh-TW* at 70% sparsity.



Figure 129: Sparsity over layers for wav2vec-base finetuned for Mandarin *zh-TW* at 80% sparsity.



Figure 130: Sparsity over layers for wav2vec-base finetuned for Mandarin *zh-TW* at 90% sparsity.

## **19** Experimental Limitations

Below, we list several limiting factors of our experimental designs:

- 1. Experiments are on contrastive pre-trained models only. It is unclear whether the results would generalize to pre-trained models with other objectives, such as mask prediction (HuBERT) or autoregressive prediction (APC), etc.
- 2. Although standard, our experiments are on relatively large pre-trained models (number of parameter is 90M for wav2vec2-base and 315M for wav2vec2-large and xlsr. It would be interesting to investigate if small pre-trained models can also be pruned and whether Observation 1 holds for them.
- 3. Our wav2vec2-base and wav2vec2-large are both pre-trained on Librispeech 960 hours. Another lack of study is the effect of pre-training data selections – what happens if pretraining and fine-tuning data are from different sources?
- 4. Our fine-tuning dataset (Librispeech and CommonVoice) are both read speech. Experiments on conversational (e.g. telephone) speech should be investigated.
- 5. In addition, though opposite to our motivation, it is unclear is the results hold for high-resource languages (e.g. 100h~1000h of fine-tuning data).
- 6. Our ASR experiments are based on self-supervised pre-trained models. It remains to be studied on applying PARP to E2E ASR without self-supervised pre-training.
- 7. Lastly, we note that this study is scientific by nature. Observation [] emerges after our initial pilot study, and it motivates the central idea of PARP. We will leave it to follow-up work to test whether such pruning method is effective in more realistic settings (e.g. noisy data, limited bandwidth, etc).