Side-Tuning: A Baseline for Network Adaptation via Additive Side Networks: Supplementary Material

Jeffrey O. Zhang[1], Alexander Sax[1], Amir Zamir[3], Leonidas Guibas[2], and Jitendra Malik[1]

[1] UC Berkeley
[2] Stanford University
[3] Swiss Federal Institute of Technology (EPFL)
http://sidetuning.berkeley.edu

Abstract. The following items are provided in the supplementary material:

1. Experimental Details
2. Additional Experiments
3. iCIFAR Data
4. iTaskonomy Data

1 Experimental Details

For full details on our configs, please refer to ./configs in provided code on our website at http://sidetuning.berkeley.edu.

1.1 Experimental Setup for Incremental Learning

Taskonomy Our data is 4M images on 12 single image tasks. The tasks that we use are the following: curvature, semantic segmentation, reshading, keypoints3d, keypoints2d, texture edges, occlusion edges, distance, depth, surface normals, object classification and autoencoding. The tasks were chosen in no particular special order. Our base model and side model are ResNet-50s. We pretrain on curvature. Then, we train each task for three epochs before moving on to the next task. We use cross entropy loss for classification tasks (semantic segmentation and object classification), L2 loss for curvature and L1 loss for the other tasks. We use Adam optimizer with an initial learning rate of 1e-4, weight decay coefficient of 2e-6, gradient clipping to 1.0, and batch size of 32. We evaluate our performance on a held out set of images, both immediately after training a specific task, and after training of all the tasks are complete.

iCIFAR We start by pretraining a model on CIFAR 10 (from `https://github.com/akamaster/pytorch_resnet_cifar10`). Then we partition CIFAR100 into 10 distinct sets of 10 classes. Then, we train for 4 epochs on these tasks using Adam optimizer, learning rate of 1e-3, batch size of 128.

1.2 Experimental Setup for Additional Domains

NLP We train and test on the the question answering dataset SQuAD2.0, a reading comprehension dataset consisting of 100,000 questions with 50,000 unanswerable questions. Both our base encoding and side network is a BERT transformer pretrained on a larger corpus. Finetuning trains a single BERT transfer. We use the training setup found at `https://github.com/huggingface/pytorch-transformers` (train for 2 epochs at a learning rate of 3e-5) with one caveat - we use an effective batch size of 3 (vs. their 24).

Habitat Experiments We borrow the experimental setup from [1]:

> We use the Habitat environment with the Gibson dataset. The dataset virtualizes 572 actual buildings, reproducing the intrinsic visual and semantic complexity of real-world scenes.
> We train and test our agents in two disjoint sets of buildings. During testing we use buildings that are different and completely unseen during training. We use up to 72 building for training and 14 test buildings for testing. The train and test spaces comprise $15678.4m^2$ (square meters) and $1752.4m^2$, respectively.
> The agent must direct itself to a given nonvisual target destination (specified using coordinates), avoiding obstacles and walls as it navigates to the target. The maximum episode length is 500 timesteps, and the target distance is between 1.4 and 15 meters from the start.

This setup is shared between imitation learning and RL, which differ in the data, architecture and optimization process.

Imitation Learning We collect 49,325 shortest path expert trajectories in Habitat, 2,813,750 state action pairs. We learn a neural network mapping from states to actions. Our base encoding is a ResNet-50 and the side network is a five layer convolutional network. The representation output is then fed into a neural network policy. We train the model for 10 epochs using cross entropy loss and Adam at an initial learning rate of 2e-4 and weight decay coefficient of 3.8e-7. We initialize alpha to 0.5. Finetuning uses the same model architecture but updates all the weights. Feature extraction only uses the ResNet-50 to collect features.

RL Similarly, we borrow the RL setup from [1].

> In all experiments we use the common Proximal Policy Optimization (PPO) algorithm with Generalized Advan- tage Estimation. Due to the computational load of ren- dering perceptually realistic images in Gibson we are only able to use a single rollout worker and we therefore decorre- late our batches using experience replay and off-policy vari- ant of PPO. The formulation is similar to Actor-Critic with Experience Replay (ACER) in that full trajectories are sampled from the replay buffer and reweighted using the first-order approximation for importance sampling.

During training, the agent receives a large one-time reward for reaching the goal, a positive reward proportional to Euclidean distance toward the goal and a small negative reward each timestep. The maximum episode length is 500 timesteps, and the target distance is between 1.4 and 15 meters from the start.

Due to this paradigms' compute and memory constraints, it would be difficult for us to use large architectures for this setting. Thus, our base encoding is a five layer convolutional network distilled from the trained ResNet-50. Our side network is also a five layer convoultional network. Finetuning is handled the same way - update all the weights in this setup. Feature extraction uses the five layer network to collect features.

1.3 Experimental Setup for Learning Mechanics

Low energy initialization In classical teacher student distillation, the student is trained to minimize the distance between its output and the teacher's output. In this setting, we minimize the distance between the teacher's output and the summation of the student's output and the teacher's output. The output space may have a different geometry than that of the input space.

2 Additional Experiments

2.1 Task relevance predicts alpha α.

In our experiments, we treat α as a learnable parameter (initialized to 0.5) and find that the relative values of α are predictive of empirical performance. In imitation learning, *curvature* ($\alpha = 0.557$) outperformed *denoising* ($\alpha = 0.252$). In iTaskonomy, the α values from training on just 100 images predicted the actual transfer performance to normals in [2], (e.g. *curvature* ($\alpha = 0.56$) outperformed *object classification* ($\alpha = 0.50$)). For small datasets, usually $\alpha \approx 0.5$ and the relative order, rather than the actual value is important.

2.2 Fusion

An alternative perspective views these methods as various fusions between some base output and new side output. In this framework, side-tuning is a late-fusion approach whereas PNN is a distributed-fusion approach. We compare various fusion methods in iCIFAR and find that late fusion performs better than early fusion and distributed fusion (error of 23.9 vs. 38.8 and 26.3 respectively). The fusion is performed with a MLP. We run this analysis in iTaskonomy as well. We fuse with no new parameters, merging with summation and combine outputs with alpha-blending. We find that late fusion outperforms early and distributed fusion as well (rank of 1.25 of 3 vs. 1.92 and 2.0 respectively).

2.3 Imitation Learning Data Study

We show additional results from the data study in imitaiton learning in Figure 1. The results overall show *Side-tuning*'s advantage.

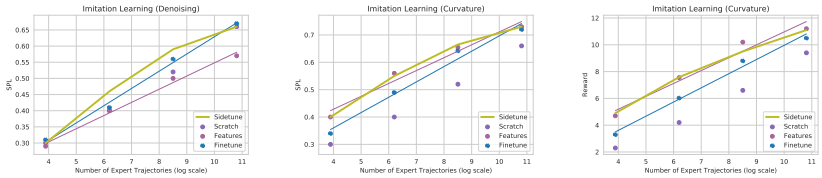

Fig. 1. Additional Imitation Learning Data Study. We ablate over different quantities of expert trajectories. We observe that when data is scarce, features is a powerful choice whereas when data is plentiful, fine-tuning performs well. In both scenarios, *side-tuning* is able to perform as well as the stronger approach.

2.4 Additional iCIFAR Comparisons with Progressive Neural Networks

In the main paper, for our iCIFAR experiments, we see that the average performance of side-tuning weaker than that of PNN. We find that side-tuning can bridge this gap with a multilayer perceptron (adapter) to merge the base and side networks. This is a common practice in PNN. We see with the adapter network, the two methods are very similar when measuring classification error (23.69 of PNN vs. 23.91 of *Side-tuning*).

2.5 Extremely Few-Shot Learning

In domains with very few examples, we found that *side-tuning* is unable to match the performance of other methods. We evaluated our setup in vision transfer for 5 images from the same building, imitation learning given 5 expert trajectories.

Methods	Nav. Rew. (4 epi) (↑)		Loss (5 ims) (↓)
	Curvature	Denoise	Curvature to Normals
Finetune	-0.1	-1.2	0.35
Features	0.4	1.2	0.36
Scratch	-0.9	-0.9	0.37
Sidetune	-0.3	-1.8	0.42

3 iCIFAR data

3.1 Average Final Classification error

Here, we show the average classification error at the end of training.

	Avg. Error
Indep.	14.81
PNN	23.69
Side-tuning (MLP)	23.91
Late Fusion	23.91
Dist. Fusion	26.34
Side-tuning	32.77
EWC	34.86
Early Fusion	38.84
EWC $\lambda = 10^4$	41.39
EWC $\lambda = 10^6$	41.88
Feat.	42.27
Res. Adapter	46.26
Piggyback	47.72
PSP	50.82
Fine-tuning	77.75

3.2 Final Classification error for all methods on all tasks

	Task 0	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9
Side-tuning	31.7	31.2	33.1	36.4	30.3	34.6	33.0	36.8	33.1	27.5
PSP	57.9	63.9	50.9	60.4	47.0	51.5	46.6	47.5	50.3	32.2
Fine-tuning	94.2	87.3	91.9	91.0	89.3	88.9	88.0	79.3	54.6	13.0
Res. Adapter	47.2	53.7	46.8	49.1	45.5	43.3	45.1	43.2	44.5	44.2
Side-tuning (MLP)	26.2	24.1	23.4	26.9	23.3	23.8	23.0	26.1	23.8	18.5
PNN	25.4	27.2	24.6	23.9	23.2	24.5	22.4	28.6	22.0	15.1
Indep.	12.2	17.2	12.8	13.8	16.1	17.1	15.9	18.5	13.5	11.0
EWC	19.6	33.1	35.6	41.6	35.6	41.4	37.1	38.7	40.1	25.8
Piggyback	48.7	48.0	45.9	51.0	49.2	48.8	46.1	49.9	43.6	46.0
Feat.	45.3	37.9	47.1	41.0	40.1	45.5	42.8	46.3	40.9	35.8
EWC $= 10^6$	14.1	41.1	45.0	47.6	42.2	44.5	46.3	47.0	47.2	38.9
EWC $= 10^4$	56.5	54.5	50.6	45.0	42.7	39.5	40.0	39.1	34.7	16.2
Early Fusion	38.4	44.9	39.4	43.6	37.8	36.5	41.0	36.9	37.8	32.1
Late Fusion	26.2	24.1	23.4	26.9	23.3	23.8	23.0	26.1	23.8	18.5
Dist. Fusion	26.1	27.7	26.6	27.7	23.9	29.4	27.1	27.9	25.8	21.2

4 iTaskonomy Data

4.1 Rank of *Side-tuning* and Baselines

Here we provide rankings for the baselines introduced.

	Rank
Side-tuning	2.000
Indep.	2.000
PNN	3.500
Piggyback	4.333
Feat.	5.000
Res. Adapter	5.250
EWC	7.583
PSP	7.667
Fine-tuning	7.667

4.2 Rank of all methods

Here, we provide the rank of all methods that we have tested in the incremental learning setup (including all ablations/analysis in addition to baselines). Note the effectiveness of using ground truth curvatures as the base model!

	Rank
Indep.	4.167
GT Curv as Base	4.333
Side-tuning	5.250
FiLM	5.667
MLP	6.417
Dist. Fusion	7.000
Xavier Init.	8.500
MLP2	8.583
PNN	8.833
Low Energy Init.	9.167
No Base	10.583
Mult.	10.583
Piggyback	10.750
Early Fusion	11.667
Feat.	12.417
No Side	12.417
Res. Adapter	14.167
EWC	17.667
$EWC(\lambda = 0.01)$	17.750
$EWC(\lambda = 100)$	18.000
Fine-tuning	18.000
PSP	18.583

4.3 Final Task Performance for all Tasks and Methods

Here we provide the final performance for all of the tested methods on all the incremental learning tasks in Taskonomy.

	Side-tuning	PSP	Fine-tuning	PNN	Indep.	EWC	Feat.
Curvature	1.234	1.810	1.787	1.230	1.251	1.829	1.240
Semantic Segmentation	1.347	1.626	1.505	1.350	1.347	1.502	1.353
Reshading	0.613	2.377	2.127	0.621	0.638	1.677	0.791
3D Keypoints	1.986	3.026	3.486	2.038	1.993	3.096	1.988
2D Keypoints	0.896	4.593	5.046	1.042	0.742	4.988	2.504
Texture Edges	0.160	1.571	1.374	0.165	0.141	1.007	0.377
Occlusion Edges	0.927	1.147	1.177	0.936	0.933	1.235	0.928
Z-buffer Depth	1.011	3.222	3.587	1.060	1.018	3.706	1.214
Distance	1.117	2.900	3.401	1.098	1.001	3.860	1.187
Surface Normals	0.618	1.912	2.040	0.644	0.569	1.990	0.649
Object Classification	2.986	15.950	5.042	2.996	2.829	5.103	3.183
Autoencoding	1.271	1.337	1.179	1.477	1.138	1.155	5.337

	Piggyback	Res. Adapter	Low E Init.	Xavier Init.	No Base	No Side
Curvature	1.245	1.370	1.239	1.237	1.294	1.240
Semantic Segmentation	1.350	1.382	1.351	1.352	1.353	1.353
Reshading	0.646	0.743	0.699	0.702	0.923	0.791
3D Keypoints	1.976	2.226	1.984	1.988	2.060	1.988
2D Keypoints	2.442	1.009	0.890	0.889	0.870	2.504
Texture Edges	0.355	0.171	0.154	0.160	0.155	0.377
Occlusion Edges	0.935	0.979	0.929	0.930	0.944	0.928
Z-buffer Depth	1.120	1.314	1.264	1.163	1.045	1.214
Distance	1.170	1.387	1.179	1.138	1.219	1.187
Surface Normals	0.638	0.723	0.645	0.638	0.651	0.649
Object Classification	3.187	3.044	3.126	3.099	2.960	3.183
Autoencoding	4.939	1.298	1.312	1.256	1.262	5.337

	GT Curv as Base	$EWC(\lambda = 0.01)$	$EWC(\lambda = 100)$	MLP2	MLP
Curvature	0.783	1.626	1.905	1.244	1.247
Semantic Segmentation	1.351	1.472	1.496	1.351	1.351
Reshading	0.591	1.955	1.668	0.603	0.844
3D Keypoints	1.702	3.125	3.229	2.000	1.997
2D Keypoints	0.879	4.649	6.445	0.967	0.865
Texture Edges	0.151	1.036	0.969	0.164	0.150
Occlusion Edges	0.928	1.191	1.320	0.932	0.926
Z-buffer Depth	1.050	3.726	3.168	1.074	1.011
Distance	1.076	3.640	3.389	1.132	1.080
Surface Normals	0.562	2.139	2.112	0.619	0.612
Object Classification	2.956	5.280	5.680	2.979	2.955
Autoencoding	1.294	1.163	1.223	1.383	1.312

	Mult.	FiLM	Early Fusion	Dist. Fusion
Curvature	1.235	1.243	1.318	1.230
Semantic Segmentation	1.354	1.352	1.364	1.348
Reshading	0.813	0.827	0.720	0.631
3D Keypoints	1.990	1.979	2.148	1.984
2D Keypoints	1.369	0.840	0.855	0.974
Texture Edges	0.224	0.145	0.166	0.167
Occlusion Edges	0.934	0.928	0.948	0.925
Z-buffer Depth	1.097	1.028	1.158	1.048
Distance	1.085	1.075	1.219	1.083
Surface Normals	0.625	0.610	0.691	0.626
Object Classification	3.005	2.991	3.022	2.999
Autoencoding	2.012	1.264	1.224	1.275

4.4 Qualitative Results of *Side-tuning* vs PNN

We show predictions for *Side-tuning* and PNN side-by-side for three tasks (relative to Ground Truth and Independent). We show two variants of PNN, the first following the paper closely, the second with minor variations.

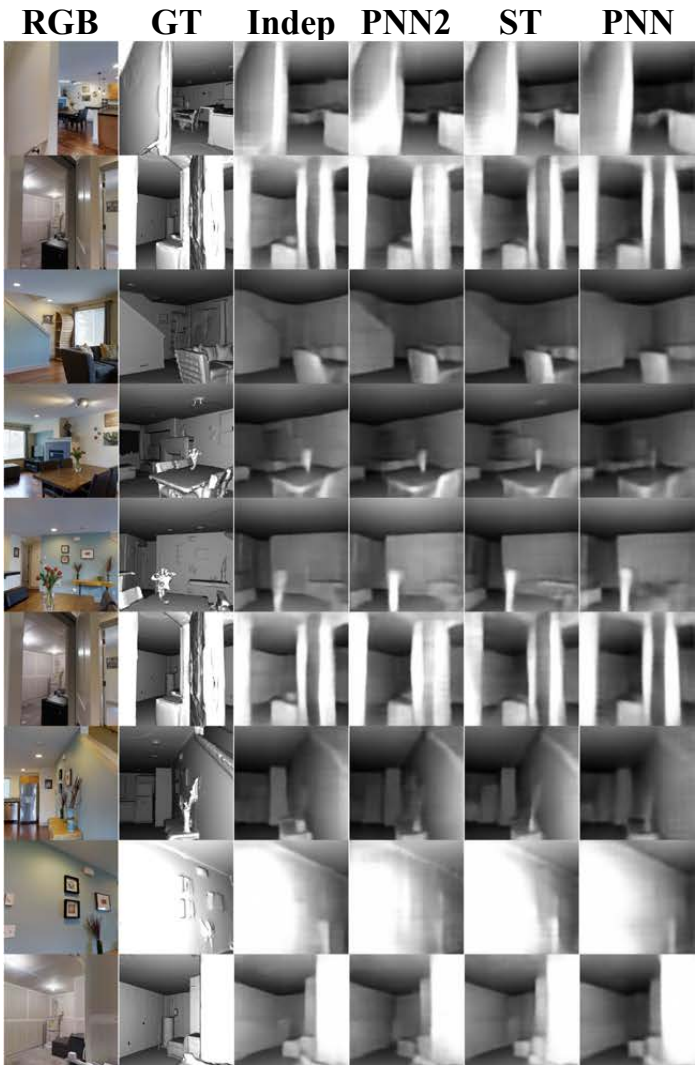

Fig. 2. Qualitative results for Reshading. Both PNN methods and Sidetune have similar qualitative results.

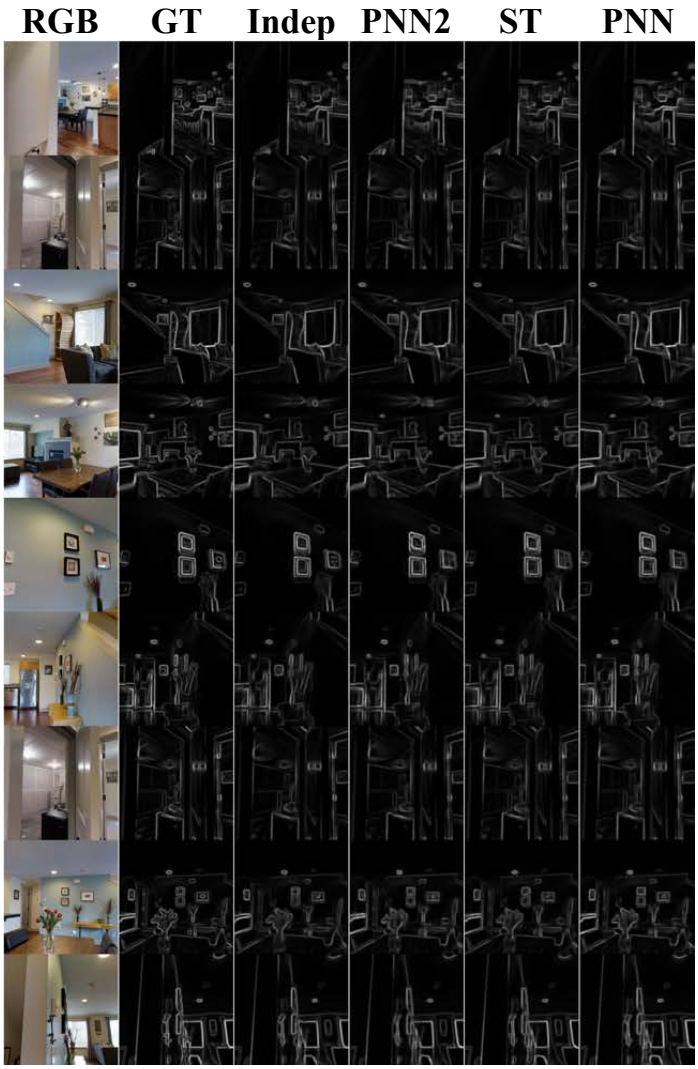

Fig. 3. Qualitative results for 2D Edges. Both PNN methods and Sidetune have similar qualitative results.

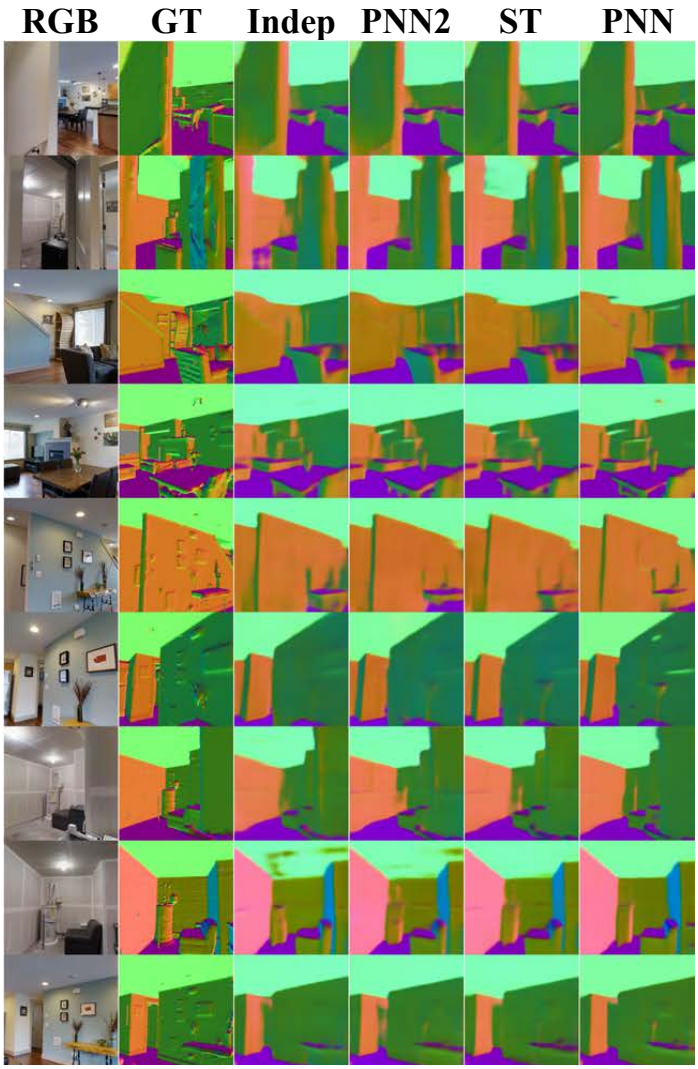

Fig. 4. Qualitative results for Surface Normals. Both PNN methods and Sidetune have similar qualitative results.

4.5 Qualitative Results of Baselines

We show predictions for each method (*Side-tuning*, EWC, PSP, PNN, Independent) separately for some fixed set of randomly selected images throughout training.

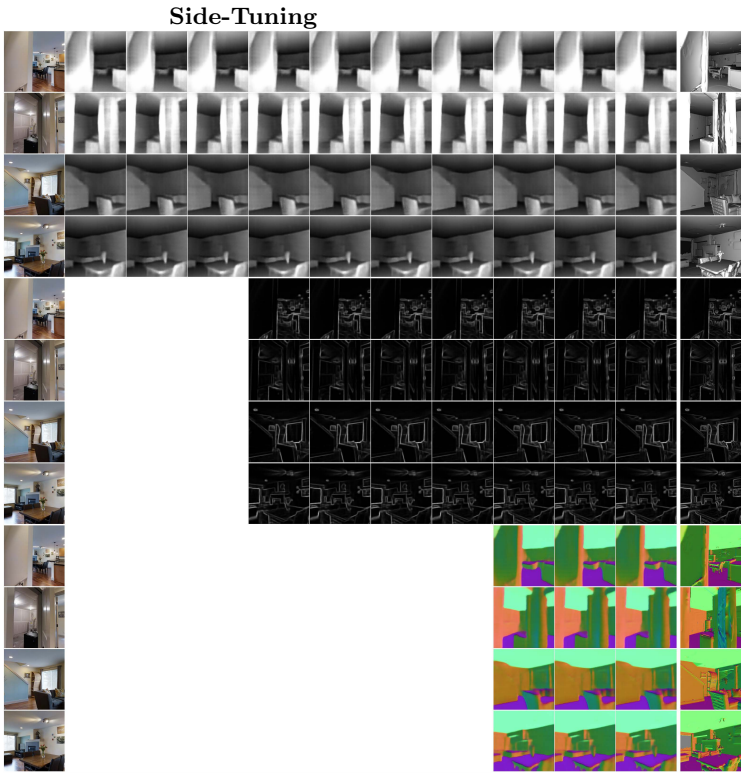

Fig. 5. More qualitative results for *side-tuning*. These images were randomly selected from the validation set. Left-hand column is input, rightmost-column is ground truth. Images from left to right show predictions as training progresses. Each block of 4 rows shows predictions on a different task (*Reshading*, *2D Edges*. *Surface Normals*.)

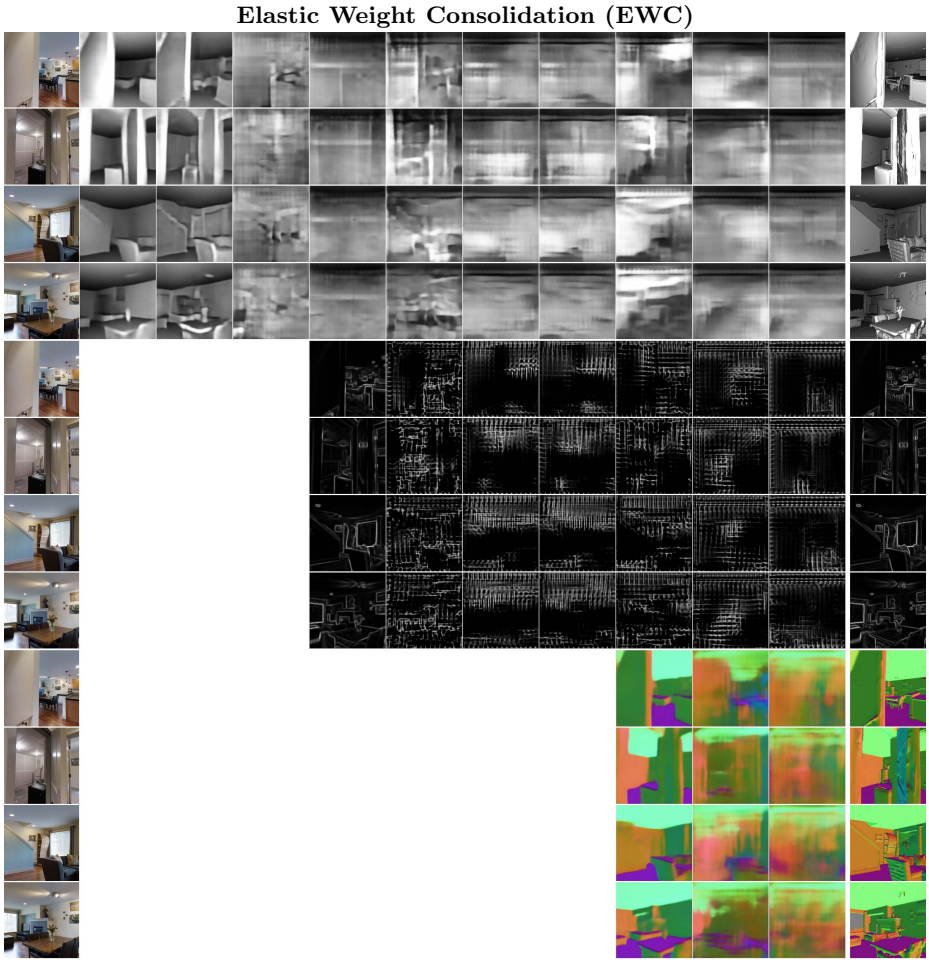

Fig. 6. More qualitative results for EWC. These images were randomly selected from the validation set. Left-hand column is input, rightmost-column is ground truth. Images from left to right show predictions as training progresses. Each block of 4 rows shows predictions on a different task (*Reshading*, *2D Edges*. *Surface Normals*.)

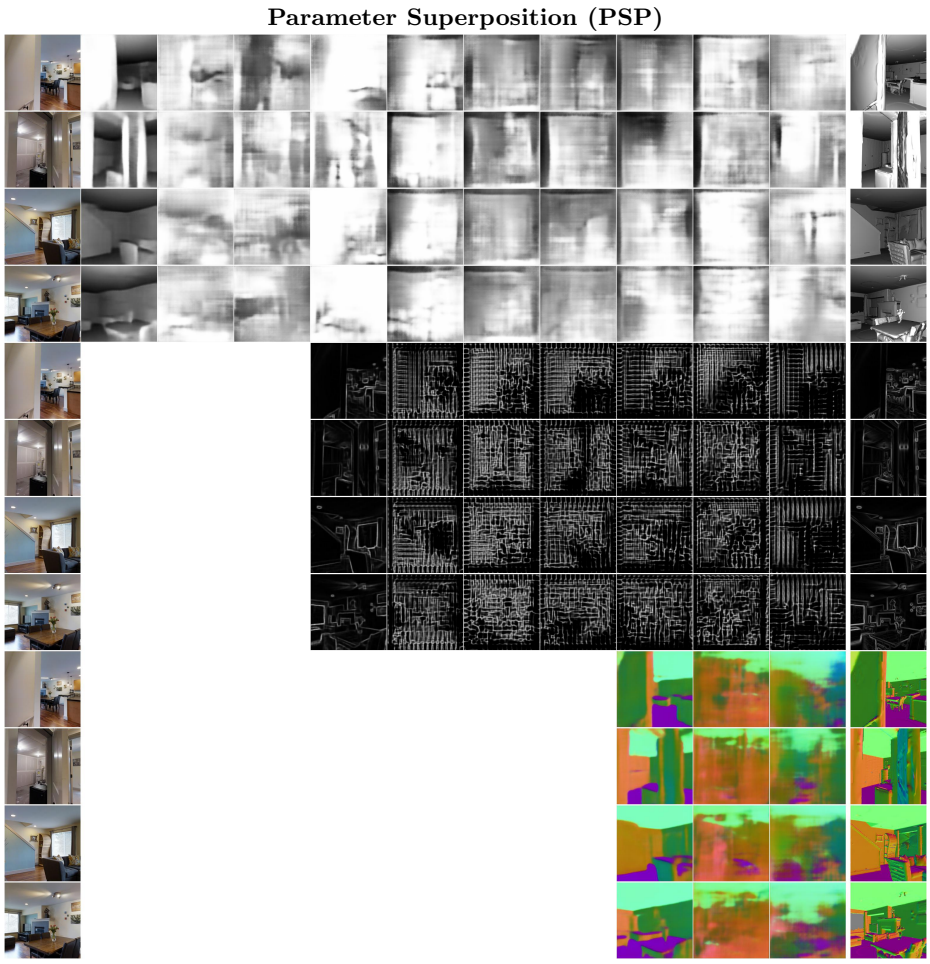

Fig. 7. More qualitative results for PSP. These images were randomly selected from the validation set. Left-hand column is input, rightmost-column is ground truth. Images from left to right show predictions as training progresses. Each block of 4 rows shows predictions on a different task (*Reshading*, *2D Edges*. *Surface Normals*.)

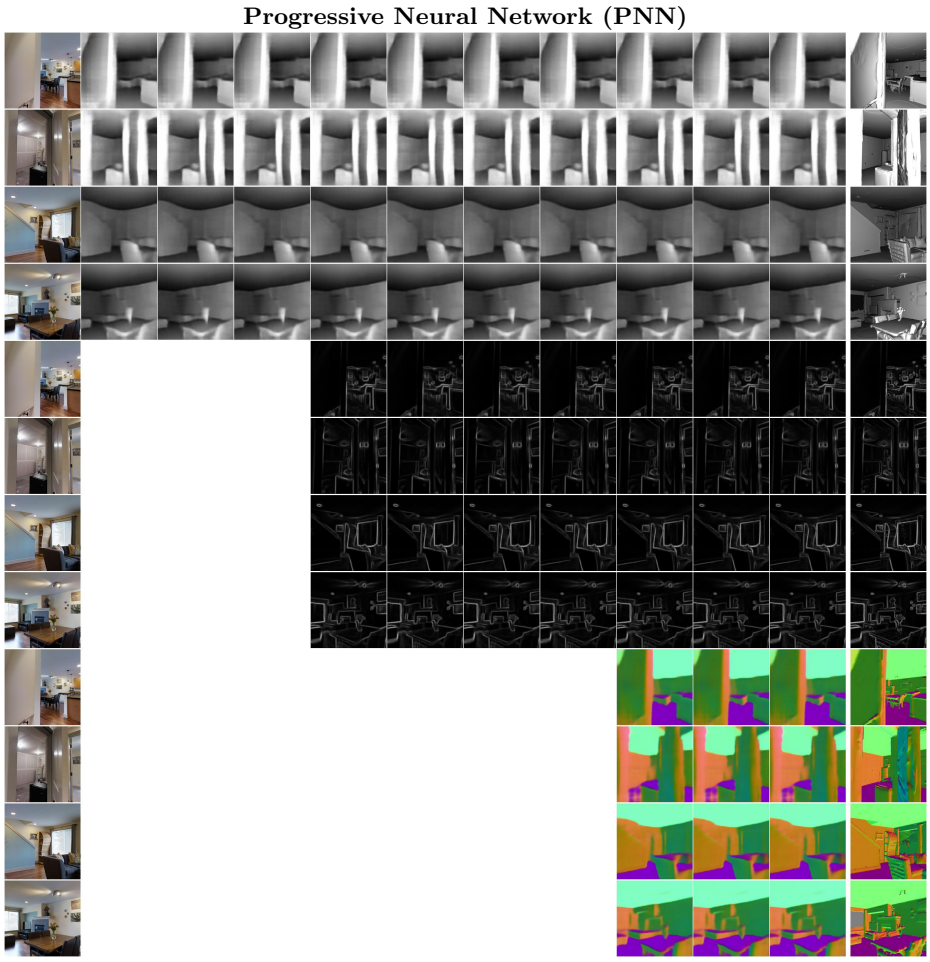

Fig. 8. More qualitative results for PNN. These images were randomly selected from the validation set. Left-hand column is input, rightmost-column is ground truth. Images from left to right show predictions as training progresses. Each block of 4 rows shows predictions on a different task (*Reshading*, *2D Edges*. *Surface Normals*.)

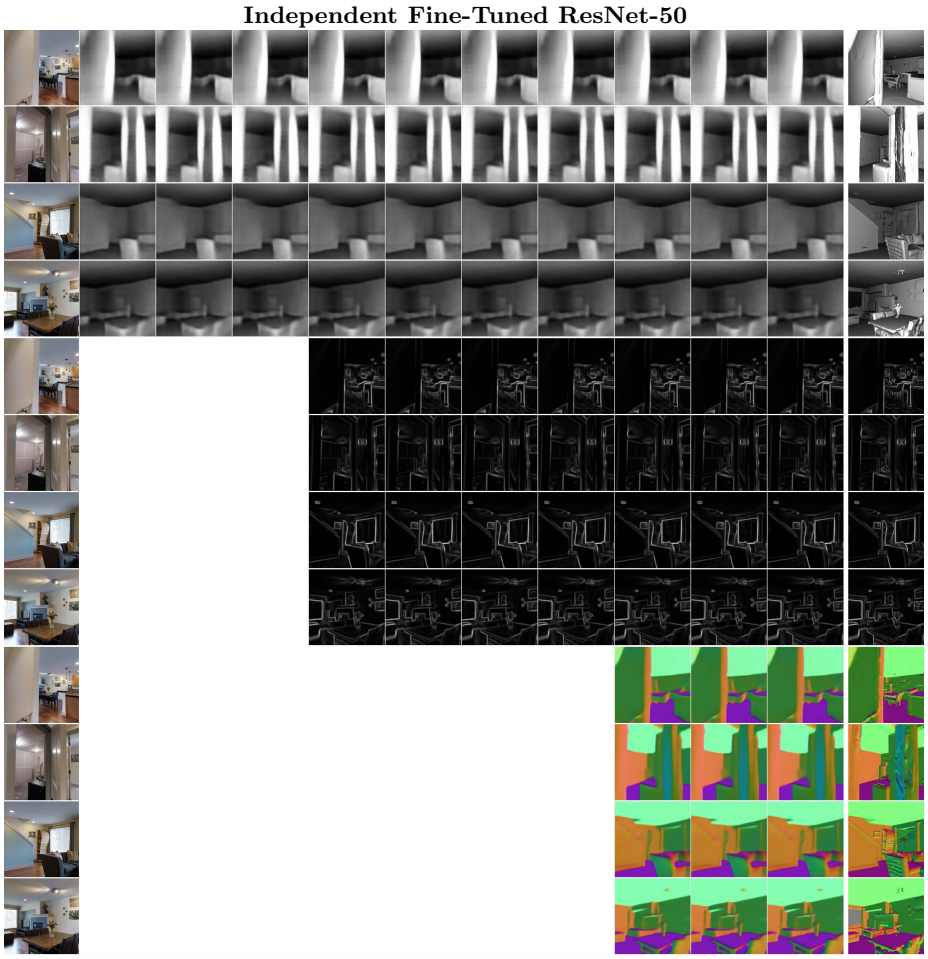

Fig. 9. More qualitative results for *independent* These images were randomly selected from the validation set. Left-hand column is input, rightmost-column is ground truth. Images from left to right show predictions as training progresses. Each block of 4 rows shows predictions on a different task (*Reshading*, *2D Edges*. *Surface Normals*.)

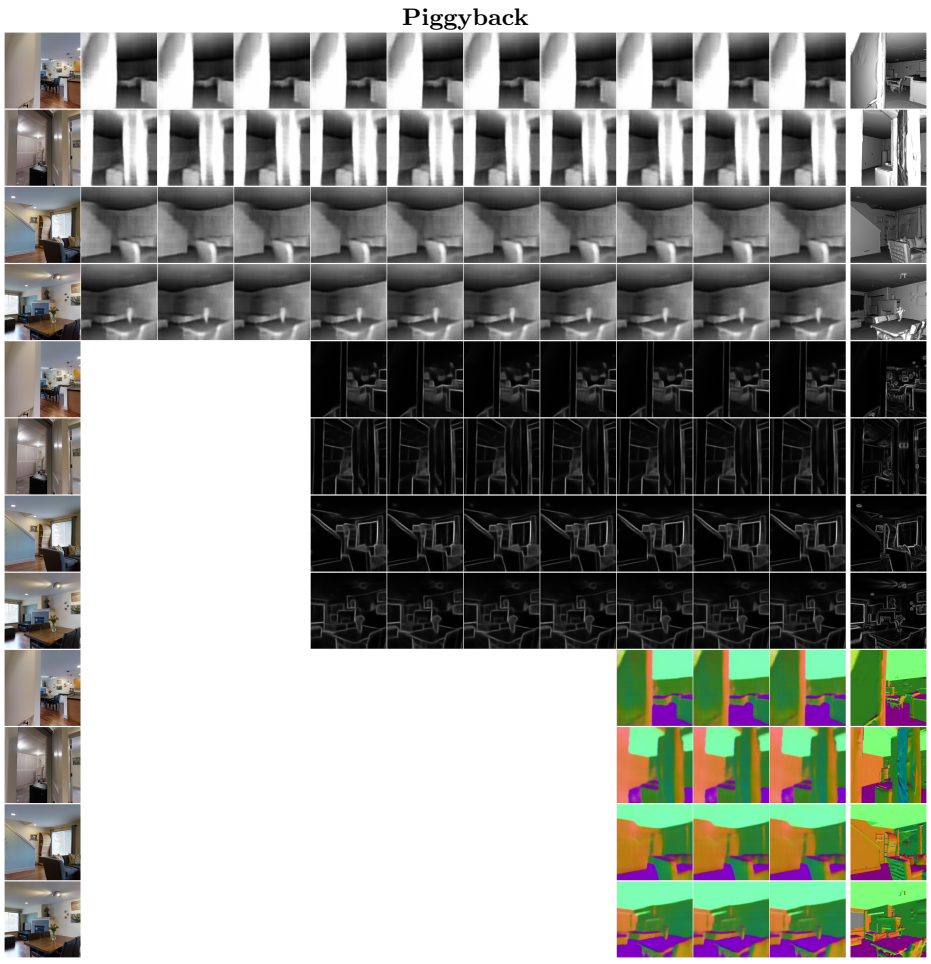

Fig. 10. More qualitative results for *piggyback* These images were randomly selected from the validation set. Left-hand column is input, rightmost-column is ground truth. Images from left to right show predictions as training progresses. Each block of 4 rows shows predictions on a different task (*Reshading*, *2D Edges*. *Surface Normals*.)

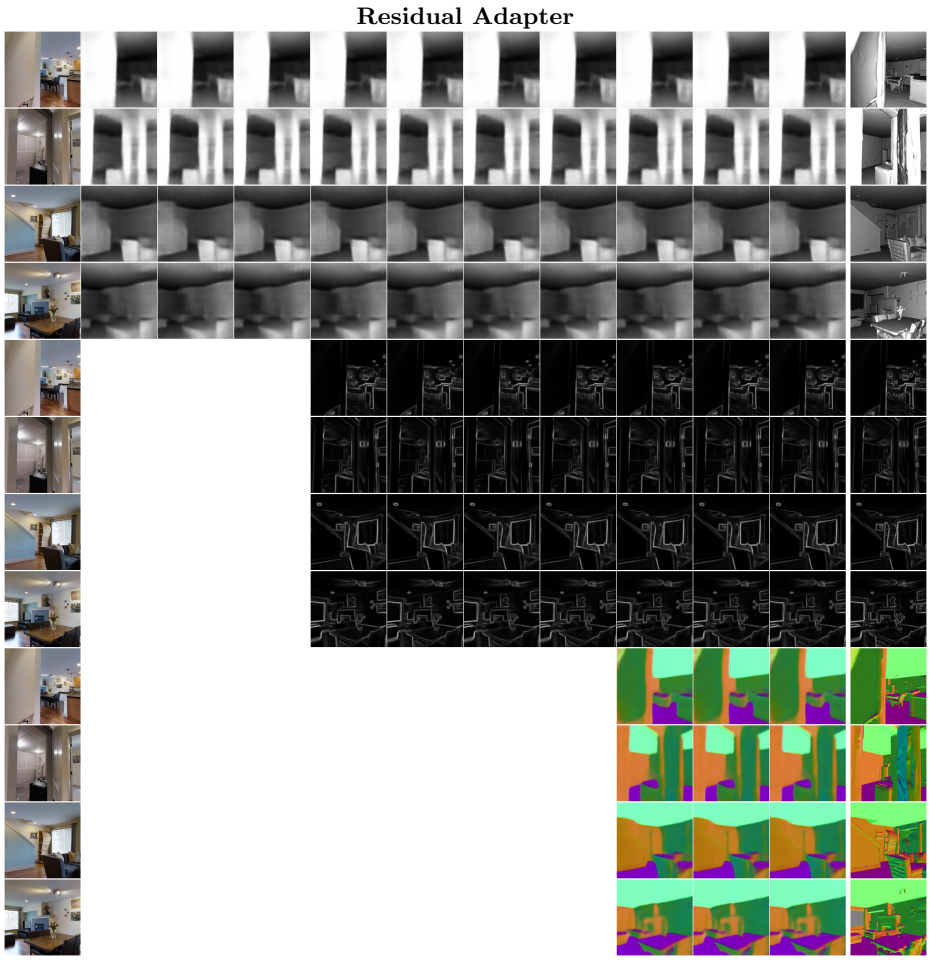

Fig. 11. More qualitative results for *Residual Adapter* These images were randomly selected from the validation set. Left-hand column is input, rightmost-column is ground truth. Images from left to right show predictions as training progresses. Each block of 4 rows shows predictions on a different task (*Reshading*, *2D Edges*. *Surface Normals*.)

References

1. Sax, A., Zhang, J.O., Emi, B., Zamir, A., Guibas, L.J., Savarese, S., Malik, J.: Learning to navigate using mid-level visual priors. (2019)
2. Zamir, A.R., Sax, A., Shen, W.B., Guibas, L.J., Malik, J., Savarese, S.: Taskonomy: Disentangling task transfer learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2018)