CURBING TASK INTERFERENCE USING REPRESENTATION SIMILARITY-GUIDED MULTI-TASK FEATURE SHARING

Naresh Kumar Gurulingan, Elahe Arani[*], Bahram Zonooz[*]
Advanced Research Lab, NavInfo Europe, The Netherlands
`{naresh.gurulingan, elahe.arani}@navinfo.eu, bahram.zonooz@gmail.com`

ABSTRACT

Multi-task learning of dense prediction tasks, by sharing both the encoder and decoder, as opposed to sharing only the encoder, provides an attractive front to increase both accuracy and computational efficiency. When the tasks are similar, sharing the decoder serves as an additional inductive bias providing more room for tasks to share complementary information among themselves. However, increased sharing exposes more parameters to task interference which likely hinders both generalization and robustness. Effective ways to curb this interference while exploiting the inductive bias of sharing the decoder remains an open challenge. To address this challenge, we propose Progressive Decoder Fusion (PDF) to progressively combine task decoders based on inter-task representation similarity. We show that this procedure leads to a multi-task network with better generalization to in-distribution and out-of-distribution data and improved robustness to adversarial attacks. Additionally, we observe that the predictions of different tasks of this multi-task network are more consistent with each other. Code is made available at github.com/NeurAI-Lab/ProgressiveDecoderFusion.

1 INTRODUCTION

Obtaining real-time predictions from neural networks is imperative for time-critical applications such as autonomous driving. These applications also require predictions from multiple tasks to shed light on varied aspects of the input scene. Multi-task networks (MTNs) can elegantly combine these two requirements by jointly predicting multiple tasks while sharing a considerable number of parameters among tasks. On the other hand, training separate single task networks could lead to different task predictions contradicting each other. Moreover, each of these networks has to be individually robust to various forms of adverse inputs such as image corruptions and adversarial attacks. MTNs include an inductive bias in the shared parameter space which encourages tasks to share complementary information to improve predictions. This information sharing also enables tasks to provide consistent predictions while holding the potential to improve robustness (Mao et al., 2020).

In an ideal scenario, this inductive bias of MTNs would lead to learning representations that generalize better and are more robust, while providing consistent predictions. In practice, however, task interference hinders the attainment of such representations (Zhao et al., 2018). Task interference is an undesired consequence of the shared parameter space enabling tasks to exchange conflicting information. One approach to reduce task interference is to modify the encoder to include isolated task-specific parameters (Liu et al., 2019) enabling tasks to encode conflicting information in these isolated parameters. Nevertheless, conflicts in the shared parameters could still occur. To eliminate task interference, other works rely on using independent subnetworks for tasks (Strezoski et al., 2019) or only update task-specific parameters with the task loss (Kanakis et al., 2020). However, Strezoski et al. (2019) do not share parameters between tasks thereby lacking the associated inductive bias of sharing while Kanakis et al. (2020) limit sharing of complementary information among tasks in the shared parameters. Alternatively, grouping tasks together based on a notion of similarity could potentially mitigate task interference as the more similar the tasks are the less likely they share conflicting information.

Intuitive notions of similarity between tasks might not necessarily hold in the feature space. Therefore, techniques to determine how tasks relate to each other at different layers of an MTN are required. Fifty et al. (2021) quantify inter-task similarity based on the effect of one task's loss when the other task's gradients are used to update shared weights. They use this similarity to group tasks into different multi-task networks. However, tasks could still require learning similar features in some layers, and in these layers, they can be combined. Determining task-sharing at a layer

[*] Equal advising.

level has not been explored much in the literature. Vandenhende et al. (2020a) use representation similarity analysis to group tasks in the encoder. They group tasks at all layers based on layerwise similarity between fully trained single task networks. However, the similarity between tasks in subsequent layers is likely to change once they have been combined at a certain layer.

Therefore, to group tasks at a certain layer, it could be necessary to compare representation similarities in light of how the tasks have been grouped in the previous layer. We propose an algorithm that only groups tasks at a layer based on a fully trained network with grouping done at the previous layer. Specifically, we propose Progressive Decoder Fusion (PDF) algorithm which alternates between fusing tasks at a decoder stage and retraining the fused network. Essentially, the PDF algorithm revolves around the question:

How to effectively determine a sharing scheme in the decoder to best curb task interference while benefiting from complementary information sharing?

To test the strengths of the proposed algorithm, we evaluate the final decoder sharing scheme under different settings such as generalization to in-distribution and out-of-distribution data and robustness to adversarial attacks. We observe that the obtained decoder sharing scheme leads to improvements in generalization, robustness, and inference speed while requiring less number of parameters. Our contributions include:

- Progressive Decoder Fusion (PDF) algorithm which groups tasks in a layer considering how tasks have been grouped in previous layers.
- Comprehensive evaluation of the final network obtained with the PDF algorithm to test its robustness to adversarial attacks and generalization to in-distribution and out-of-distribution data.
- Exploring the role of different tasks with regards to the consistency between predictions, generalization, and robustness.

2 Related Works

Progress in multi-task learning (MTL) has come from varied directions including custom architectures (Section 2.1), task balancing (Section 2.2) and task grouping (Section 2.3). Since the primary goal of this work is to curb task interference in MTL, we probe works in each of these categories and discuss how they address task interference.

2.1 Custom Architectures

One way to alleviate task interference is to introduce task-specific parameters in the shared encoder. This modification could enable the network to encode task information that might conflict with other tasks in the task-specific parameters. In MTAN (Liu et al., 2019), each task is equipped with its own attention modules at different stages of the encoder. Kanakis et al. (2020) use task-specific 1×1 convolutions after each convolution in the encoder. Only these 1×1 convolutions are trained with the task gradients to explicitly avoid task interference. Strezoski et al. (2019) propose task-specific routing to create randomly initialized task-specific subnetworks to reduce interference. Sun et al. (2020) propose to learn task-specific policies with a sharing objective and a sparsity objective to balance the number of ResNet blocks shared between tasks and task interference. These methods likely reduce conflicts on the shared parameter space. However, they require architecture modifications in the encoder and could require training on ImageNet to initialize weights.

2.2 Task balancing

Task interference can be seen as the consequence of conflicting task gradient directions in the shared parameters. Task gradients can be modified such that the disagreement between them is reduced to mitigate task interference. The PCGrad (Yu et al., 2020) uses cosine similarity to identify different pairs of task gradients with contradicting directions in each shared parameter. In each of these pairs, one of the gradients is projected onto the normal vector of the other to reduce conflict. Chen et al. (2020) reduce the probability of using negative task gradients during backward pass thereby reducing gradient conflict. As gradients are modified, these approaches might lose task-specific information.

Individual task losses can be weighted in different ways to address the variance in loss scales (Liu et al., 2019; Kendall et al., 2018; Guo et al., 2018; Chen et al., 2018). These methods primarily attempt to avoid certain tasks from dominating gradient updates. However, they can only be viewed as a means to loosely modulate task interference as they affect the extent to which tasks affect the shared parameters. Our approach can be used in tandem with task balancing techniques.

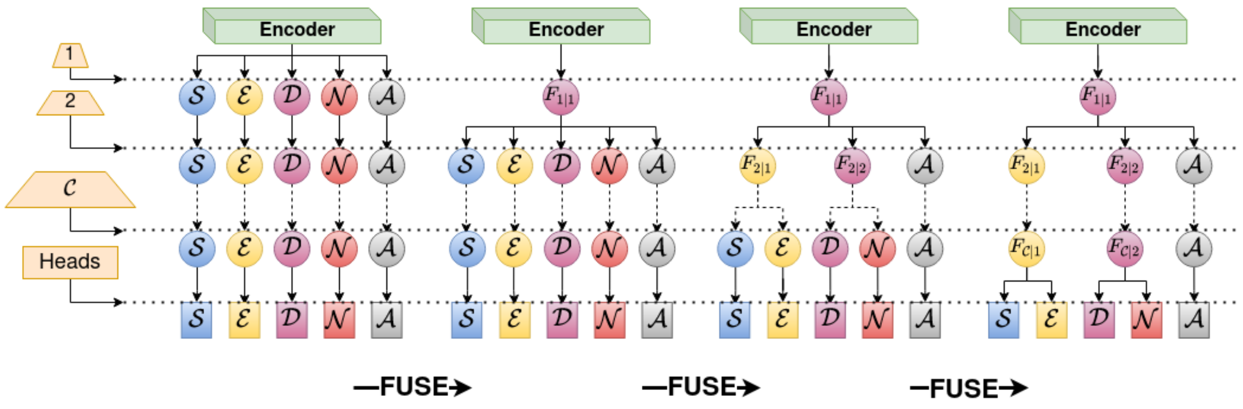

Figure 1: Fusing decoders at different candidate stages. From left to right decoder fusion at candidate stages 1, 2 and $\mathcal{C}$ is shown. $\mathcal{S}, \mathcal{E}, \mathcal{D}, \mathcal{N}$ and $\mathcal{A}$ denote different tasks, namely semantic segmentation, edge detection, depth estimation, surface normals and autoencoder. The dotted horizontal lines show the candidate decoder stages and the task-specific heads. The lines connecting decoder stages in 2 and $\mathcal{C}$ are dotted to show that there are more stages in between. The task-specific heads are not fused in any of the approaches. $\mathcal{F}$ denotes fused decoder and $\mathcal{F}_{i|j}$ denotes i^{th} candidate stage of j^{th} task decoder.

2.3 Task grouping

If only similar tasks are grouped together either at the network level (each task group split into separate networks) or at a layer level (across the network depth), task interference can be reduced. Standley et al. (2020) train several MTNs created with all possible combinations of tasks. They pick the combination with the lowest total loss across tasks under a certain computation budget as the desired MTN. Fifty et al. (2021) quantify affinity of task i on task j based on the relative change of task j's loss before and after updating shared parameters with task i. This affinity between different task pairs is accumulated throughout the training and tasks are grouped into different networks such that the overall affinity is maximized.

Other approaches restrict all the tasks to remain in the same network and group them layerwise. Grouping layerwise reduces the required computation but task interference could increase. Guo et al. (2020) use an automated approach based on Gumbel-Softmax sampling of connections between a child layer and a number of parent layers in a topology. After training, at every child layer, only the connection to the parent layer with the highest probability is retained leading to the effect of tasks being separated at a certain layer. This separation reduces task interference. Vandenhende et al. (2020a) use similarity between representations of trained single task networks to determine how to group tasks at different stages in the encoder. Unlike Vandenhende et al. (2020a), we study task grouping in the decoder and propose grouping tasks in a progressive fashion.

3 Sharing task decoders while curbing task interference

The inductive bias of sharing parameters among different tasks is intuitively desirable as it enables tasks to share complementary information. The early layers learn general features and as we progress deeper through the network, the features learnt become increasingly task-specific. There is often no clear indication as to where in the network the transition between generic to task-specific features happens. In dense prediction tasks, the representations learnt in the decoder for similar tasks might only diverge and become task-specific in the later layers. Most of the early decoder layers could likely be shared among tasks while providing improved generalization, improved robustness, and reduced computational overhead.

In the upcoming sections, we formally define the problem (Section 3.1), go over the similarity measure we use (Section 3.2), and discuss our proposed algorithm (Section 3.3 to arrive at an effective decoder sharing scheme.

3.1 Problem Setup

Given a set of $\mathcal{T}$ tasks with each having its own decoder $\mathbb{D}_t$, we intend to combine the decoders while incurring limited task interference. All the decoders have the same architecture. We consider $\mathcal{C}$ candidate stages $\{\mathbb{D}_{1|t}, \mathbb{D}_{2|t}, ..., \mathbb{D}_{\mathcal{C}|t}\}$,

$\forall t \in \{\mathcal{T}\}$ where the task decoders can be combined. We refer to combining decoders as fusion. For ease of reading, we assign a letter to each task and use this letter to denote the task decoder. The tasks involved and their associated decoders are semantic segmentation: $\mathbb{D}_{1:\mathcal{C}|\mathcal{S}}$, depth estimation: $\mathbb{D}_{1:\mathcal{C}|\mathcal{D}}$, edge detection: $\mathbb{D}_{1:\mathcal{C}|\mathcal{E}}$, surface normals: $\mathbb{D}_{1:\mathcal{C}|\mathcal{N}}$ and autoencoder: $\mathbb{D}_{1:\mathcal{C}|\mathcal{A}}$. Figure 1 depicts three different decoder fusions done at candidate stages 1, 2 and $\mathcal{C}$.

Algorithm 1: Progressive Decoder Fusion

```
Initialize network with each task having a separate decoder,
  i.e., $\mathbb{D}_{1:\mathcal{C}|\mathcal{S}} \neq \mathbb{D}_{1:\mathcal{C}|\mathcal{D}} \neq \mathbb{D}_{1:\mathcal{C}|\mathcal{E}} \neq \mathbb{D}_{1:\mathcal{C}|\mathcal{N}} \neq \mathbb{D}_{1:\mathcal{C}|\mathcal{A}}$;
Task set $\mathcal{T} \leftarrow \{\mathcal{S}, \mathcal{D}, \mathcal{E}, \mathcal{N}, \mathcal{A}\}$;
Candidate stage $c \leftarrow 0$ (last encoder stage);
Train network until convergence;
for $c \in \{1, ..., \mathcal{C}\}$ do
    $g \leftarrow$ number of fused groups in stage $c-1$;
    for fused group $\{\mathcal{F}_{c-1|1}, ..., \mathcal{F}_{c-1|g}\}$ do
        $t \leftarrow$ number of tasks branching from $g$;
        if $t = 1$ then
            Continue to next $g$;
        else
            Measure $t \times t$ CKA similarity matrix;
            Run Grouping Algorithm 2;
        end
    end
    Retrain the fused network until convergence;
end
```

Algorithm 2: Grouping

```
Input: Task set $T$ and $T \times T$ similarity
  matrix $S$;
Group $G \leftarrow$ {tasks};
Grouping $\mathcal{G} \leftarrow$ {groups};
Task value in group $G$ is $\mathcal{V}_t \leftarrow \frac{1}{i}\sum_i S_{ti}$,
  where $i \subset G \backslash t$, $S_{ti}$ is the similarity
  between task t and i;
Group value $\mathcal{V}_G \leftarrow \frac{1}{t}\sum_t \mathcal{V}_t$, where t is the
  #tasks in group ;
Unique Grouping $\mathbb{G} \subset \mathcal{G}$, such that all
  tasks are present exactly once;
for all unique groupings do
    $\mathcal{V}_{\mathbb{G}} \leftarrow \frac{1}{g}\sum_g \mathcal{V}_g$, where g is the
      #groups in grouping;
end
Final grouping $\leftarrow$ grouping with
  maximum $\mathcal{V}_{\mathbb{G}}$;
Result: Final grouping
```

3.2 Representation similarity

In MTL, a well accepted notion is that jointly learning similar tasks would improve overall performance. This notion is intuitively well placed as similar tasks would need similar feature representations which can be attained by sharing parameters instead of using individual networks. However, this notion does not necessarily have to be only at the network level (each task group split into separate networks) but can also be used to combine tasks at the layer level (across the network depth) (Guo et al., 2020; Vandenhende et al., 2020a).

Two tasks can be combined at a particular candidate stage of the MTN if they require learning similar representations at that stage. Central Kernel Alignment (CKA) is a similarity metric with desirable properties such as invariance to orthogonal transformations and isotropic scaling enabling meaningful comparisons of learnt representations (Kornblith et al., 2019). Further details regarding CKA is provided in the appendix Section A.2. At a particular decoder stage, we quantify the pairwise similarity between the representations of all task decoders using CKA. Specifically, given decoder activations $\{\mathbb{D}_{c|1}, ..., \mathbb{D}_{c|\mathcal{T}}\}$ at candidate stage c, we construct a pairwise CKA similarity matrix of shape $\mathcal{T} \times \mathcal{T}$ where each element represents similarity between the tasks corresponding to the row and column. Since CKA is symmetric, i.e., CKA($\mathbb{D}_{c|i}$, $\mathbb{D}_{c|j}$) = CKA($\mathbb{D}_{c|j}$, $\mathbb{D}_{c|i}$), the $\mathcal{T} \times \mathcal{T}$ similarity matrix is also symmetric. This pairwise similarity matrix is used for task grouping in the PDF algorithm (Section 3.3).

3.3 Similarity guided Progressive Decoder Fusion

In Section 3.2, we saw that CKA can be used to quantify the similarity between two learned representations. Equipped with this, we now look at ways to arrive at a decoder sharing scheme that provides good generalization and robustness.

We first train a network where each of the tasks has its own decoder and calculate the similarity of learned representations at the first candidate stage of the decoder using a subset of training images. With the similarity scores, we group tasks at the first decoder stage using a grouping algorithm. Essentially, this grouping algorithm lists all possible task groupings and identifies a set of groups that cover all tasks exactly once such that the overall similarity is maximized (details in Algorithm 2). This new model is reinitialized[1]. We repeat this process for all decoder stages unless each task branched separately. This approach is outlined in Algorithm 1.

[1]While reinitializing, the parameters are reset to the original intialization where the encoder is initialized with ImageNet pre-trained weights while the decoder(s) and task heads are randomly initialized. and retrained after which grouping is done at the second decoder stage among tasks which grouped together in the first stage

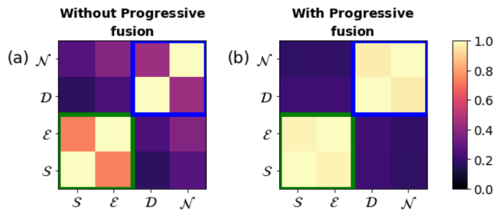

Figure 2: Similarity change between tasks at decoder stage 4 without (a) and with progressive fusion (b). Task similarities within both the green and blue boxes differ but within the blue box there is considerable difference.

Unlike Vandenhende et al. (2020a), we do not group tasks across all decoder stages using the initial network where each task has its own decoder. This difference stems primarily from our experimental observation that learned representations at subsequent candidate stages change once the previous stage is fused as is illustrated in Figure 2. In (a), the task similarities are determined using the initial network while (b) show task similarities with progressive fusion. We see that the similarities have considerably changed. Thus, fusing decoder parameters at all candidate stages based only on the similarity measure between individual decoders could result in sub-optimal solutions. We present more empirical evidence for change in the similarity between tasks once previous stage decoders are fused in Section 5.2.

4 Role of different tasks

The tasks in an MTN can share complementary information with each other to gain performance improvements due to the inductive bias of shared parameter space. Auxiliary tasks act as a source of complementary information to help improve the performance of main tasks (Xu et al., 2018). For the task of edge detection, we only consider the delineating edges between regions segmented by semantic segmentation. Essentially, the edge detection task is a binary pixel-wise classification task that predicts the delineating pixels in segmentation ground truth. Thus, semantic segmentation and edge detection are related semantically. On the other hand, depth estimation and surface normals share a geometric relationship (Qi et al., 2018). These pairs of tasks could potentially share complementary information and could lead to improved performance overall (Xu et al., 2018).

Since it is possible to transform semantic segmentation predictions to edge detection and depth estimation predictions to surface normals, we measure two prediction consistencies. We transform semantic segmentation to edge detection and measure semantic consistency as the number of pixels matching between transformed segmentation predictions and edge predictions relative to the total number of pixels (pixel accuracy). Likewise, we measure geometric consistency as the mean of all per pixel cosine similarities between transformed depth predictions and surface normal predictions. The obtained prediction consistencies are presented and discussed in Section 5.4.

The four tasks which we addressed so far provide diverse predictions. This diversity could in and of itself aid in learning a shared representation that exhibits improved generalization as it needs to address these diverse predictions. On top of these tasks, we also include an autoencoder to force the shared representation to not just rely on features that aid in-distribution performance. We then analyze the role of the autoencoder and evaluate whether it aids in generalization and robustness improvements in Section 5.5.

5 Experiments

To demonstrate the strengths of the PDF algorithm, we present experimental results obtained with UniNet (Gurulingan et al., 2021) architecture. To study fusion in the various stages of the decoder, we pick a hard parameter sharing architecture that shares the entire encoder with all tasks. We consider UniNet which has five different decoder stages branching from the encoder. We use the implementation provided by Fifty et al. (2021) for obtaining the best possible task grouping at a decoder stage using CKA similarity.

We use two datasets for experimentation. **Cityscapes** (Cordts et al., 2016) is a driving scenes dataset consisting of images from various European cities. The dataset consists of 2975 training images and 500 validation images with a resolution of 1024×2048. For training and validation, we use an input resolution of 512×1024. **NYUv2** (Silberman et al., 2012) consists of indoor scene images of resolution 480×640. There are 795 training and 654 validation images. To calculate CKA similarities, we use all the training images and 800 training images for the NYUv2 and Cityscapes datasets, respectively. For both datasets, all numbers are reported on the validation set. CS and NYU denote to the Cityscapes and NYUv2 datasets, respectively.

We refer to complementary information sharing between tasks as *positive transfer*. Also, we collectively refer to inference speed and parameter count of a network as *inference efficiency*. An inference efficient network has high

Table 1: Generalization results on in-distribution and OOD (four different natural corruption categories: noise, blur, weather, and digital). # (M) denotes parameter count in millions. In majority of the cases, Sep-De achieves better generalization than One-De.

	Network	$\mathcal{S}$ (mIoU %)↑					$\mathcal{D}$ (RMSE)↓					FPS	# (M)
		IID	noise	blur	weather	digital	IID	noise	blur	weather	digital		
CS	One-De	72.11	20.28	48.45	35.86	60.29	5.36	**15.48**	13.20	**11.82**	7.60	**32.08**	**51.56**
CS	Sep-De	**72.72**	**20.69**	**49.47**	**35.95**	**61.39**	**5.33**	16.39	**12.95**	11.93	**7.20**	22.31	57.03
NYU	One-De	41.79	**5.43**	23.34	19.04	26.12	47.98	**112.45**	**83.01**	82.11	67.51	**50.70**	**51.57**
NYU	Sep-De	**41.99**	4.86	**23.53**	19.04	**26.70**	**47.68**	115.36	85.74	**79.75**	**64.69**	31.24	57.04

Table 2: Robustness to PGD attack results reported as average over low and high ϵ bounds. In majority of the cases, Sep-De achieves more robustness over One-De.

Network	CS				NYU			
	$\mathcal{S}$ (mIoU %)↑		$\mathcal{D}$ (-RMSE)↓		$\mathcal{S}$ (mIoU %)↑		$\mathcal{D}$ (-RMSE)↓	
	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ
One-De	45.04	6.91	**15.16**	46.32	19.34	2.73	110.86	309.56
Sep-De	**45.95**	**7.92**	15.24	**45.14**	**20.36**	**3.29**	**108.07**	**303.61**

inference speed and low parameter count. We quantify the generalization of all models to in-distribution validation set (IID) and out-of-distribution datasets (OOD) obtained using natural image corruptions Hendrycks & Dietterich (2019). The OOD numbers are reported for four corruption types (**noise, blur, weather, digital**) as the average across 5 severity levels and all corruptions belonging to the corresponding type. We also quantify the robustness of these models to adversarial attacks with the help of Projected Gradient Descent (PGD) attack conducted using a step size of 1 and for $\min(\epsilon + 4, \lceil 1.25\epsilon \rceil)$ iterations (Kurakin et al., 2017). PGD numbers are reported for **low ϵ** (average across ϵ bounds $\{0.25, 0.5, 1\}$) and **high ϵ** (average across ϵ bounds $\{4, 8\}$) by using the corresponding task loss as the attack objective. All the numbers are reported as the mean of 3 random seed runs.

We train all the models for 140 epochs using the RAdam optimizer with a learning rate of 0.0001 and stepwise learning rate schedule with a learning rate reduced by a factor of 10 at epochs 98 and 126. The batch size used is 8. The same hyperparameters are used for both datasets. We consider two baselines, one in which each task has its own decoder and another where all tasks use one decoder. Based on these baselines, we establish the need to find a better decoder sharing scheme in Section 5.1. We then motivate the need for the PDF algorithm (Section 5.2) and show how it aids in generalization and robustness (Section 5.3). Section 4 demonstrates how the baselines and different ways of grouping affect the consistency between task predictions. Finally in Section 5.5, we analyze the role of autoencoder.

5.1 Baselines

Since we take the setting where the encoder is always shared between all tasks, the baselines are only based on the decoder. We consider two baselines, the network with all tasks having their own decoder (referred as **Sep-De**) and the network with one decoder (referred as **One-De**) for all tasks. Since Sep-De has no shared decoder parameters, there is no task interference in the decoder. However, there is no positive transfer between tasks in the decoder and the overall inference efficiency is reduced. On the other hand, One-De shares all decoder parameters among tasks and therefore could have high task interference. However, One-De has the best inference efficiency and could also leverage positive transfer. An ideal decoder sharing scheme would surpass the performance of Sep-De by curbing task interference and leveraging positive transfer while rivaling One-De in inference efficiency.

Table 1 presents the results obtained with both the baselines on the Cityscapes and the NYUv2 datasets. We report only the segmentation (S) and depth (D) results for brevity and because these tasks are the main tasks of concern. Except for a few cases, we see that Sep-De always results in a performance gain relative to One-De. We attribute this gain to the absence of task interference between all tasks in the decoder parameter space. However, this gain in performance of Sep-De comes at the cost of reduced inference efficiency as is evident from the slower FPS and higher parameter count in comparison to One-De. In Table 2, we see that Sep-De also achieves better robustness to PGD attack in majority of the cases. The PDF algorithm intends to retain or improve over the generalization and robustness gain of Sep-De while finding a balance in inference efficiency.

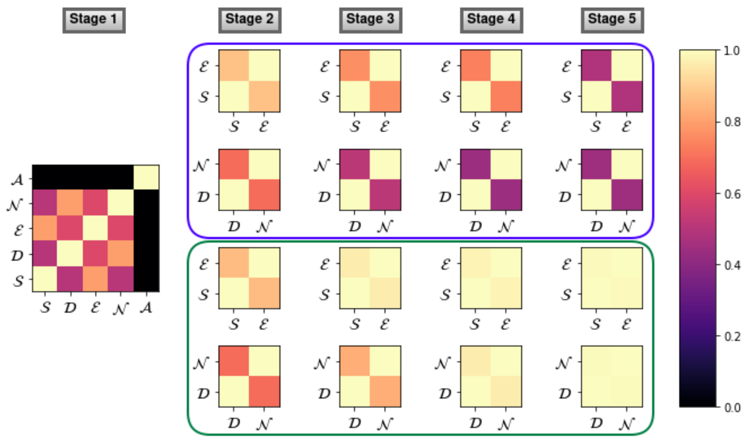

Figure 3: CKA similarity matrix at different decoder stages obtained from offline (blue box) and progressive (green box) grouping on Cityscapes dataset. As we move towards the right, decoder representations become increasingly similar with progressive grouping but increasingly dissimilar with offline grouping.

5.2 Why Progressive Decoder Fusion?

In Section 5.1, we established the baselines and saw that Sep-De results in a performance gain. To build upon this performance gain by leveraging positive transfer and increasing inference efficiency, we intend to curb task interference by only sharing similar decoder representations. Based on the 5×5 CKA similarity matrix, we group the tasks at candidate stage 1. The grouping algorithm (detailed in Algorithm 2) first lists all possible combinations of task groupings. Out of all these combinations, the one which has the highest value defined by the average similarity across all tasks in the grouping is picked. For instance in the Cityscapes dataset, the best grouping obtained using Sep-De is [$\mathcal{S} = 0.86$, $\mathcal{E} = 0.86$], [$\mathcal{D} = 0.76$, $\mathcal{N} = 0.76$], [$\mathcal{A} = 0$] with overall value of 0.65. In the picked grouping, if a two-task group has a value of less than 0.5, the tasks in the group are split back into their own branches. After grouping tasks in candidate stage 1, we retrain this new network until convergence. In the next stage, grouping is only done based on the $t \times t$ for the t tasks sharing the same group in the previous stage. We continue this process until grouping is done in all stages.

The changes in the similarity between tasks are depicted by visualizing the CKA similarity matrix in Figure 3 for the Cityscapes dataset. The leftmost plot shows the 5×5 pairwise similarity across the 5 tasks in Sep-De. In offline grouping, tasks are fused at all stages based on these inter-task similarities. This grouping is similar to how Vandenhende et al. (2020a) group tasks in the encoder. In offline grouping (blue box), at stage 2, the two task groups picked based on stage 1 similarities are [$\mathcal{S}$, $\mathcal{E}$] and [$\mathcal{D}$, $\mathcal{N}$]. The similarity between the same task groups is visualized across the rest of the stages. These results show that in offline grouping, tasks become increasingly dissimilar across stages (from left to right).

On the other hand, in progressive grouping (green box), the tasks become increasingly similar across stages. This observation shows that once tasks are grouped at a stage, the similarity between grouped tasks in the subsequent stages increases. This effect supports the hypothesis that task similarities change based on where they branch in the network. However, offline grouping does not consider this effect leading to groupings at stages 2 to 5 which might not be ideal. This possibility serves as a motivation to set up and evaluate the PDF algorithm. We train a network with the decoder sharing scheme obtained with offline grouping, referred to as **Offline** and compare with **PDF** which is the final model obtained with the PDF algorithm in Section 5.3.

5.3 Generalization and Robustness

Results in Section 5.2 suggest that grouping task decoders progressively at each stage could result in a decoder sharing scheme that generalizes better and is also more robust. This expectation stems from the nature of the PDF algorithm which only groups similar tasks at every candidate stage. Since only similar tasks are grouped, positive transfer could improve while reducing task interference leading to richer representations that leverage the strengths of each task.

The groupings done at different decoder stages using progressive and offline grouping for both datasets are provided in Table 6 in Appendix. For Cityscapes in stage 1, the task groups are [$\mathcal{S}$, $\mathcal{E}$], [$\mathcal{D}$, $\mathcal{N}$], [$\mathcal{A}$]. This grouping indicates that the tasks $\mathcal{S}$, $\mathcal{E}$ are to be combined in stage 1. Likewise, $\mathcal{D}$, $\mathcal{N}$ are combined while $\mathcal{A}$ has its own decoder. We see that the grouping at stages 4 and 5 differ between PDF algorithm and offline grouping. Therefore, these two ways of grouping result in different final models. In NYUv2 the groupings differ right from stage 1.

Table 3: Generalization and robustness results of Sep-De, Offline and PDF networks. Results of IID and OOD (noise, blue, weather, and digital) generalization and robustness (low ϵ and high ϵ) are reported as percentage improvements over One-De. # (M) denotes parameter count in millions. PDF network outperforms Offline network in most of the cases while being more inference efficient.

	Network	IID	noise	blur	weather	digital	low ϵ	high ϵ	FPS	# (M)
CS	Sep-De	0.66	-1.90	2.00	-0.35	3.57	0.75	8.62	22.31	57.03
CS	Offline	0.71	-3.54	0.28	-0.26	**2.05**	**0.73**	**10.35**	24.19	54.57
CS	PDF	**1.00**	**-0.10**	**1.36**	**-0.01**	1.13	0.70	9.97	**25.44**	**54.30**
NYU	Sep-De	0.55	-6.51	-1.24	1.43	3.19	3.89	11.13	31.24	57.04
NYU	Offline	0.80	-5.95	**1.93**	0.48	1.61	2.96	15.05	33.02	55.86
NYU	PDF	**1.38**	**-4.04**	1.46	**1.88**	**3.62**	**4.11**	**16.16**	**34.74**	**55.67**

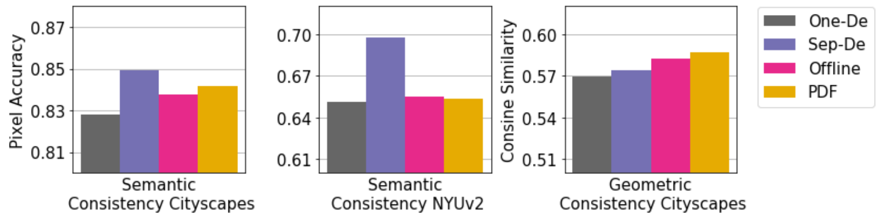

Figure 4: Semantic and geometric consistency results. PDF network provides more consistent predictions than the offline network except in NYUv2 semantic consistency. Compared to Sep-De, PDF results in higher consistency only in cityscapes geometric consistency.

Table 3 details the generalization and robustness results of PDF and Offline networks. The numbers are reported as the average relative improvement of the segmentation and depth tasks over One-De using Equation 1. [2]

$$\frac{1}{2} * \left(\frac{(\mathcal{S}_{net} - \mathcal{S}_{\text{One-De}})}{\mathcal{S}_{\text{One-De}}} - \frac{(\mathcal{D}_{net} - \mathcal{D}_{\text{One-De}})}{\mathcal{D}_{\text{One-De}}} \right) * 100, \quad \text{where } net \in \{\text{Sep-De}, \text{PDF}, \text{Offline}\} \tag{1}$$

The PDF network surpasses the Offline network in a majority of cases. These results support the PDF algorithm and show that changing grouping decisions in a stage based on a fully trained network with tasks grouped up to the previous stage is helpful. We also see that PDF is more inference efficient than the Offline network as is evident from the FPS and number of parameters. Overall, the decoder sharing scheme of PDF mostly achieves better generalization and robustness than the Offline network while being more inference efficient.

5.4 INTER-TASK PREDICTION CONSISTENCY

The consistency between predictions of different tasks could be seen as an indicator of task interference. When the predictions are consistent, tasks could have shared more complementary information rather than conflicting information. Thus, higher prediction consistency may indicate reduced task interference. We explore this possibility and look at the consistency between two sets of predictions.

The consistencies of different models are visualized in Figure 4[3]. In Cityscapes, in both semantic and geometric consistency, PDF provides more consistent predictions compared to the Offline network suggesting that it has leveraged positive transfer. However, in NYUv2 semantic consistency, task interference likely hasn't been reduced enough as PDF falls slightly behind the Offline network. PDF falls behind Sep-De in semantic consistency in both datasets suggesting that there is room to further curb task interference. These consistencies likely only provide weak evidence for task interference and further evaluations would be required to draw insights from consistency.

[2]Relative depth RMSE is subtracted as lower depth RMSE is better.

[3]We do not include NYUv2 geometric consistency as we have used the precomputed surface normals for training and have not reproduced computation of normals from depth using the NYUv2 toolbox.

Table 4: The IID and OOD (noise, blue, weather, and digital) generalization results of PDF network with and without autoencoder ($\mathcal{A}$). Overall, autoencoder results in generalization improvements except in Cityscapes depth.

	Network	$\mathcal{S}$ (mIoU %)↑					$\mathcal{D}$ (RMSE) ↓				
		IID	noise	blur	weather	digital	IID	noise	blur	weather	digital
CS	PDF	**72.92**	**19.72**	**49.19**	36.68	**60.98**	**5.31**	15.09	13.04	12.09	7.52
	PDF-No-$\mathcal{A}$	72.17	18.56	48.76	**36.72**	60.88	5.32	**13.84**	**12.99**	**11.91**	**7.29**
NYU	PDF	**42.28**	5.04	**24.27**	18.83	**27.14**	**47.22**	**113.53**	**83.89**	**78.14**	**65.23**
	PDF-No-$\mathcal{A}$	42.09	**5.49**	23.98	**19.01**	26.79	47.39	134.45	86.86	80.31	65.31

Table 5: Robustness to PGD attack results reported as average over low and high ϵ bounds for PDF network with and without autoencoder ($\mathcal{A}$). Overall, autoencoder results in robustness improvements.

Network	CS				NYU			
	$\mathcal{S}$ (mIoU %) ↑		$\mathcal{D}$ (RMSE) ↓		$\mathcal{S}$ (mIoU %) ↑		$\mathcal{D}$ (RMSE) ↓	
	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ
PDF	**45.75**	8.07	**15.19**	44.93	**20.38**	**3.58**	**107.71**	**304.94**
PDF-No-$\mathcal{A}$	45.65	**8.29**	15.29	**44.82**	20.34	3.56	121.67	323.66

5.5 Role of Autoencoder

The autoencoder learns to reconstruct the input image with the aid of an MSE loss and would require learning a different set of features that takes into account the varied aspects of the input scene. One evidence for this requirement is seen in the CKA similarity matrix obtained with Sep-De in Figure 3 (leftmost heatmap). The autoencoder feature representations at decoder stage 1 are different from all other task representations in both datasets. Likewise, a different set of features could also be learnt in the encoder. These different features enforced by the autoencoder are likely to be more general to facilitate image reconstruction. Since all tasks branch from a potentially more general encoder representation, the generalization and robustness of tasks could have improved.

We evaluate the effect of the autoencoder by comparing the generalization and robustness of the PDF network with and without the autoencoder. Table 4 shows that adding an autoencoder generally improves or retains the generalization except in Cityscapes depth estimation. Table 5 shows that autoencoder leads to robustness improvements in most cases with considerable improvements in NYUv2 depth estimation. Overall, we note that autoencoder provides reasonable generalization and robustness improvements in most cases across tasks and datasets.

6 Limitations

The datasets used have limited training images likely leading to a small difference between Sep-De and One-De in our experimental setup. This small performance gap stands as a hindrance to sufficiently gauge the utility of the PDF algorithm. With more complex and larger datasets, this issue could be overcome. Additionally, the training time required to arrive at the final network is higher than normal as multiple retraining from initialization is required rendering the method infeasible in situations where only limited training resources is available. Another limitation is that we assume that the encoder is always shared among all tasks. We intend to explore a more training time efficient approach while also considering the encoder for fusion in our future work.

7 Conclusion

To curb task interference in the decoder while increasing positive transfer and improving inference efficiency, we proposed progressive decoder fusion starting from each task having its own decoder and grouping tasks in subsequent decoder stages in every progression step. At every step in this progression, tasks are grouped based on CKA similarity and the obtained network is retrained. We showed that the proposed PDF algorithm leads to a network that surpasses the offline grouped network in inference efficiency, generalization, and robustness. With this work, we wish to have reinforced interest in the research community to curb task interference and expose the full potential of MTNs.

References

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 2018.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2039–2050. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/16002f7a455a94aa4e91cc34ebdb9f2d-Paper.pdf`.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12379–12388, 2019.

Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=hqDb8d65Vfh`.

Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *ECCV*, 2018.

Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3854–3863. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/guo20e.html`.

Naresh Kumar Gurulingan, Elahe Arani, and Bahram Zonooz. Uninet: A unified scene understanding network and exploring multi-task relationships through the lens of adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 2239–2248, October 2021.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool. Reparameterizing convolutions for incremental multi-task learning without task interference. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, pp. 689–707, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58565-5.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/kornblith19a.html`.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=HJGU3Rodl`.

Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7337–7345, 2019. doi: 10.1109/CVPR.2019.00752.

An-An Liu, Yu-Ting Su, Wei-Zhi Nie, and Mohan Kankanhalli. Hierarchical clustering multi-task learning for joint human action grouping and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1): 102–114, 2017. doi: 10.1109/TPAMI.2016.2537337.

Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1871–1880, 2019.

Yao Lu, Sören Pirk, Jan Dlabal, Anthony Brohan, Ankita Pasad, Zhao Chen, Vincent Casser, Anelia Angelova, and A. Gordon. Taskology: Utilizing task relations at scale. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8696–8705, 2021.

Chengzhi Mao, Amogh Gupta, Vikram Nitin, Baishakhi Ray, Shuran Song, Junfeng Yang, and Carl Vondrick. Multitask learning strengthens adversarial robustness. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, volume 12347 of *Lecture Notes in Computer Science*, pp. 158–174. Springer, 2020. doi: 10.1007/978-3-030-58536-5_10. URL https://doi.org/10.1007/978-3-030-58536-5_10.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KJNcAkY8tY4.

Xiaojuan Qi, Renjie Liao, Zhengzhe Liu, Raquel Urtasun, and Jiaya Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 283–291, 2018.

N. Silberman, Derek Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

Trevor Scott Standley, Amir Roshan Zamir, Dawn Chen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *ICML*, 2020.

Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Many task learning with task routing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33, 2020.

S. Vandenhende, S. Georgoulis, B. De Brabandere, and L. Van Gool. Branched Multi-Task Networks: Deciding What Layers To Share. In *BMVC*, 2020a.

Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *ECCV*, 2020b.

Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 675–684, 2018. doi: 10.1109/CVPR.2018.00077.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf.

Amir R. Zamir, Alexander Sax, William Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Xiangyu Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. In *ECCV*, 2018.

A Appendix

A.1 Additional related works

In addition to the related works presented in the main paper, we differentiate our work with other research directions for completeness. An alternate line of work concerns fusing the learned task specific features using additional parameters to enhance task interaction (Xu et al., 2018; Vandenhende et al., 2020b). Fusion has also been done at the input, where data from multiple modalities are fused together (Liang et al., 2019). Unlike these works, we fuse the parameters of the networks themselves. Liu et al. (2017) consider a training scheme in which they alternate between multi-task learning and finding task-relatedness. In a training iteration, only the parameters of related tasks are trained thereby reducing interference from the learning objectives of other tasks. Similarly, notions of task relatedness have been proposed in the context of multitask learning (Standley et al., 2020; Lu et al., 2021) and transfer learning (Zamir et al., 2018; Dwivedi & Roig, 2019) to help improve generalization. Unlike these works, we group tasks and directly fuse task-specific parameters.

A.2 Centered Kernel Alingment (CKA)

CKA provides a way to compare different representations learned by a neural network. We use CKA to compare the output features from different tasks at a decoder stage. CKA is computed between every pair of task representations X and Y obtained using N images. We first take the mean across spatial locations[4]. We use the unbiased CKA estimation method provided by Nguyen et al. (2021) to first calculate $HSIC_1$ using Equation 2 where K and L represent Gram matrices XX^T and YY^T, respectively. $HSIC_1$ is used to calculate CKA with the aid of Equation 3 where $\tilde{K}$ and $\tilde{L}$ are modified versions of K and L with the diagonal entries made zero.

$$HSIC_1(K,L) = \frac{1}{n(n-3)}\left(tr(\tilde{K}\tilde{L}) + \frac{1^T\tilde{K}11^T\tilde{L}1}{(n-1)(n-2)} - \frac{2}{n-2}1^T\tilde{K}\tilde{L}1\right) \tag{2}$$

$$CKA = \frac{HSIC_1(K,L)}{\sqrt{HSIC_1(K,L)}\sqrt{HSIC_1(K,L)}} \tag{3}$$

Vandenhende et al. (2020a) use Representation Similarity Analysis (RSA). Similar to CKA, the mean across either the channel dimensions or the spatial dimensions is taken to obtain a vector for each input image. Next, a Representation Dissimilarity Matrix (RDM) of dimension N×N for each representation X and Y is calculated as 1 - Person correlation between all pairs of N feature vectors. From both the RDM matrices, a flattened vector from the upper triangle matrix is obtained. The Spearman's correlation between the two flattened vector represents the RSA[5] between representations X and Y.

A.3 Additional training details

We use UniNet architecture for the experiments and we refer to D6-D2 as stages 1-5. We use ResNet50 as backbone architecture to provide the encoder features E2 to E5 while E6 and E7 are obtained using the first two ResNet18 encoder stages. The encoder extracts the features from the input image and provides a feature representation with low spatial resolution. The decoder maps the extracted encoder features to a higher spatial resolution which is in turn mapped to the corresponding output space by each task head. To train the edge detection task in both datasets, we transform the semantic segmentation ground truth to edge detection ground truth. For the surface normals in Cityscapes, we use the depth ground truth transformed to surface normals ground truth for training. In NYUv2, we use the precomputed surface normals ground truth available in the dataset. Class balanced cross entropy loss, RMSE loss, class balanced binary cross entropy loss, L1 loss and MSE loss is used to train semantic segmentation, depth estimation, edge detection, surface normals and autoencoder, respectively.

A.4 Obtained task groupings

The obtained task groupings for both datasets at different stages using the PDF algorithm and Offline grouping are listed in Table 6. Additionally, the effect on the resultant task groupings while changing the grouping threshold is presented in Table 7.

A.5 NYUv2 CKA similarity matrices

The CKA similarity matrix obtained at different stages is provided in Figure 5. We only visualize the similarities between $\mathcal{S}$ and $\mathcal{E}$ through different stages as all other tasks split into their own branch right from stage 1. Similar to

[4]Note that alternatively, mean across channels can be taken followed by flattening along the spatial dimensions.

[5]Note there could be other versions of RSA with different formulations.

Table 6: Task groupings for both datasets at all candidate stages obtained with PDF algorithm and offline grouping. Blue highlighted cells show the stages that offline grouping deviates from progressive grouping.

Stage	Cityscapes		NYUv2	
	PDF	Offline	PDF	Offline
1,2,3	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
4	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
5	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$

Table 7: Task groupings for both datasets obtained with PDF algorithm and offline grouping using different thresholds for grouping (denoted as τ). Blue highlighted cells show the stages where the offline grouping deviates from progressive grouping.

Stage	τ	Cityscapes		NYUv2	
		PDF	Offline	PDF	Offline
1,2,3,4,5	0.0	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$
1,2,3,4,5	0.1	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$
1,2,3,4,5	0.2	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$
1,2,3,4	0.3	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$
5		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
1,2,3,4	0.4	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
5		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
1,2,3	0.5	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
4		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
5		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
1,2	0.6	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
3,4		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
5		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
1	0.7	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
2,3,4		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
5		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
1	0.8	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}, \mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
2		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
3,4,5		$[\mathcal{S}, \mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
1,2,3,4,5	0.9	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$
1,2,3,4,5	1.0	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$	$[\mathcal{S}], [\mathcal{E}], [\mathcal{D}], [\mathcal{N}], [\mathcal{A}]$

Cityscapes similarities in Figure 3, as we move towards the right, the decoder representations become increasingly similar with progressive grouping (green box) but increasingly dissimilar with offline grouping (blue box).

A.6 Detailed results of all networks

We report the mean and standard deviation of generalization results in Table 10 and Table 12, respectively. Likewise the mean and standard deviation of robustness results are reported in Table 11 and Table 13, respectively. PDF-1 to PDF-5 refer to the various networks obtained at different stages of the decoder progression. Table 8 and Table 9 summarize the generalization and robustness results of PDF and Offline algorithms over 3 runs to ease comparison. In these tables, we highlight a result if the difference in performance is greater than the standard deviation of both PDF and Offline network results. Although PDF achieves better results than Offline, average over 3 runs brings the results of the two algorithms close to each other in most cases. However, in Table 8 we see that PDF performs better than Offline in 8 cases. In robustness results presented in Table 9 PDF performs on par with Offline in most cases and better in 1 case. Overall, PDF can still be considered to provide a network with better decoder sharing scheme than Offline.

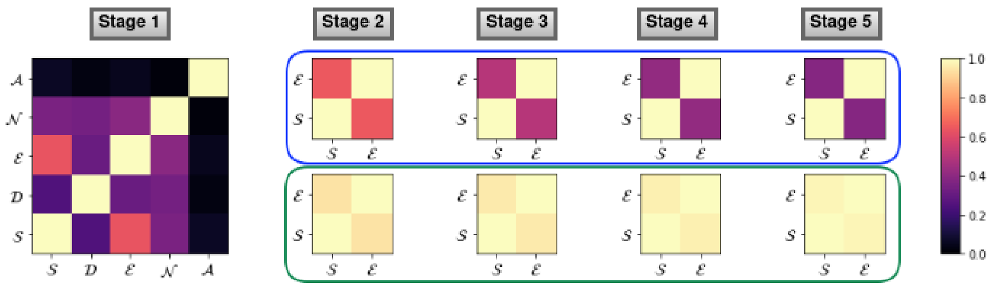

Figure 5: CKA similarity matrix at different decoder stages obtained with offline grouping (blue box) and with progressive (green box) in the NYUv2 dataset. As we move towards the right, decoder representations become increasingly similar with progressive grouping but increasingly dissimilar with offline grouping.

Table 8: Mean and standard deviation of generalization results of PDF and Offline algorithms over 3 seeds.

Network		$\mathcal{S}$ (mIoU %) ↑					$\mathcal{D}$ (RMSE) ↓				
		IID	noise	blur	weather	digital	IID	noise	blur	weather	digital
CS	Offline	73.01$\pm$0.34	18.02$\pm$0.21	48.91$\pm$0.33	35.85$\pm$0.11	61.13$\pm$0.48	5.35$\pm$0.02	14.85$\pm$0.67	13.25$\pm$0.11	11.88$\pm$0.44	7.40$\pm$0.15
	PDF	72.92$\pm$0.25	**19.72**$\pm$1.39	49.19$\pm$0.22	**36.68**$\pm$0.29	60.98$\pm$0.24	5.31$\pm$0.01	15.09$\pm$0.52	13.04$\pm$0.38	12.09$\pm$0.03	7.52$\pm$0.33
NYU	Offline	42.46$\pm$0.28	4.51$\pm$0.32	23.87$\pm$0.32	18.93$\pm$0.13	26.54$\pm$0.42	47.98$\pm$0.24	106.92$\pm$3.88	**81.66**$\pm$0.82	80.88$\pm$1.18	66.40$\pm$0.53
	PDF	42.28$\pm$0.19	**5.04**$\pm$0.33	**24.27**$\pm$0.22	18.83$\pm$0.14	**27.14**$\pm$0.08	**47.22**$\pm$0.49	113.53$\pm$10.19	83.89$\pm$0.70	**78.14**$\pm$1.39	**65.23**$\pm$0.85

Table 9: Mean and standard deviation of robustness of PDF and Offline algorithms results over 3 seeds.

Network	CS				NYU			
	$\mathcal{S}$ (mIoU %) ↑		$\mathcal{D}$ (RMSE) ↓		$\mathcal{S}$ (mIoU %) ↑		$\mathcal{D}$ (RMSE) ↓	
	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ
Offline	46.02$\pm$0.27	8.12$\pm$0.33	15.27$\pm$0.13	44.90$\pm$0.46	20.46$\pm$0.40	3.55$\pm$0.29	110.72$\pm$1.19	308.61$\pm$4.96
PDF	45.75$\pm$0.34	8.07$\pm$0.47	15.19$\pm$0.13	44.93$\pm$1.03	20.38$\pm$0.24	3.58$\pm$0.06	**107.71**$\pm$0.19	304.94$\pm$4.74

Table 10: Mean of generalization results of networks obtained at various stages of the PDF algorithm over 3 seeds.

Network		$\mathcal{S}$ (mIoU %) ↓					$\mathcal{D}$ (RMSE) ↑				
		IID	noise	blur	weather	digital	IID	noise	blur	weather	digital
CS	PDF-5	72.92	19.72	49.19	36.68	60.98	5.31	15.09	13.04	12.09	7.52
	PDF-4	72.41	18.67	48.77	35.54	60.93	5.33	15.70	13.15	11.89	7.41
	PDF-3	72.63	19.25	48.88	35.86	60.69	5.33	14.25	13.11	11.98	7.57
	PDF-2	72.99	16.39	49.08	36.37	61.13	5.32	15.78	12.94	12.21	7.36
	PDF-1	72.17	19.27	48.34	35.72	60.76	5.34	15.11	13.20	12.29	7.34
NYU	PDF-5	42.28	5.04	24.27	18.83	27.14	47.22	113.53	83.89	78.14	65.23
	PDF-4	42.44	4.64	24.10	19.04	26.79	47.84	115.47	83.13	80.34	65.34
	PDF-3	42.12	4.84	23.88	18.73	26.28	47.84	99.48	83.44	82.21	64.92
	PDF-2	41.90	5.09	23.95	18.95	26.48	47.70	120.09	83.62	81.66	65.34
	PDF-1	42.33	4.30	24.00	18.80	26.76	48.00	120.28	83.79	80.40	65.53

Table 11: Mean of robustness results of networks obtained at various stages of the PDF algorithm over 3 seeds.

Network	CS				NYU			
	$\mathcal{S}$ (mIoU %) ↑		$\mathcal{D}$ (RMSE) ↓		$\mathcal{S}$ (mIoU %) ↑		$\mathcal{D}$ (RMSE) ↓	
	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ
PDF-5	45.75	8.07	15.19	44.93	20.38	3.58	107.71	304.94
PDF-4	46.21	8.08	15.22	44.85	20.29	3.48	110.98	309.45
PDF-3	46.13	8.14	15.28	44.72	20.00	3.34	110.38	313.14
PDF-2	46.44	8.64	15.16	43.57	20.33	3.39	110.74	311.96
PDF-1	45.46	8.40	15.26	45.60	20.22	3.33	109.66	300.99

Table 12: Standard deviation of generalization results of different networks over 3 seeds.

Network		$\mathcal{S}$ (mIoU %)					$\mathcal{D}$ (RMSE)				
		IID	noise	blur	weather	digital	IID	noise	blur	weather	digital
CS	One-De	0.25	0.54	0.09	1.31	0.24	0.01	0.10	0.19	0.50	0.28
	Sep-De	0.09	1.76	0.22	0.92	0.06	0.01	1.10	0.15	0.23	0.05
	Offline	0.34	0.21	0.33	0.11	0.48	0.02	0.67	0.11	0.44	0.15
	PDF-5	0.25	1.39	0.22	0.29	0.24	0.01	0.52	0.38	0.03	0.33
	PDF-4	0.57	0.38	0.14	0.44	0.10	0.01	0.42	0.20	0.40	0.16
	PDF-3	0.45	0.74	0.09	0.07	0.17	0.02	1.06	0.11	0.03	0.33
	PDF-2	0.39	0.76	0.35	0.56	0.76	0.02	0.78	0.17	0.38	0.15
	PDF-1	0.63	1.01	0.09	0.77	0.40	0.03	2.14	0.17	0.21	0.12
	PDF-5-No-$\mathcal{A}$	0.38	0.88	0.20	0.93	0.59	0.01	1.15	0.16	0.32	0.04
NYU	One-De	0.42	0.67	0.48	0.32	0.20	0.10	11.35	0.49	1.17	0.37
	Sep-De	0.40	0.10	0.15	0.33	0.25	0.16	5.43	0.77	0.77	0.96
	Offline	0.28	0.32	0.32	0.13	0.42	0.24	3.88	0.82	1.18	0.53
	PDF-5	0.19	0.33	0.22	0.14	0.08	0.49	10.19	0.70	1.30	0.85
	PDF-4	0.41	0.48	0.45	0.11	0.29	0.27	8.15	1.01	1.14	1.74
	PDF-3	0.23	0.79	0.33	0.19	0.41	0.14	7.70	1.33	2.54	0.64
	PDF-2	0.68	0.26	0.55	0.27	0.41	0.16	9.95	1.01	2.45	1.10
	PDF-1	0.35	0.52	0.51	0.23	0.50	0.12	4.24	0.93	2.42	0.71
	PDF-5-No-$\mathcal{A}$	0.10	1.06	0.29	0.09	0.22	0.24	4.07	0.77	0.40	0.71

Table 13: Standard deviation of robustness results of different networks over 3 seeds.

Network	CS				NYU			
	$\mathcal{S}$ (mIoU %)		$\mathcal{D}$ (RMSE)		$\mathcal{S}$ (mIoU %)		$\mathcal{D}$ (RMSE)	
	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ	low ϵ	high ϵ
One-De	0.64	0.60	0.11	0.33	0.28	0.06	1.09	2.12
Sep-De	0.73	0.61	0.06	1.03	0.14	0.16	0.19	3.55
Offline	0.27	0.33	0.13	0.46	0.40	0.29	1.14	4.96
PDF-5	0.34	0.47	0.13	1.03	0.24	0.06	0.19	4.74
PDF-4	0.78	0.29	0.09	0.74	0.18	0.19	0.50	8.77
PDF-3	0.27	0.08	0.10	1.08	0.35	0.05	0.55	2.96
PDF-2	0.35	0.50	0.09	0.54	0.25	0.10	0.79	4.42
PDF-1	0.84	0.38	0.06	1.00	0.38	0.14	0.20	3.05
PDF-5-No-$\mathcal{A}$	0.05	0.03	0.20	0.36	0.22	0.17	15.53	25.76