Model Weights Reflect a Continuous Space of Input Image Domains

Supplementary Material

6. Synthetic Dataset

The synthetic dataset is based on Cityscapes [9]. The different domains are simply colour alterations according to certain channels and strengths, which is achieved by adding a constant value to the entire image. More concretely, a particular fraction $f \in [0, 1]$ of the maximal pixel value (255) is added to each pixel along the involved colour channels. Algorithm 1 gives an overview of the procedure.

Algorithm 1 An algorithm with caption

```
procedure ColourAdjust($img, channels, f$)
    for $c \in channels$ do
        $img[c] \leftarrow img[c] + f \times 255$
        $img[c] \leftarrow \text{clip}(img[c], 0, 255)$
    end for
end procedure
```

There are six colours: red (red channel adjustment), yellow (red and green channels), green (green channel), cyan (green and blue channels), blue (blue channel), and purple (red and blue channels). Additionally, the alteration is done in different strengths where $f \in \{0.2, 0.4, 0.6, 0.8\}$. This produces a total of 24 domains, in addition to the unaltered one. Figure 13 shows an example image of each domain (the first column is considered as one, since it is simply the set of unaltered images).

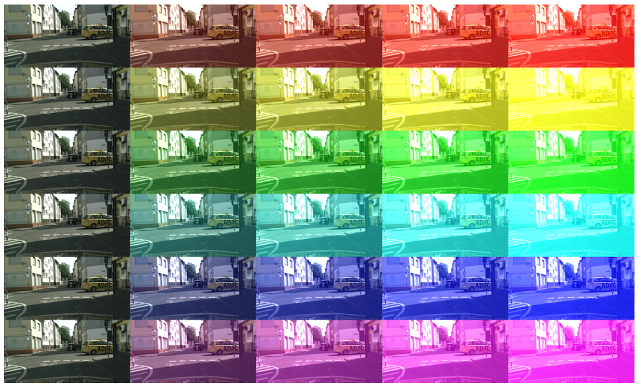

Figure 13. Example of all the synthetic colour adjustments for each combination of colour channels (red, red and green, green, green and blue, blue, and red and blue) and each of the strength factors $(0.2, 0.4, 0.6, 0.8)$.

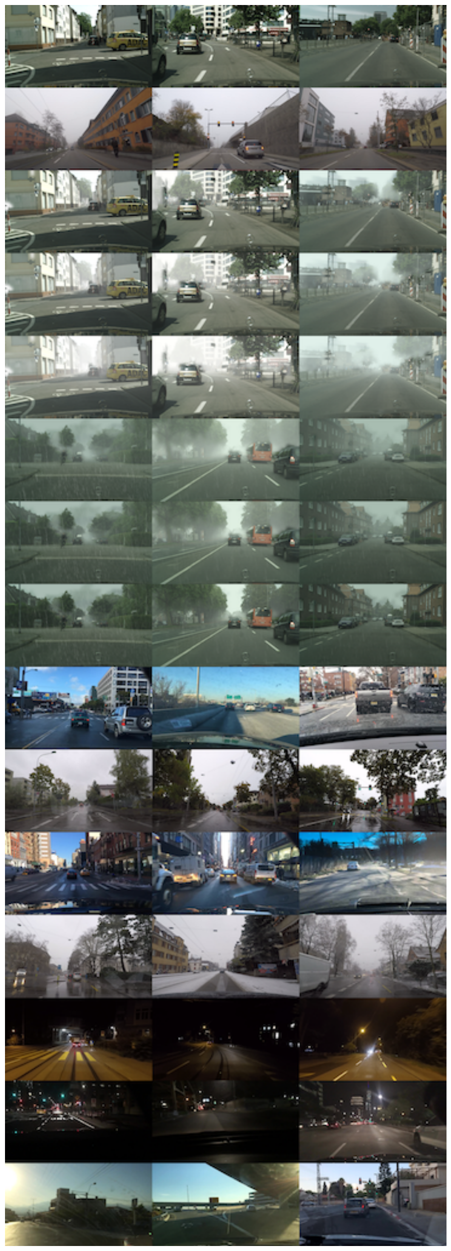

Figure 14. Example of all the real weather domains. Top to bottom: Day (Cityscapes), Fog (ACDC), FOG (Foggy Cityscapes - Low), FOG (Foggy Cityscapes - Medium), FOG (High), Rain (Rainy Cityscapes - High), Rain (Rainy Cityscapes - Medium), Rain (Rainy Cityscapes - Low), Rain (BDD), Rain (ACDC), Snow (BDD), Snow (ACDC), Night (ACDC), Night (BDD), and Dawn (BDD).

	R.2	R.4	R.6	R.8	RG.2	RG.4	RG.6	RG.8	G.2	G.4	G.6	G.8	*.0
All	48.9	49.6	49.2	46.8	49.9	49.2	48.0	44.2	50.2	50.0	49.7	48.4	48.0
Experts	**51.2**	**51.1**	**51.0**	**50.7**	**51.3**	**50.6**	**50.2**	**49.0**	**51.2**	**51.2**	**50.7**	**50.3**	**53.9**

	GB.2	GB.4	GB.6	GB.8	B.2	B.4	B.6	B.8	RB.2	RB.4	RB.6	RB.8	Avg
All	48.9	49.6	48.7	45.7	48.7	49.0	48.6	48.1	50.2	49.8	49.2	46.6	48.6
Experts	**51.1**	**50.7**	**50.1**	**48.1**	**51.7**	**50.8**	**50.5**	**49.7**	**51.2**	**51.6**	**50.3**	**49.1**	**50.7**

Table 5. Results on the synthetic colour adaptations. Altered channels are indicated by the letter (RGB mode) and the decimal indicates the strength of the alteration. "Experts" indicates the models trained on the respective alteration only, "All" indicates the model train on all channel variations. Best performance per column is in bold. *.0 indicates the original unaltered dataset.

7. Real Dataset

The real data mix is made up of different datasets that provide either explicit weather splits or additional metadata to create them. Figure 14 shows a few example images of each domain.

Clear Day. This domain is represented by Cityscapes [9]. It contains road scenes from German and Swiss cities labelled for semantic segmentation. Bounding boxes are obtained by extracting the extreme points of the instance masks.

Fog. There are four different foggy image splits. Three of them come from Foggy Cityscapes [20], a dataset containing the same images as Cityscapes but with a synthetic fog augmentation. The fog is controlled by the parameter β, and the dataset includes images with three different strengths $\beta \in \{0.005, 0.01, 0.02\}$ (corresponding to a low, medium, and high level of fog, respectively), which this work uses as three different domains. While it is handy to have such a gradation in fog, it is still a synthetic addition, which is often quite visible. The fourth fog split comes from ACDC [21], a dataset of real images taken in Swiss cities. While these images are real, the fog in them is very faint. Which is why they closely resemble overcast daytime images.

Rain. There are five rainy image splits. Three of them come from Rainy Cityscapes [13], a synthetic rain adaptation of Cityscapes very similar to Foggy Cityscapes, now controlled by the parameters $(\alpha, \beta, dropsize) \in \{(.01, .005, .01), (.02, .01, .005), (.03, .015, .002)\}$ (corresponding to a low, medium, and high intensity of rain, respectively). Again, this work keeps the three levels as separate splits within the rain domain. The augmentation also uses twelve different rain patterns for each image at each rain intensity, which have all been used in this work (*i.e.* no distinction has been made on the basis of rain pattern). The fourth split is composed of real rain data from ACDC, and the fifth split comes from BDD100k [26] using the metadata on weather and time of day, in this case rainy and daytime, respectively.

Snow. There are two snow splits. One from ACDC. One from BDD100k, with metadata corresponding to snowy weather and daytime.

Night. There are two nighttime splits. One from ACDC and one from BDD100k. The latter uses the data annotated as night and clear, partly cloudy, or overcast weather.

Dawn. Data for the domain of dawn comes from BDD100k, with metadata annotations for weather of clear, partly cloudy, or overcast and time of day indicating dawn/dusk.

8. Fine-tuning experiments

Model. The model used for the experiments is an FCOS [24] (where the centredness is based on the class branch), with a VGG-16 backbone [22] and an FPN. The loss function is based on the Harmonious Loss from [10], where only the supervised component is used. Tables 6 and 7 show additional configuration parameter values for the model and loss, respectively.

Parameter	Value
Trainable layers	5 (all)
Returned layers	3, 4, 5
Extra blocks	P6 and P7
NMS threshold	0.6
Score threshold	0.05
Center sampling radius	0

Table 6. Parameter values used to configure the FCOS model.

Parameter	Value
α	0.75
γ	2
Class weight	1
Loss type	IoU

Table 7. Parameter values used to configure the loss function.

Pre-training. The model is pre-trained on Cityscapes [9] starting from the Pytorch (v2.5) Imagnet backbone weights. The best checkpoint is used to initialize the model for fine-tuning. Performance is measured according to mean Average Precision (mAP) with a threshold of 0.5 (*i.e.* mAP$_{50}$) according to the method used for COCO. Table 8 shows the relevant training parameters that were used. As augmentations the training uses: horizontal flipping (probability 0.5), vertical flipping (probability 0.2), colour jitter (probability 0.5, brightness 0.4, contrast 0.4, hue 0.1, and saturation 0.4), grayscale (probablility 0.2), Gaussian blur (probability 0.5, $\sigma = (0.1, 2)$), scaling (probability 0.5, range factor [0.8, 1.5]), and random erasing (probability 0.7). Finally, all inputs are scaled to the shortest edge length of 800px or maximum size of 1333px, and the image values are normalised to the range [0, 1] by dividing by 255.

	Parameter	Value
	Iterations	60k
	Learning rate	0.008
	Batch size	8
Optimiser	Method	SGD
	Momentum	0.9
	Weight decay	$5 * 10^{-4}$
Learning rate scheduler	Method	Warmup Step
	γ	0.1
	Milestones	20k, 40k
	Warm-up factor	0.001
	Warm-up iterations	1k
	Warm-up method	Linear

Table 8. Parameter values used to configure the training.

Synthetic fine-tuning. Fine-tuning refers to the creation of expert models on each individual domain. This is done by initialising the model with the best weights from the pre-training and starting a new training with only data from a single colour and strength (*e.g.* red with adjustment factor 0.8, as explained in Sec. 6). Importantly, during this stage only the backbone (including FPN) is trained, the heads are frozen. Also note that none of the models see any unaltered images during this training. The model that is trained on all input variations simply samples data from the joint dataset containing all images from all domains (but not the unaltered ones). The training uses the same augmentations as the pre-training minus the colour jitter and grayscale, to prevent any alterations to the domain information or provide information from other domains. Table 9 shows the training parameters used for this stage.

	Parameter	Value
	Iterations	10k
	Learning rate	0.001
	Batch size	8
Optimiser	Method	SGD
	Momentum	0.9
	Weight decay	$5 * 10^{-4}$
Learning rate scheduler	Method	Warmup Step
	γ	0.1
	Milestones	4k, 8k
	Warm-up factor	0.001
	Warm-up iterations	1k
	Warm-up method	Linear

Table 9. Parameter values used to configure the synthetic data fine-tuning training.

	Parameter	Value
	Iterations	15k
	Learning rate	0.0008
	Batch size	8
Optimiser	Method	SGD
	Momentum	0.9
	Weight decay	10^{-4}
Learning rate scheduler	Method	Warmup Step
	γ	0.1
	Milestones	3k, 10k
	Warm-up factor	0.001
	Warm-up iterations	1k
	Warm-up method	Linear

Table 10. Parameter values used to configure the real data fine-tuning training.

Real data fine-tuning. The fine-tuning of models on real data is fully analogous to the fine-tuning on synthetic data, with some small changes to training parameters as shown in Tab. 10. The augmentations are the same as in the pre-training stage. To strengthen the idea that the weight space is distributed according to input data, three separate models are trained per domain. As shown, they converge to similar points in weight space with very similar performance.

9. Domain Arithmetic Analysis

This section contains the full tables of results on the synthetic dataset. Tab. 11 contains the results for the linear interpolation of secondary colours (yellow, cyan, and magenta) using two primary colour experts (red, green, and blue). Table 12 shows the performance comparison between the experts, the jointly trained model, and the adaptive model on all synthetic data domains. Table 13 shows the results of strength interpolation results of the adaptive method compared to the model trained on all domains. The domain encoder used to learn the PC projections has the following structure (where ChannelMean() simply takes the mean per channel):

```
Sequential(
    (0): Conv2d(3, 20, kernel_size=(3, 3), stride=(3, 3))
    (1): ReLU()
    (2): Conv2d(20, 20, kernel_size=(5, 5), stride=(5, 5))
    (3): ReLU()
    (4): AvgPool2d(kernel_size=4, stride=4, padding=0)
    (5): Conv2d(20, 20, kernel_size=(5, 5), stride=(5, 5))
    (6): ReLU()
    (7): ChannelMean()
    (8): Linear(in_features=20, out_features=20, bias=True)
    (9): ReLU()
    (10): Linear(in_features=20, out_features=20, bias=True)
    (11): ReLU()
    (12): Linear(in_features=20, out_features=10, bias=True)
)
```

	RG.8	RG.8 (Lin.)	RB.8	RB.8 (Lin.)	GB.8	GB.8 (Lin.)
Source	8.8	8.8	5.6	5.6	19.5	19.5
Expert X.8	11.4	15.9	15.0	12.6	23.3	30.5
Expert Y.8	9.5	25.9	8.6	18.8	17.7	12.7
Interpolation	8.1	22.8	29.1	20.6	25.4	31.4
Expert XY.8	49.8	37.3	49.9	36.4	48.8	40.0

Table 11. Linear interpolation results of single colour channel models tested on two channel adjustments. "Expert X" and "Y" for the combination "XY" mean the first and second single channel experts (*i.e.* for RG, X is R and Y is G). "Lin." indicates linearised training following Ortiz-Jimenez *et al.* [18], opposed to standard training without any constraints. "Source" indicates the pre-trained model that has not seen any altered data.

	R.2	R.4	R.6	R.8	RG.2	RG.4	RG.6	RG.8	G.2	G.4	G.6	G.8	*.0
All	48.9	$\underline{49.6}$	$\underline{49.2}$	46.8	49.9	$\underline{49.2}$	$\underline{48.0}$	44.2	50.2	$\underline{50.0}$	$\underline{49.7}$	48.4	48.0
Adaptive	$\underline{50.0}$	**50.7**	**50.6**	$\underline{50.1}$	**51.1**	**50.9**	$\underline{47.6}$	$\underline{46.0}$	$\underline{50.7}$	$\underline{50.1}$	$\underline{50.0}$	$\underline{49.3}$	$\underline{50.6}$
Experts	**51.2**	**51.1**	**51.0**	**50.7**	**51.3**	**50.6**	**50.2**	**49.0**	**51.2**	**51.2**	**50.7**	**50.3**	**53.9**
	GB.2	**GB.4**	**GB.6**	**GB.8**	**B.2**	**B.4**	**B.6**	**B.8**	**RB.2**	**RB.4**	**RB.6**	**RB.8**	**Avg**
All	$\underline{48.9}$	$\underline{49.6}$	48.7	45.7	48.7	$\underline{49.0}$	48.6	48.1	$\underline{50.2}$	49.8	$\underline{49.2}$	$\underline{46.6}$	48.6
Adaptive	$\underline{48.9}$	**50.4**	$\underline{49.3}$	$\underline{46.9}$	$\underline{50.0}$	**50.8**	$\underline{49.2}$	$\underline{49.2}$	**51.3**	$\underline{51.1}$	$\underline{49.4}$	$\underline{46.9}$	$\underline{49.6}$
Experts	**51.1**	**50.7**	**50.1**	**48.1**	**51.7**	**50.8**	**50.5**	**49.7**	**51.2**	**51.6**	**50.3**	**49.1**	**50.7**

Table 12. Performance comparison (mAP) between the experts, the jointly trained model ("All"), and the adaptive method that learns to recombine PCs. Altered channels and strength are indicated by the letter (RGB mode) and the decimal, respectively. "*.0" indicates the original unaltered dataset. Best performance per column is bolded, second best is underlined (models are considered to be on par if the difference is less than 0.5).

	R.1	R.3	R.5	R.7	RG.1	RG.3	RG.5	RG.7	G.1	G.3	G.5	G.7
All	48.3	49.5	49.5	47.8	48.6	49.8	48.6	46.8	**50.0**	**50.1**	**49.9**	**49.3**
Adaptive	**49.0**	**50.3**	**50.5**	**50.2**	**49.9**	**51.2**	**50.0**	**48.0**	49.4	**50.5**	**50.0**	**49.7**
	GB.1	GB.3	GB.5	GB.7	B.1	B.3	B.5	B.7	RB.1	RB.3	RB.5	RB.7
All	**48.5**	**50.0**	**49.3**	47.4	48.2	49.2	**49.0**	48.2	48.2	50.1	49.8	**48.0**
Adaptive	**48.9**	**50.2**	**50.1**	**48.5**	**49.4**	**51.6**	**49.4**	**49.1**	**50.0**	**51.2**	**50.4**	**47.8**

Table 13. Performance comparison (mAP) between the experts, the jointly trained model ("All"), and the adaptive method that learns to recombine PCs. Altered channels are indicated by the letter (RGB mode) and the decimal indicates the strength of the alteration. "*.0" indicates the original unaltered dataset. Best performance per column is bolded, second best is underlined (models are considered to be on par if the difference is less than 0.5).

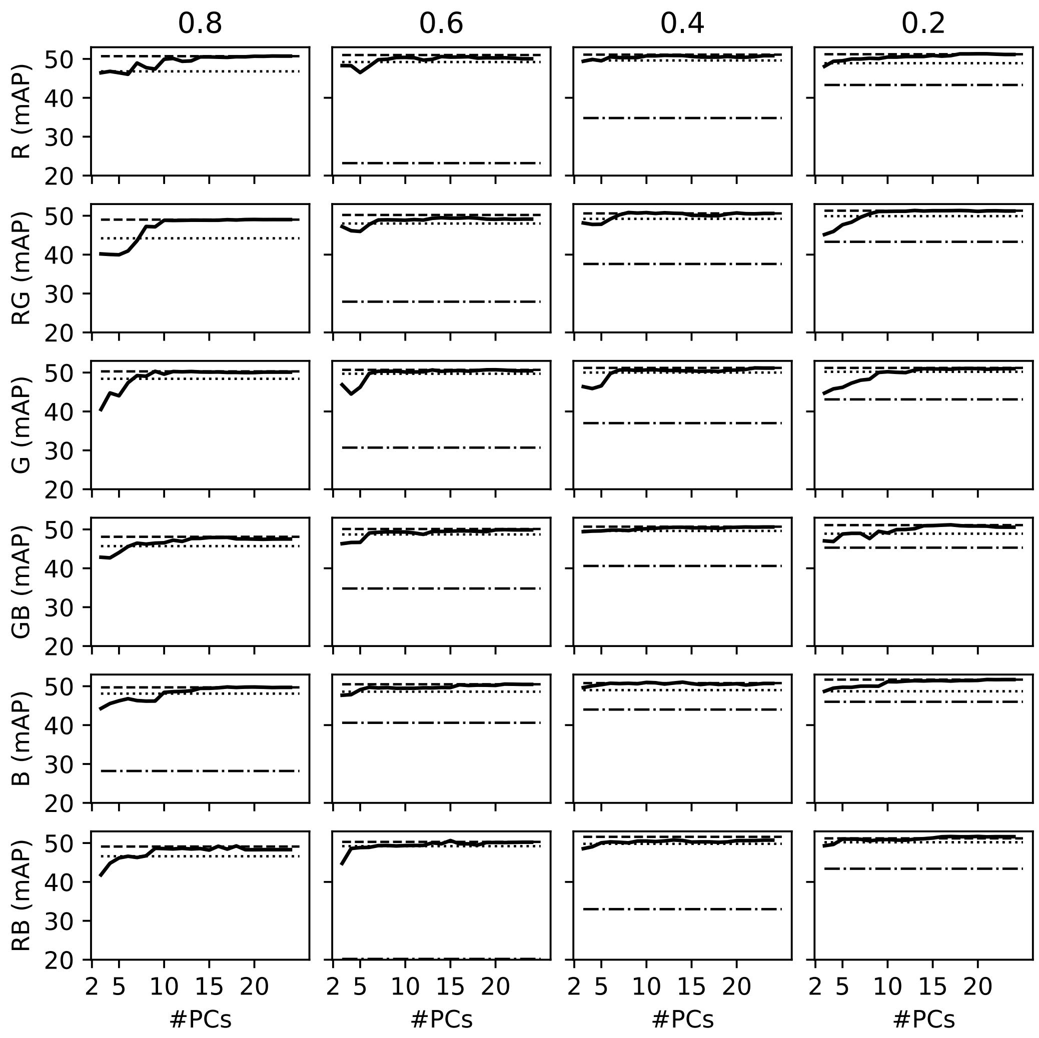

Figure 15. Model performance as a function of number of Principal Components (PCs) used for the reconstruction on the red domains (adjustment strength decreases left to right). The dashed line (- -) indicates the original performance, the dotted line (..) indicates the performance of the model trained with all data, and the dash-dot line (-.) indicates the performance of the model trained on unaltered images.