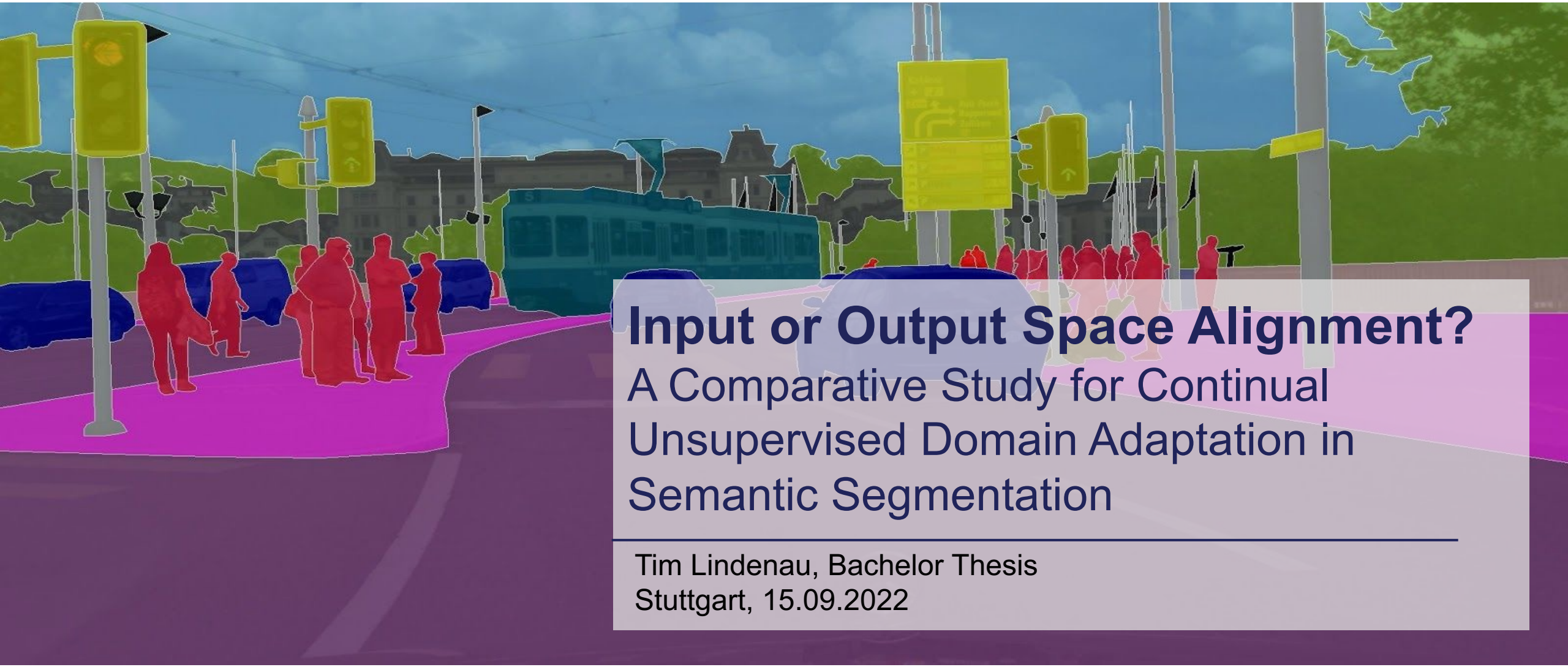


Input or Output Space Alignment?

A Comparative Study for Continual Unsupervised Domain Adaptation in Semantic Segmentation

Tim Lindenau, Bachelor Thesis
Stuttgart, 15.09.2022

Agenda

A Motivation for Continual UDA in Semantic Segmentation

B 5 Frameworks for Continual UDA

C Experiment 1: Comparison of the 3 Different Style Transfer Algorithms

D Experiment 2: Comparison of All 5 Frameworks for Continual UDA

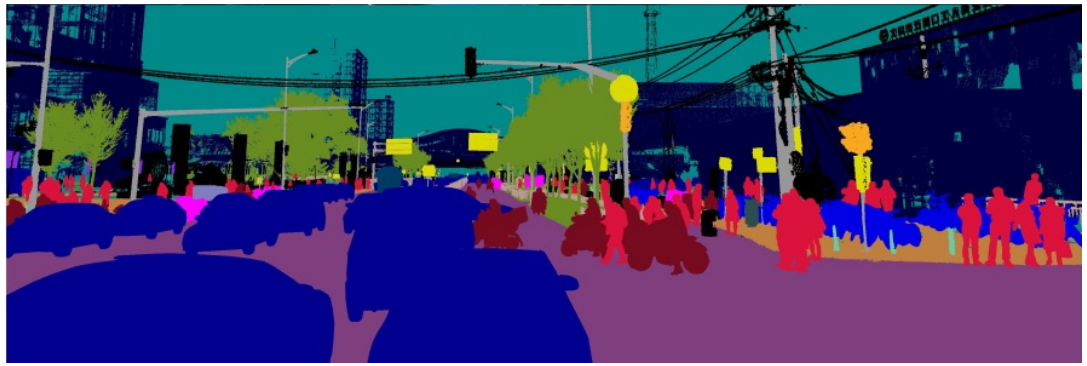
E Conclusion

Motivation for Continual UDA in Semantic Segmentation

Semantic Segmentation Provides Good Scene Understanding Required for Autonomous Driving – However: High Effort for Image Labeling

Original Image

Segmented Image: Every pixel assigned to one class



- **Semantic Segmentation:** Task of assigning every pixel in an image to one class, e.g. car, street, people
- Good results with methods based on **Convolutional Neural Networks**
- **High manual effort** for labeling training images
  - Ca. 90 minutes per image to identify & label all relevant objects
  - Large number of images required (‚ballpark figure‘: several thousand)

Continual UDA Helps to Achieve Good Segmentation Performance for Different Environments

The challenge: many different environments

Sunny **Fog** **Night** **Snow**

- Car experiences **many different environments** and segmentation performance must be good for all of them
- Segmentation performance decreases for environments unlike training environment. Reason: **Shift in the distribution** of input data

  → **Labeled training data required for all environments.**
  In practice impossible due to high manual labeling effort

The technical solution: Continual UDA

(1) **Unsupervised Domain Adaptation (UDA):**

- Labeled training data for **one source environment**/domain available
- Improve performance for **one target environment** without any labels

(2) **Continual learning:** Sequentially adapt model to **many target environments** without forgetting

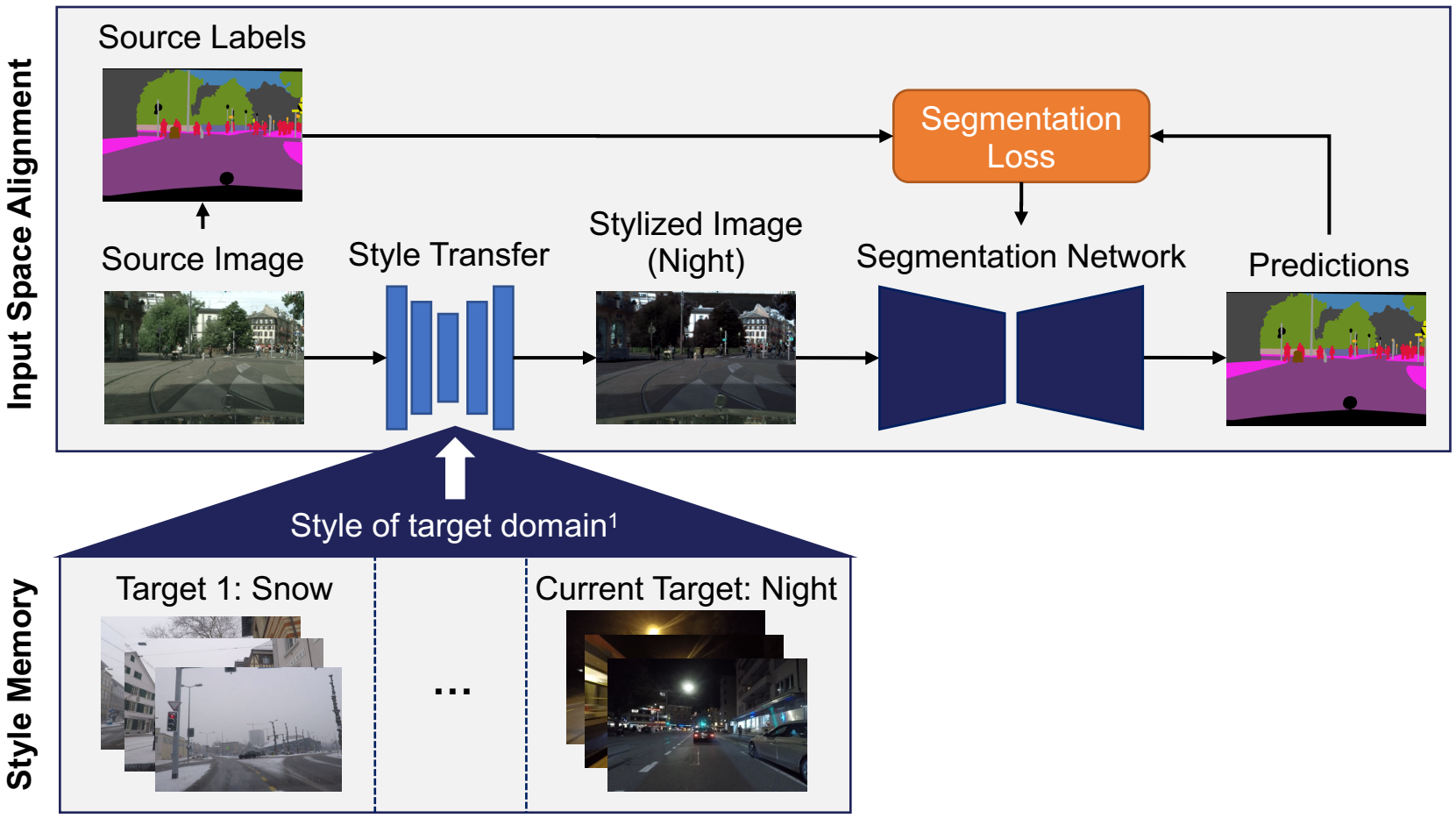
Images from Cordts et al., 2016 & Sakaridis et al., 2021

5 Frameworks for Continual UDA

I Will Discuss Five Frameworks for Continual UDA

	Framework		UDA	Continual Learning
a	**CACE**	**Established framework** for continual UDA	**Input space** alignment using **style transfer**	**Replay** to **prevent forgetting**
b	**C-WCT²**	New framework with **enhanced details**		
c	**C-CMD**	New framework with **more accurate stylization**		
d	**ETM**	New framework, **never compared against CACE**	**Output space** alignment using **adversarial learning**	**Network expansion**
e	**MuHDi**	New framework, **never compared against CACE**		**Multiple specialized classifiers**

a The CACE Framework for Continual UDA: Input Space Alignment to Adapt to New Domains and a Style Memory to Prevent Forgetting



[1]Unlike illustrated not the images themselves, but only their style is stored which is significantly more memory efficient

UDA: Input space aligned using style transfer

- Style transfer to stylize source images in look of target domain
- Source labels remain valid and can be used to train segmentation network

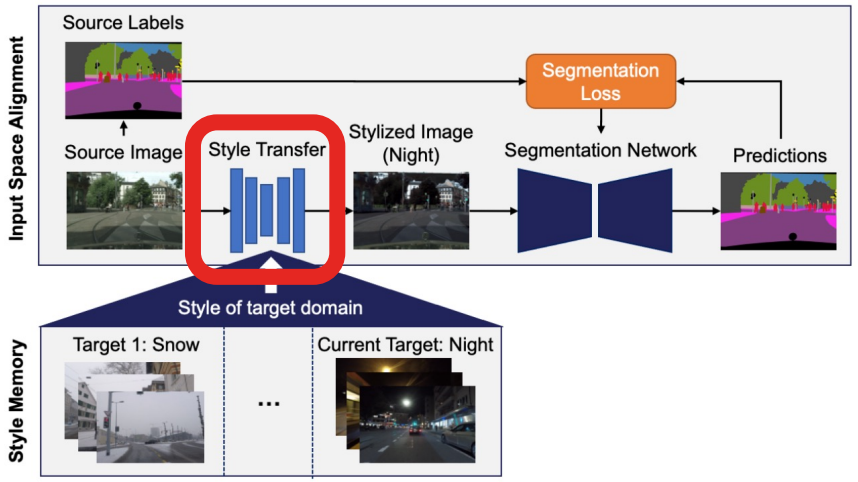
Continual learning: Style memory

- Style memory stores styles of all experienced target domains
- When training, alternate between style from current target domain (UDA) and random previous domain (prevent forgetting)

CACE framework presented by Marsden et al., 2022
Images from Cordts et al., 2016 & Sakaridis et al., 2021

C-WCT² & C-CMD: Two New Frameworks Based on CACE With Enhanced Style Transfer Algorithms

CACE-like frameworks can be created by exchanging the style transfer algorithm



- Style transfer from CACE has some known limitations
- Created two new frameworks just by exchanging the style transfer algorithm **(C-WCT², C-CMD)**

Advanced style transfer algorithms WCT² & CMD

CACE limitations	Potential solution
Loss of fine details Inevitable information loss in encoder/decoder structure (max pooling)	**WCT²-algorithm** Increased level of detail by overcoming information loss with new pooling operation (Wavelet pooling)
Theoretical limitations Matching mean and variance cannot provide for perfect distribution alignment	**CMD-algorithm** More exact stylizations using higher-order moments

2 New Frameworks, Better Results?

WCT² presented by Yoo et al., 2019
CMD presented by Kalischek et al., 2021

Output Space Alignment: Improving Target Domain Performance Using Structural Similarity in the Output Space

Input space:

Output space:

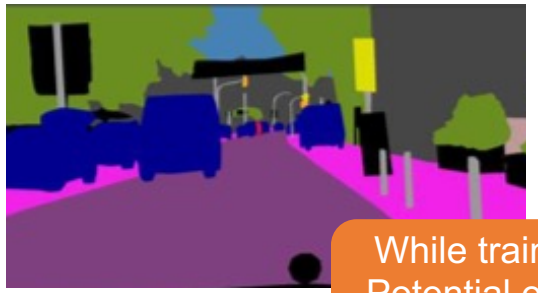
Predictions good due to available labels

Source

Large difference in appearance

Structurally similar (e.g. cars driving on street)

Target



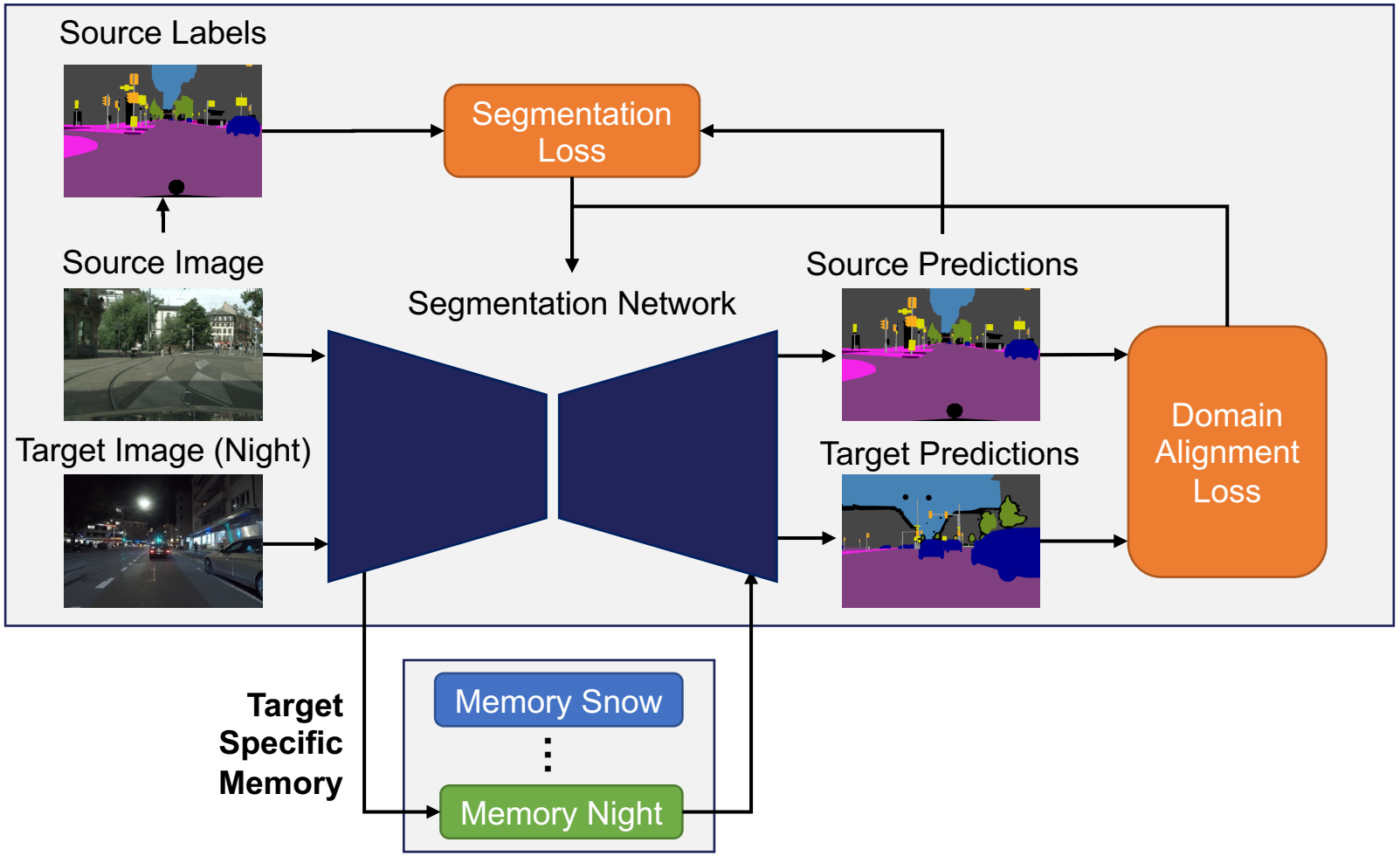
While training: Potential errors in predictions

Output space alignment to improve predictions for the target domain:

The neural network is trained to make the target domain predictions structurally similar to the predictions of the source domain

Idea and images from Tsai et al., 2020

d The ETM Framework for Continual UDA: Output Space Alignment to Adapt to New Domains and Target Specific Memory Modules to Prevent Forgetting



UDA: Output space alignment

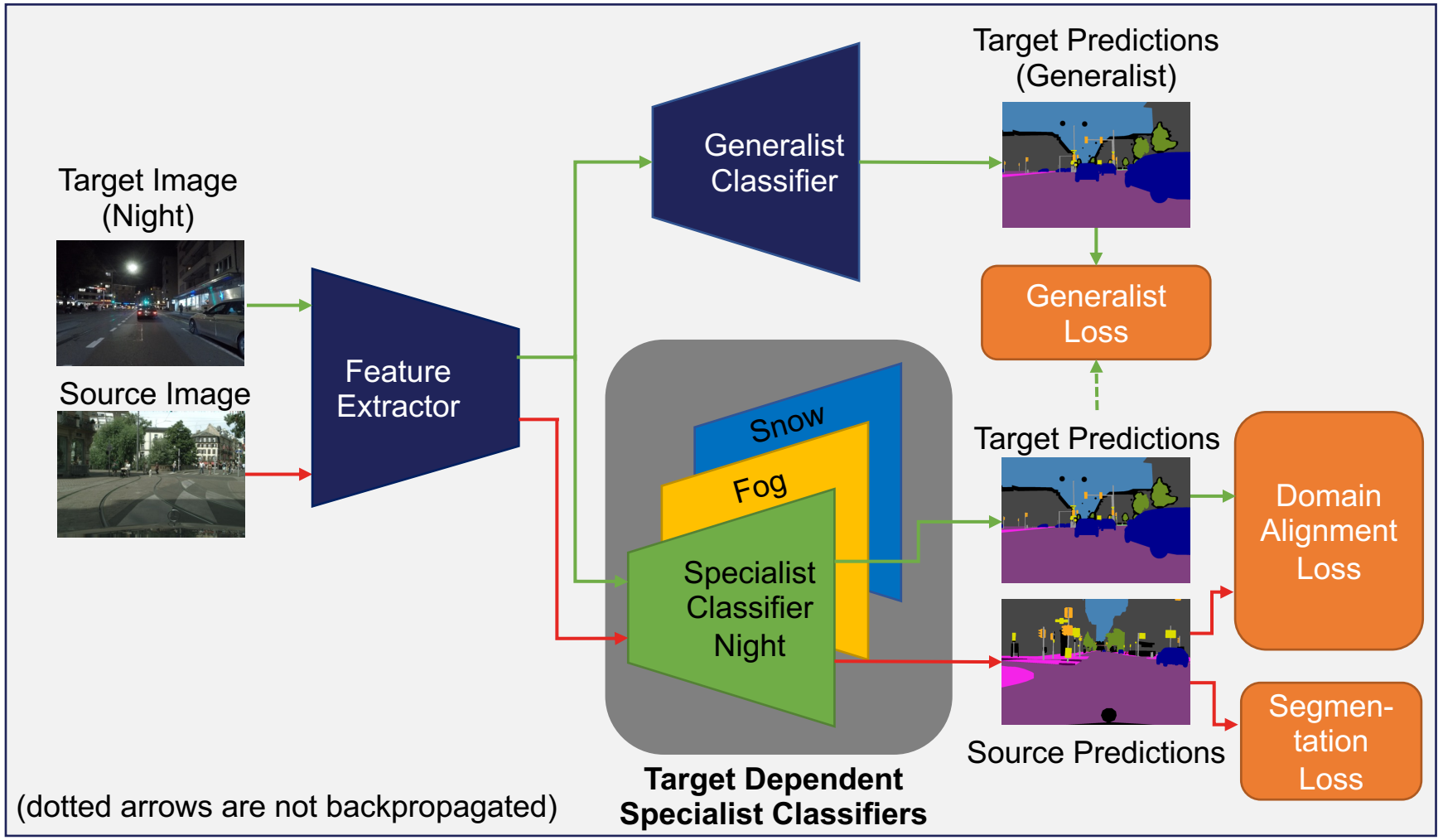
- Minimize discrepancy between source domain predictions and target domain predictions

Continual learning: Target specific memory

- For each target domain, extra memory module parallel to segmentation network
- Memory stores domain discrepancy from current target domain to source domain
- Segmentation network learns domain invariant information and can be used for every domain

ETM framework presented by Kim et al., 2020
Images from Cordts et al., 2016 & Sakaridis et al., 2021

e The MuHDi Framework for Continual UDA: Output Space Alignment to Adapt to New Domains and Multiple Classifiers to Prevent Forgetting



UDA: Output space alignment

- Minimize discrepancy between source domain predictions and target domain predictions

Continual Learning: Target dependent specialist classifiers

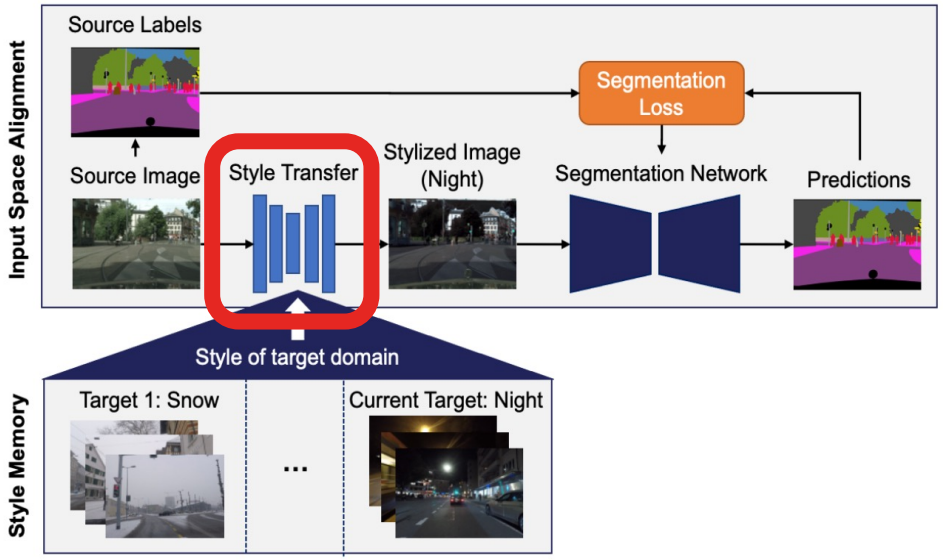
- One specialist classifier trained for each target domain
  - Segmentation performance on source domain
  - Domain alignment between source and target domain
- Knowledge of all specialist classifiers combined in one domain independent generalist classifier

MuHDi framework presented by Saporta et al., 2022
Images from Cordts et al., 2016 & Sakaridis et al., 2021

Experiment 1: Comparison of the 3 Different Style Transfer Algorithms

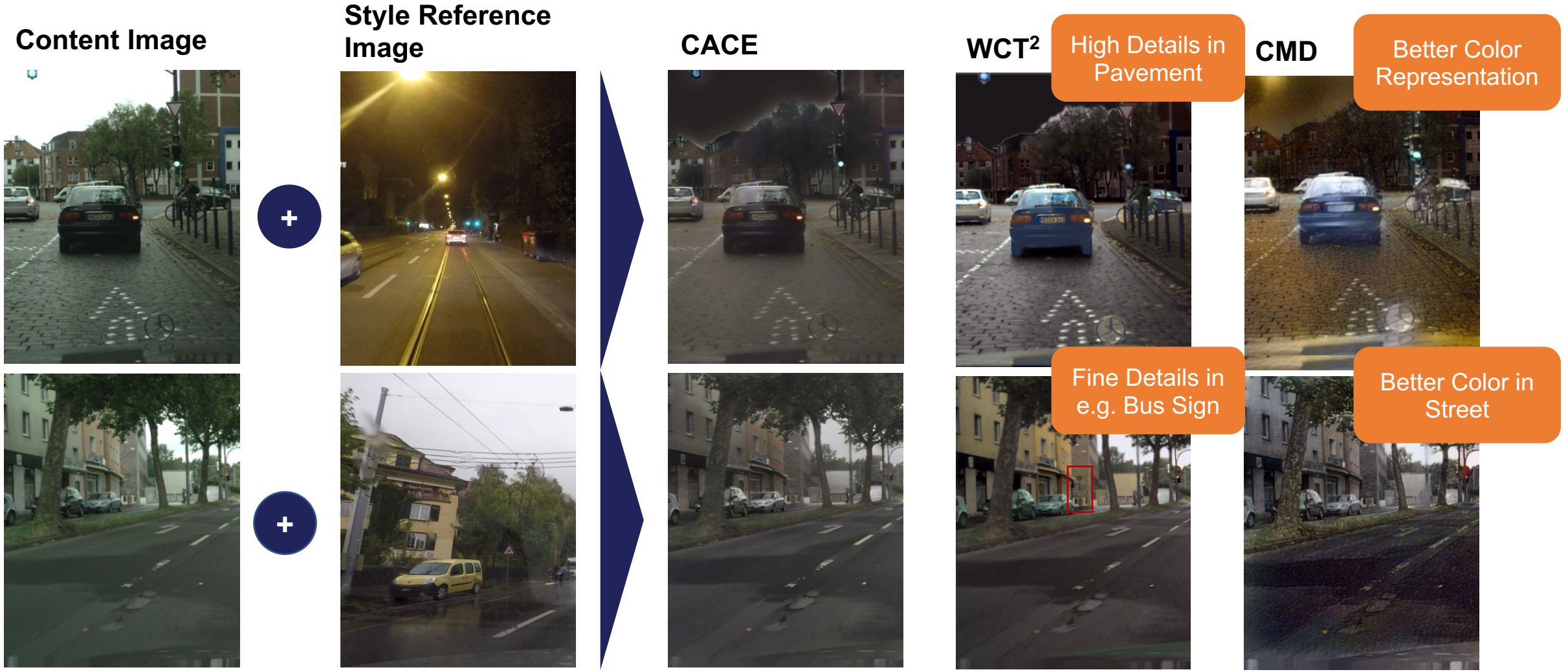
Experiment 1: A Visual Analysis of the Three Style Transfer Algorithms – Are the Expected Enhancements Visible?

Experiment 1 focuses only on the style transfer algorithm



- **I tested the three style transfer algorithms for transferring style in real-world street scenes**
  - CACE's style transfer algorithm
  - WCT^2 – enhanced photorealism
  - CMD – enhanced stylization accuracy
- **Question:** Are the expected enhancements visible in the stylized images?

All Style Transfer Algorithms Stylize Images Differently: WCT² with the Expected Higher Level of Details, CMD with Color Enhancements



Content images from Cordts et al., 2016;
Style images from Sakaridis et al., 2021

Experiment 2: Comparison of All 5 Frameworks for Continual UDA

Experiment 2: Comparison of the Five Frameworks For Their Performance in Continual UDA

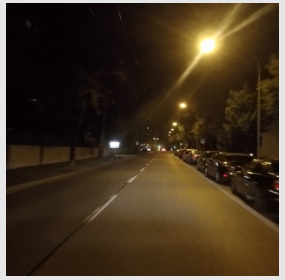
The Cityscapes/ACDC sequence is used to evaluate the performance of all five frameworks for continual UDA

Source Domain: Cityscapes[2]

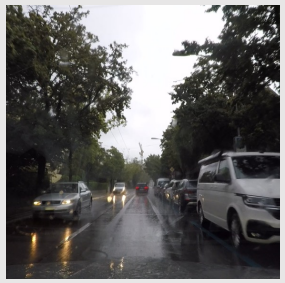
Target Domains: ACDC[3] dataset; 4 domains of different conditions

Target 1: Fog

Target 2: Night



Target 3: Rain



Target 4: Snow

training time

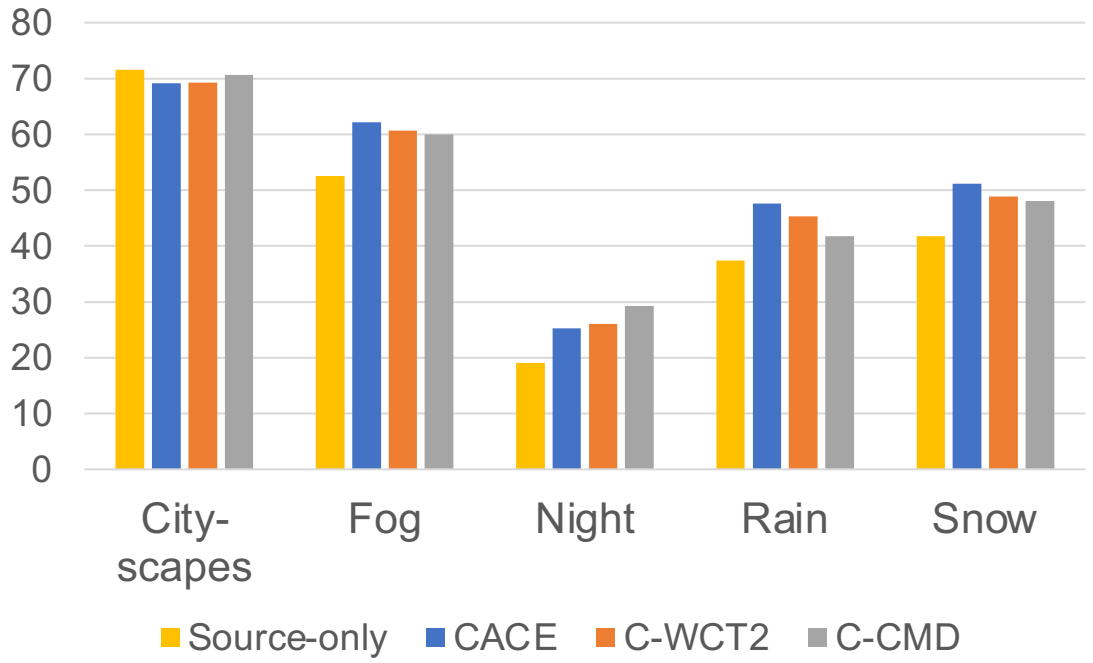
Frameworks compared for **segmentation performance** (mIoU) averaged over **all domains** (mean mIoU) after adapting to the last target domain (Snow)

- I tested the five **frameworks** for **three sequences of domains;** in today's presentation **focusing** on the **Cityscapes/ACDC** sequence[1]
- **2 Questions:**
  - Is there an advantage from the **enhanced style transfer** algorithms?
  - Which approach**, input space alignment or output space alignment**, delivers better performance?

[1]Other Sequences are (1) SYNTHIA: 13 synthetic domains diverging in weather and season, (2) GTA-5/Cityscapes/IDD: synthetic source domain, two real world target domains from Europe and India. Focused on Cityscapes/ACDC because it shows the relevant behavior well.
[2] Presented by Cordts et al., 2016; images taken from them [3] Presented by Sakaridis et al., 2021; images taken from them

More Advanced Style Transfer Algorithms Do Not Yield a Benefit Compared to CACE, CACE Performs Best Averaged over All Domains

Domain-wise performance after adapting to the last target domain [All values measured in mIoU]



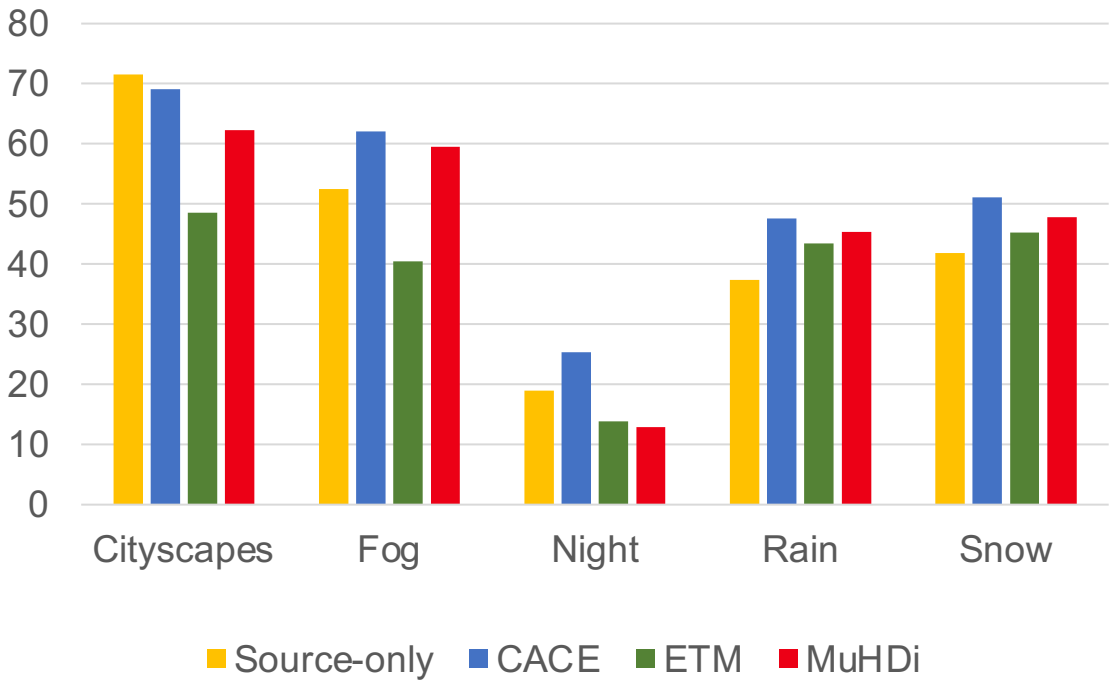
Average performance (in mean mIoU):
- Source-only: 44.4%
- CACE: 51.0%
- C-WCT^2: 50.1%
- C-CMD: 49.9%

- All style-transfer-based frameworks achieved **clear performance improvements** over the source-only baseline
- On average CACE performed best, **no advantage** of using **enhanced style transfer** algorithms WCT^2 & CMD
- Relative performance is **dependent on domain**
  - No style transfer algorithm best over all domains
  - CACE best for three out of the four target domains (Fog, Rain, Snow)
- Additional disadvantage: C-WCT^2 and C-CMD suffer from **limitations in implementation**[1]
  - Longer runtime & increased memory consumption

[1] Explained in detail in my bachelor thesis

Output Space Aligned Frameworks ETM & MuHDi Show Significantly Worse Performance than the Input Space Aligned CACE Framework

Domain-wise performance after adapting to the last target domain. [All values measured in mIoU]



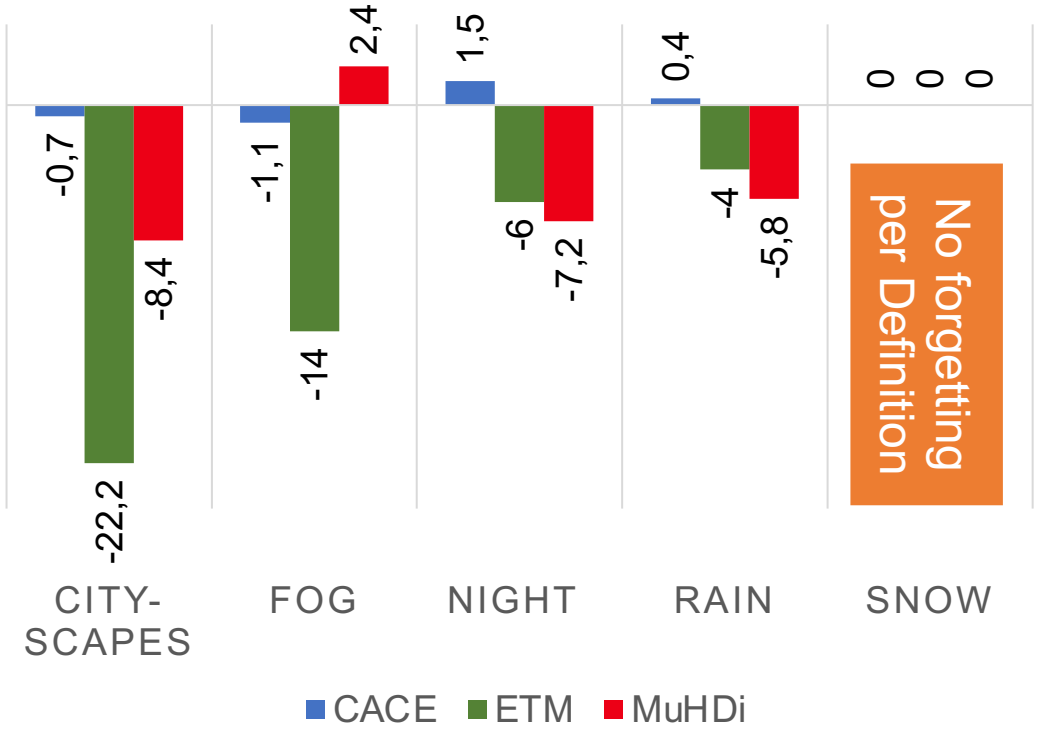
Average performance (in mean mIoU):
- Source-only: 44.4%
- CACE: 51.0%
- ETM: 38.3%
- MuHDi: 45.6%

- **Input space aligned frameworks** (CACE) **better** than **output space aligned frameworks** (ETM, MuHDi) for **every domain**
- Output space aligned frameworks with **unsatisfactory performance**
  - ETM on average 6% points below source-only
  - MuHDi better but still only in the region of source-only
- ETM & MuHDi with **adequate performance** for **later trained domains** (Rain, Snow), however, performance for the **early trained domains** (Cityscapes, Fog, Night) **is poor**

→ **Assumption**: Forgetting leads to poor performance on early trained domains

ETM & MuHDi Both Suffer from Forgetting, They Cannot Maintain Performance when Adapting to New Domains

Illustration of forgetting. For each domain, the difference between the performance immediately after adapting to this domain and the performance after adapting to the last target domain is illustrated. [All values measure in % points ΔmIoU]



- **Major problem: Forgetting of old knowledge when adapting to new target domains**
  - ETM & MuHDi both with large performance drops for previous domains when adapting to a new target domain
  - CACE prevents forgetting well, for no domain significant loss in performance after adapting to additional domains

Conclusion

Conclusion: The CACE Framework is the Currently Best Approach for Continual UDA in Semantic Segmentation

Result 1:

Comparison of different algorithms for style transfer

- **CACE performed well** and achieves clear improvements over training only on the source domain
- **No advantage** from using the more advanced style transfer algorithms WCT^2 and CMD
- Both more advanced algorithms additionally lead to **limitations in implementation**

Result 2:

Comparison of input space and output space aligned frameworks

- Output space aligned frameworks (ETM, MuHDi) **performed poorly**
- Output space alignment **effective for adapting to only one target domain (UDA)**
- Both frameworks **incapable of sufficiently preventing forgetting** when adapting to additional domains

Out of the five, the **CACE** framework is the currently **best approach** for **continual UDA** in **semantic segmentation**

Extras

Performance Values for the Cityscapes/ACDC sequence

Domain-wise performance after adapting to the last target domain.
[All values measure in mIoU]

	Cityscapes	Fog	Night	Rain	Snow	mean mIoU
Source-only	**71.5**	52.5	19.0	37.4	41.8	44.4
CACE	69.1	**62.1**	25.3	**47.6**	**51.1**	**51.0**
C-WCT^2	69.3	60.7	26.1	45.3	48.9	50.1
C-CMD	70.6	60.0	**29.3**	41.7	48.1	49.9
ETM	48.5	40.5	13.8	43.4	45.3	38.3
MuHDi	62.3	59.5	12.9	45.5	47.8	45.6

Performance Values for the Cityscapes/ACDC Sequence

Illustration of pure UDA capabilities: Performance Values immediately after adapting to each target domain are illustrated for every framework. [All values measure in mIoU]

(a) CACE

	CS	Fog	Night	Rain	Snow
CS	**71.5**	52.5	19.0	37.4	41.8
Fog	72.4	**61.9**	19.0	37.4	41.8
Night	70.6	57.6	**25.8**	44.8	45.9
Rain	69.1	59.0	27.3	**45.0**	46.1
Snow	70.8	60.8	27.3	46.0	**49.9**

(b) ETM

	CS	Fog	Night	Rain	Snow
CS	**70.7**	52.8	16.4	38.9	40.8
Fog	66.4	**54.4**	15.5	39.7	40.0
Night	63.4	48.9	**19.2**	37.9	38.4
Rain	56.5	46.1	17.7	**47.1**	39.7
Snow	48.5	40.5	13.8	43.3	**45.3**

(c) MuHDi

	CS	Fog	Night	Rain	Snow
CS	**70.7**	52.8	16.4	38.9	40.8
Fog	67.4	**57.1**	16.8	41.8	42.0
Night	66.8	56.3	**20.1**	41.6	40.1
Rain	66.6	60.8	15.3	**51.1**	47.2
Snow	62.3	59.5	12.9	45.4	**47.8**

Performance Values for the Cityscapes/ACDC Sequence

Illustration of pure UDA capabilities with adapted Implementation: Performance Values immediately after adapting to each target domain are illustrated for adapted CACE and C-CMD. [All values measure in mIoU]

	Fog	Night	Rain	Snow
CACE	58.0	24.6	46.6	47.2
C-CMD	59.6	29.2	42.4	48.1

Performance Values for the Synthetic SYNTHIA Sequence

Domain-wise performance after adapting to the last target domain.
[All values measure in mIoU]

	Dawn	Fall	Fog	Night	Rainnight	Softrain	Spring	Summer	Sunset	Winter	Winternight	mean mIoU
Source-only	70.5	56.5	58.6	53.7	26.2	37.1	56.6	47.6	66.0	40.0	40.7	50.3
Jitter	70.3	62.4	64.0	62.3	29.2	38.7	57.5	53.8	70.6	40.4	49.7	54.4
CACE	**71.4**	69.0	**68.5**	66.5	49.1	57.9	67.3	69.7	71.3	53.7	59.1	63.9
C-WCT^2	71.1	69.6	67.6	68.8	42.5	51.4	68.5	71.4	**71.9**	58.2	60.1	63.7
C-CMD	69.1	66.5	66.6	65.8	**51.0**	**59.1**	66.3	68.0	69.3	60.0	57.5	63.3

Performance Values for the Synthetic to Real GTA5/Cityscapes/IDD Sequence

Domain-wise performance after adapting to the last target domain.
[All values measure in mIoU]

	GTA-5	Cityscapes	IDD	mean mIoU
Source-only	76.2	33.8	41.2	50.4
Jitter	75.0	40.6	42.7	52.8
CACE	**76.2**	**45.0** (-0.2)	44.6	**55.3**
C-WCT^2	76.1	44.0 (-1.5)	43.5	54.5
ETM	66.6	38.7 (-2.5)	**45.5**	50.3
MuHDi	71.2	33.6 (-6.0)	40.0	48.3