MulT: An End-to-End Multitask Learning Transformer

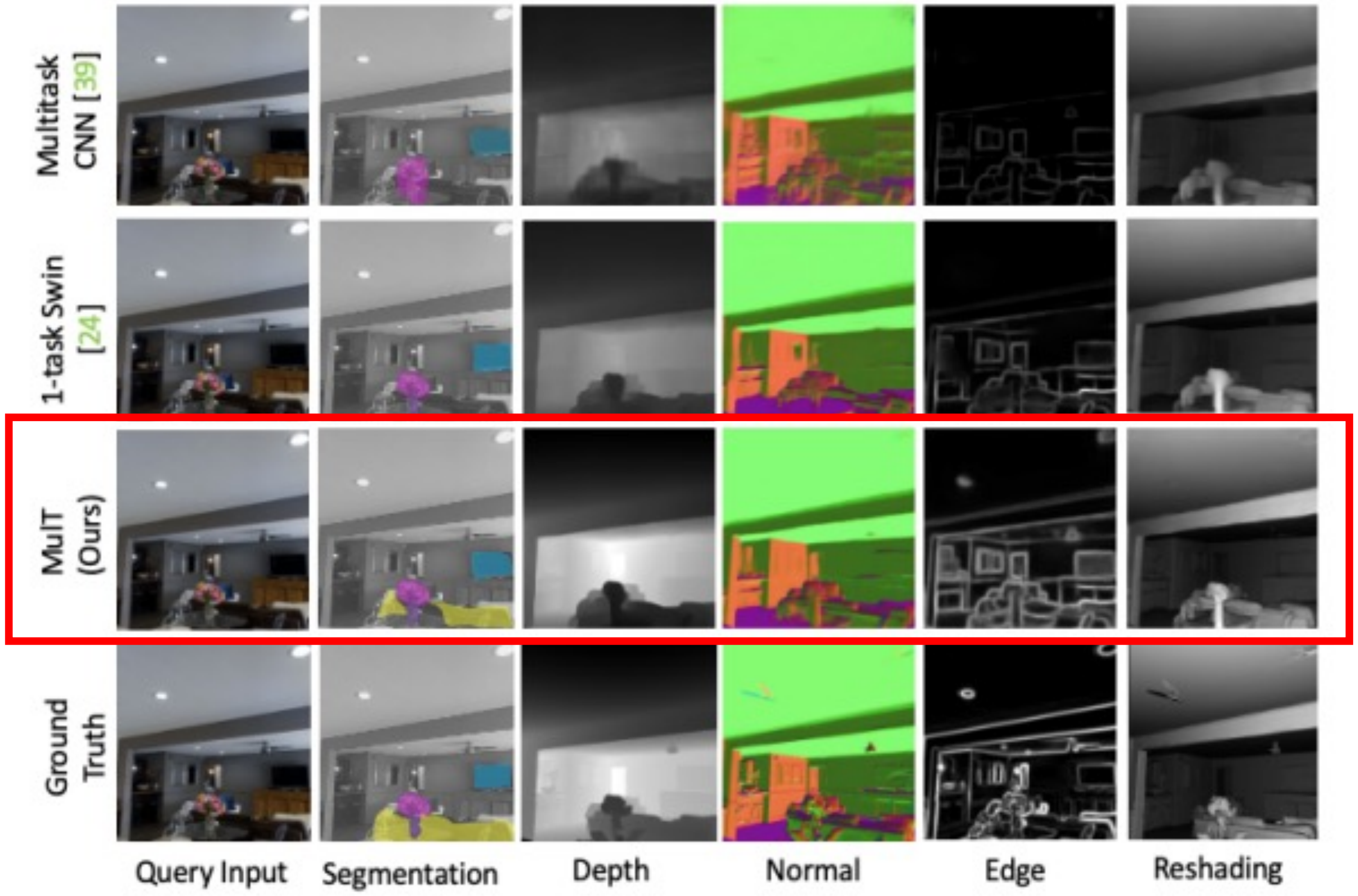
Author: Deblina Bhattacharjee, Tong Zhang,
Sabine Susstrunk, Mathieu Salzmann
From: School of Computer and Communication Sciences, EPFL, Switzerland
Publication: CVPR2022
URL: https://ivrl.github.io/MulT/

Contents

- What will be possible?
- Background
- Related Work
- MulT: A Multitask Transformer
  - Encoder
  - Decoder
  - Shared Attention
  - Task head and Loss
- Experiment Details
- Results
- Conclution
- Future Work

What will be possible?

MulT



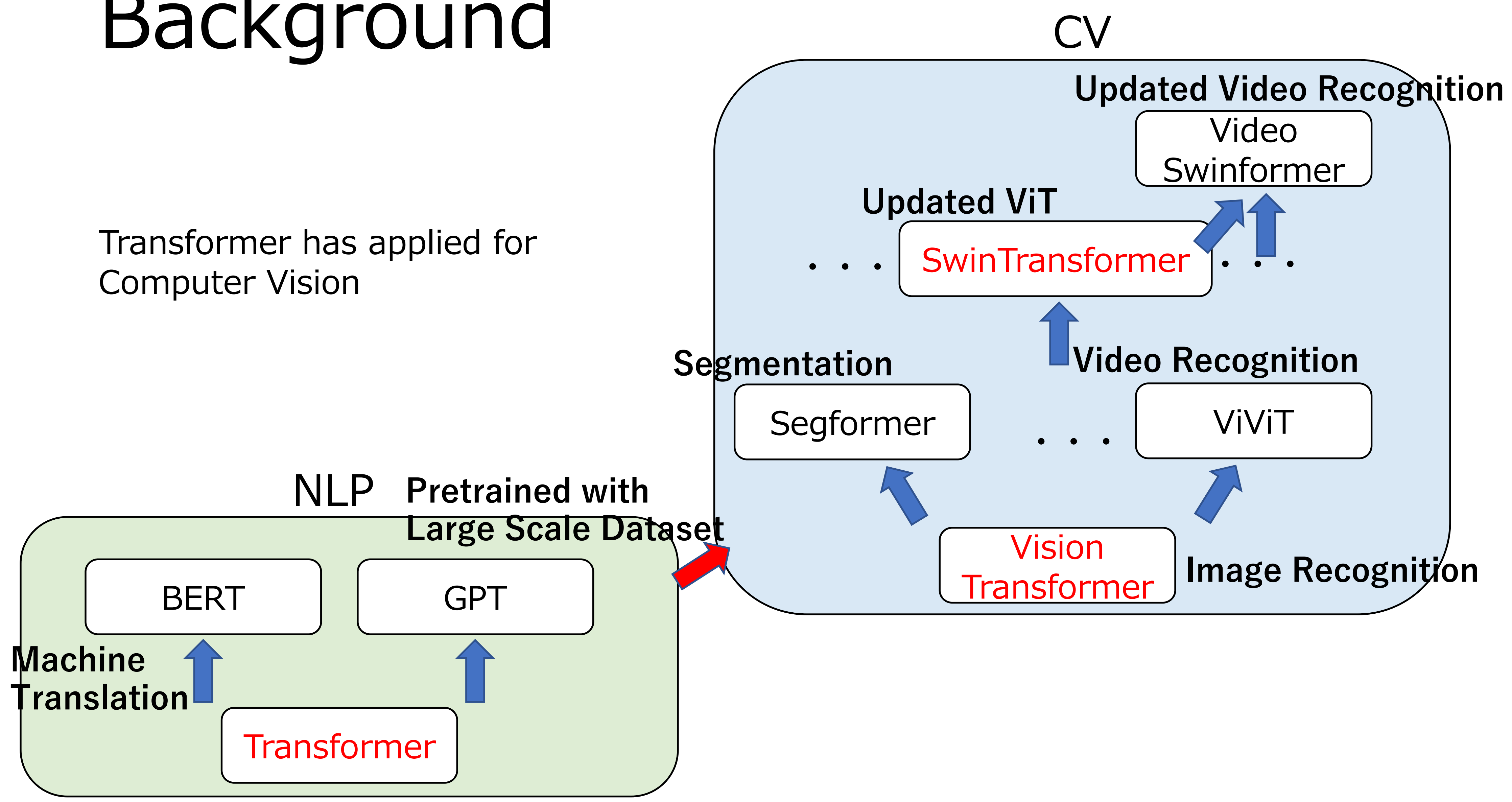
"End-to-End Multitask LearningTransformer"

<u>・Can perform multiple tasks with high accuracy</u>

e.g.
Segmentation, Depth Estimation, Normal Surace Estimation, Extract edge, Reshading

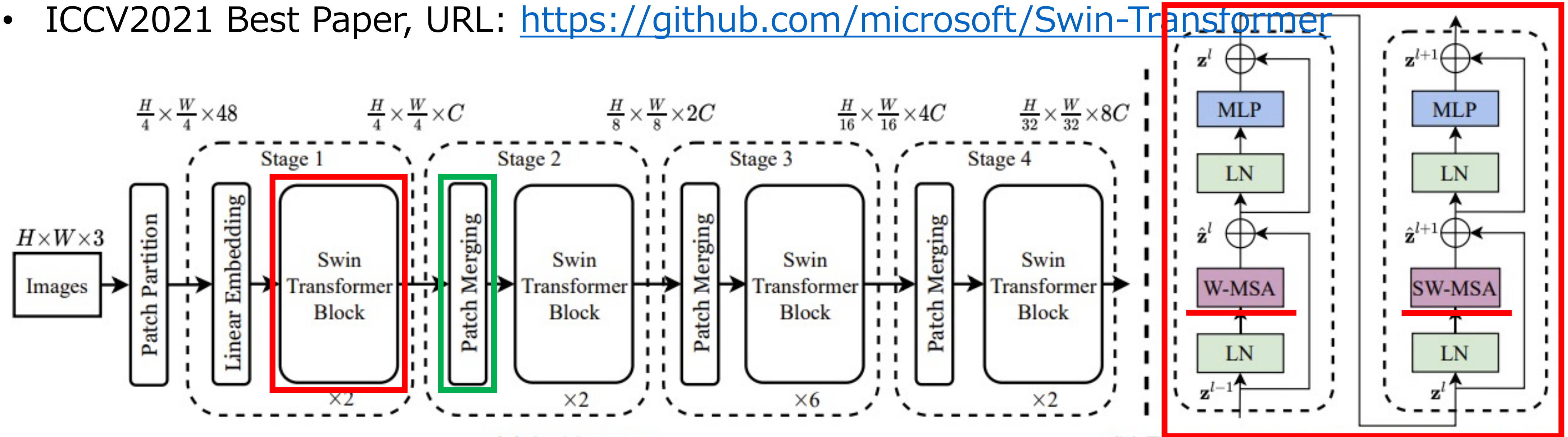
Background

Transformer has applied for Computer Vision



Background: Swin Transformer

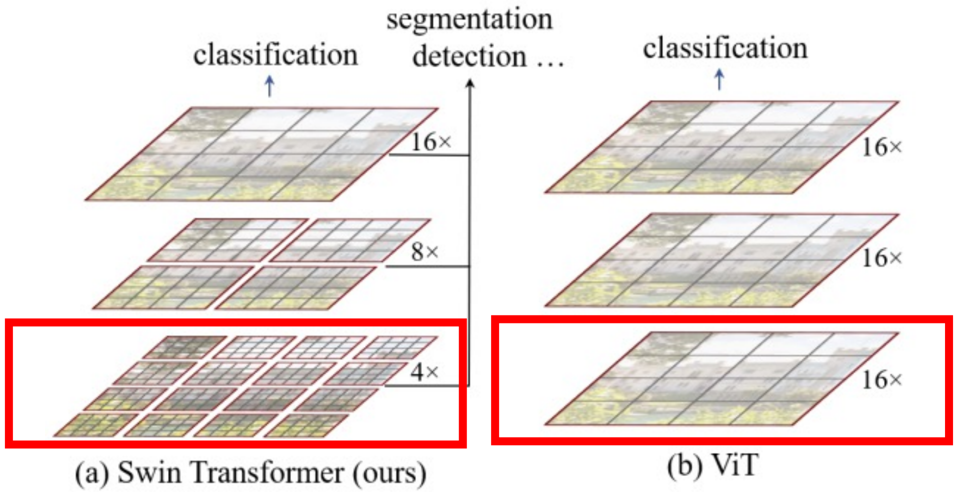
- **Swin Transformer: Hierarchical Vision Transformer using Shifted Windows**
- Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo
- ICCV2021 Best Paper, URL: https://github.com/microsoft/Swin-Transformer



(a) Architecture (b) Two Successive Swin Transformer Blocks

- Key Points
  - ✓ Shift Window-based Multi head self-attention
  - ✓ Patch Merging

Window-based Multi head self-attention



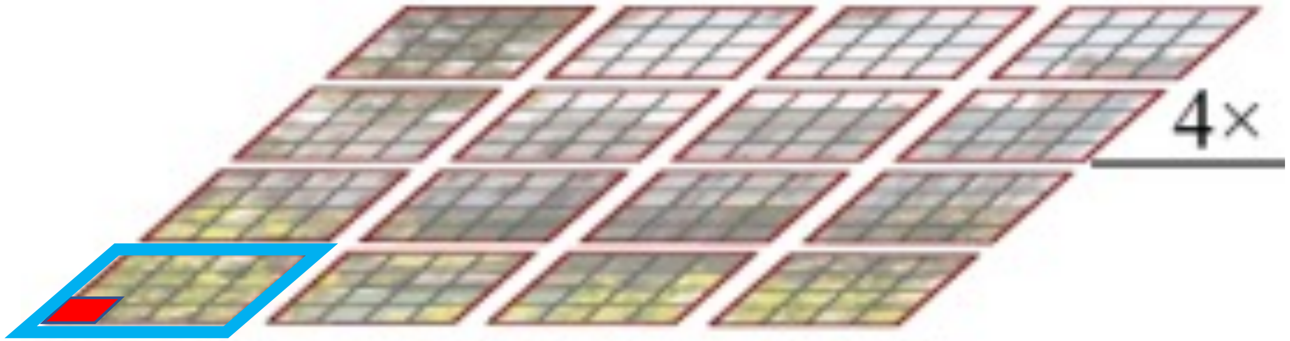
Patch: 4x4x3

Patch: 16x16x3

Output each stage

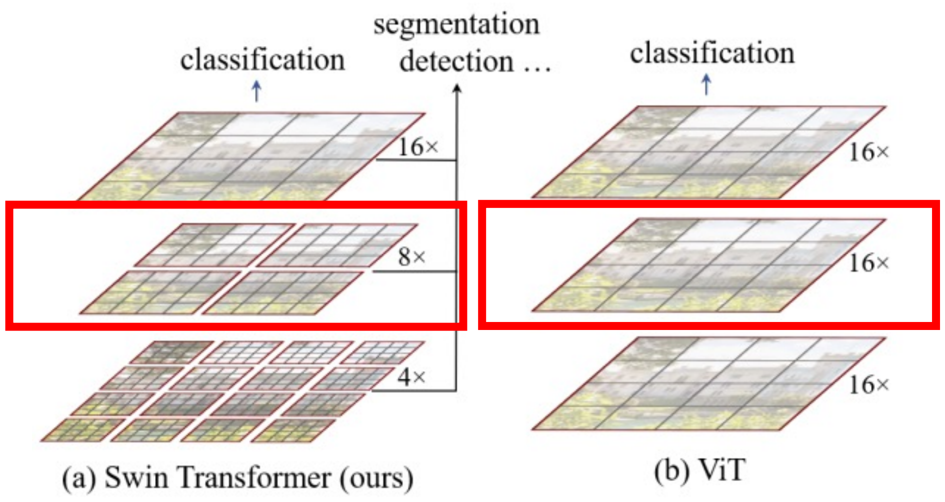
$$H \times W \times 3 \rightarrow \frac{H}{4} \times \frac{W}{4} \times 48$$

ViT: Patch and Window to perform self-attention



Swin: Patch and Window to perform self-attention

Window-based Multi head self-attention

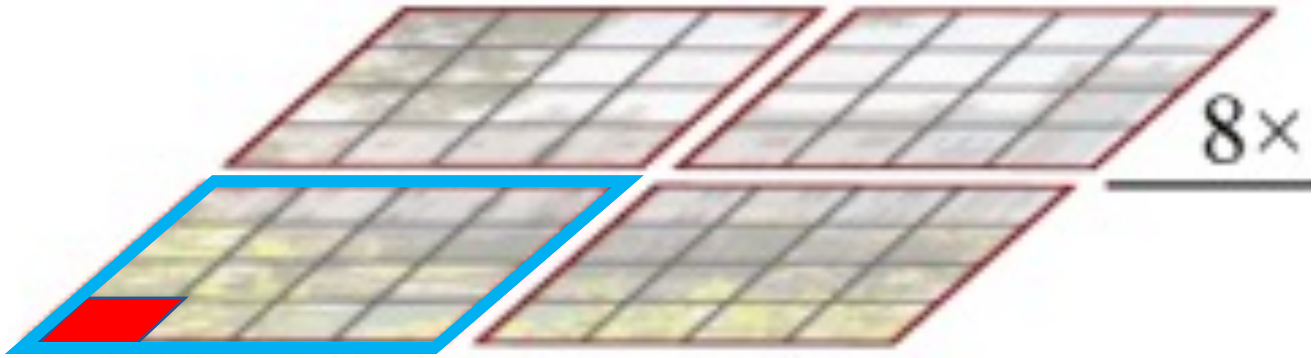


Patch: 4x4x3

Patch: 16x16x3

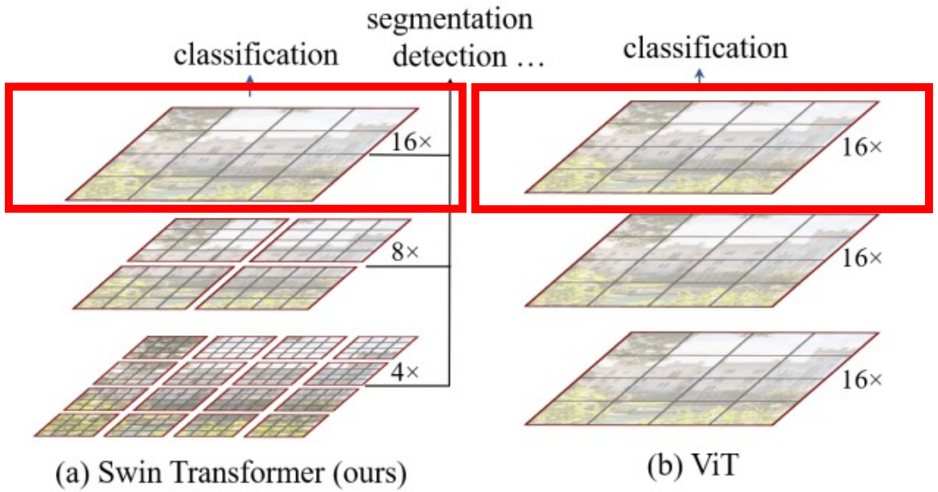
$$\frac{H}{4}\times\frac{W}{4}\times C \rightarrow \frac{H}{8}\times\frac{W}{8}\times 2C$$

ViT: Patch and Window to perform self-attention



Swin: Patch and Window to perform self-attention

Window-based Multi head self-attention



Patch: 4x4x3

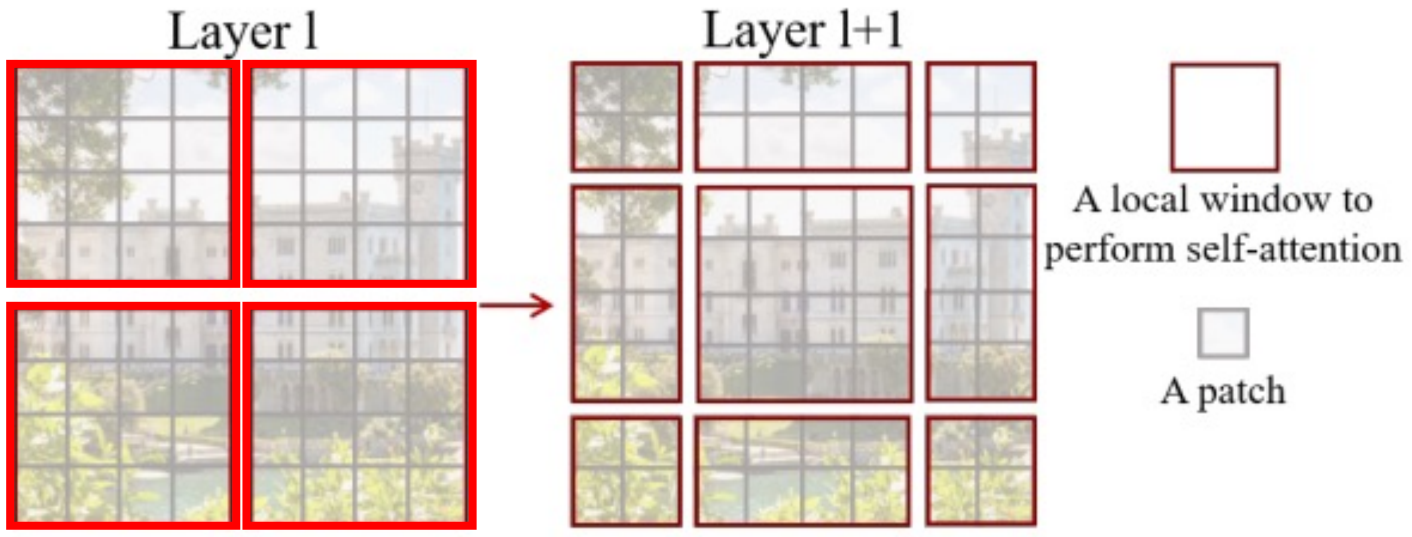
Patch: 16x16x3

$$\frac{H}{8}\times\frac{W}{8}\times 2C \rightarrow \frac{H}{16}\times\frac{W}{16}\times 4C$$

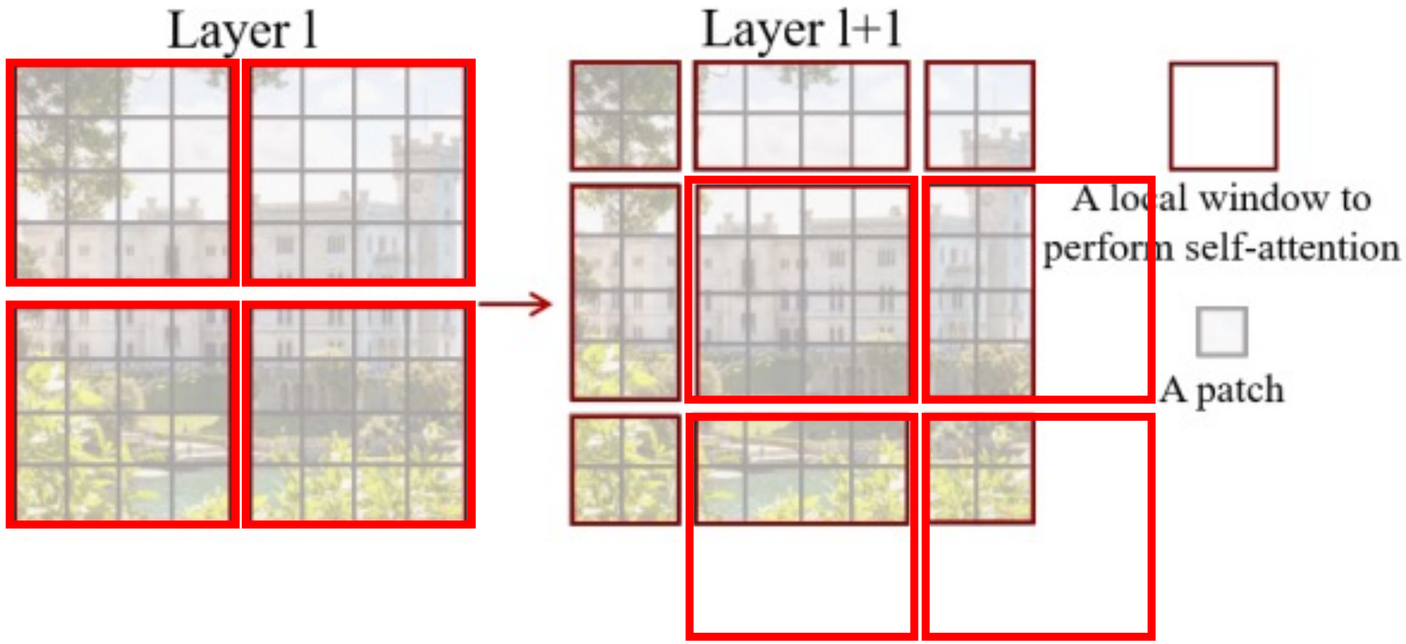
ViT: Patch and Window to perform self-attention

Swin: Patch and Window to perform self-attention

Shift Window-based Multi head self-attention

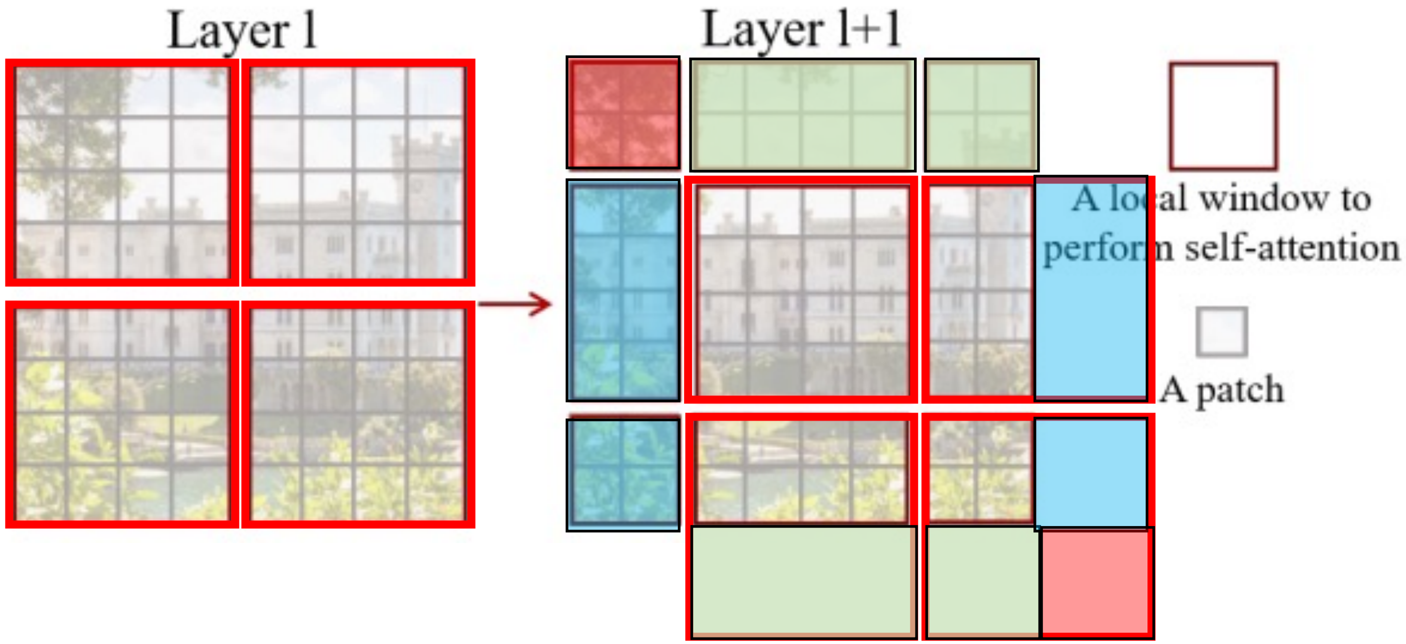


Shift Window-based Multi head self-attention



Shift Window-based Multi head self-attention

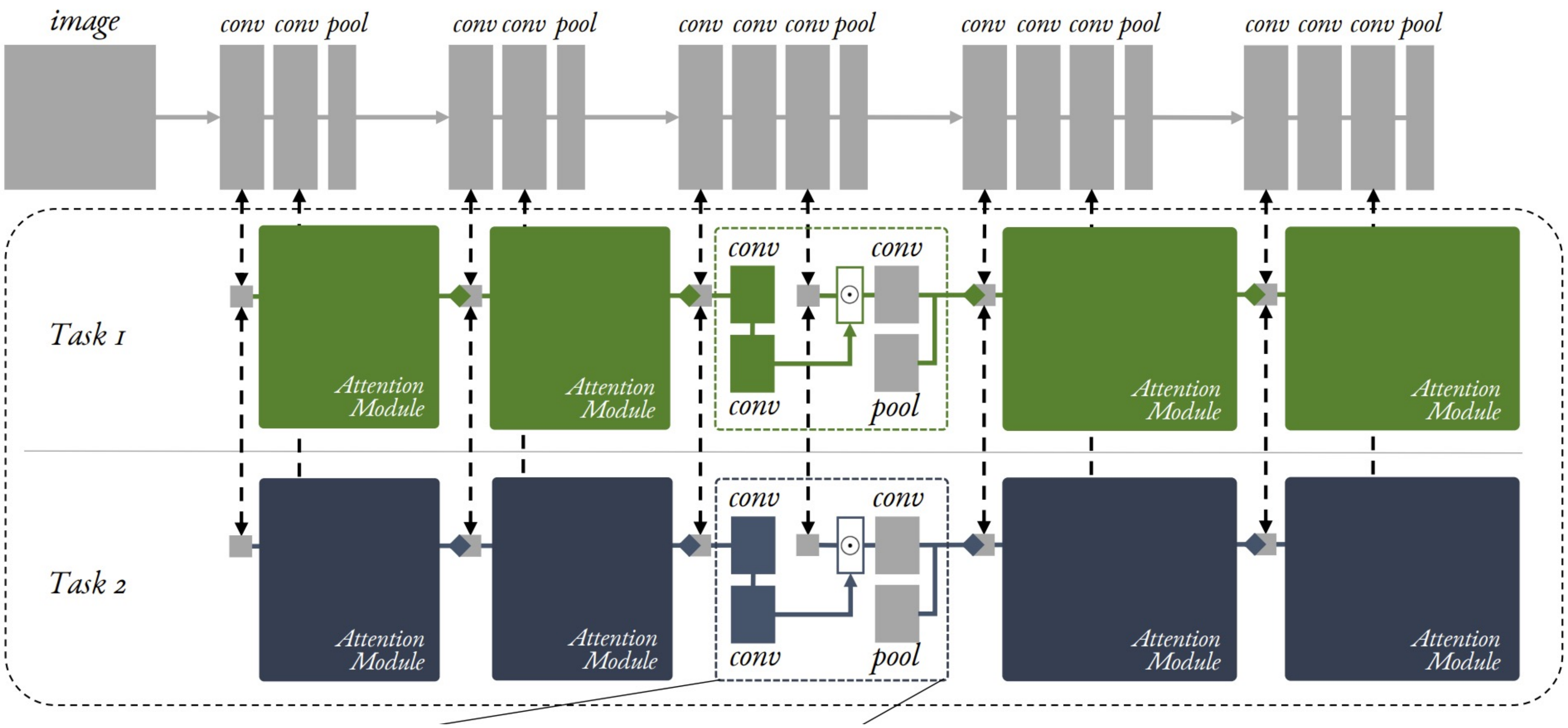
Cyclic shift



The overflow area created when shifting is moved as shown in the figure.

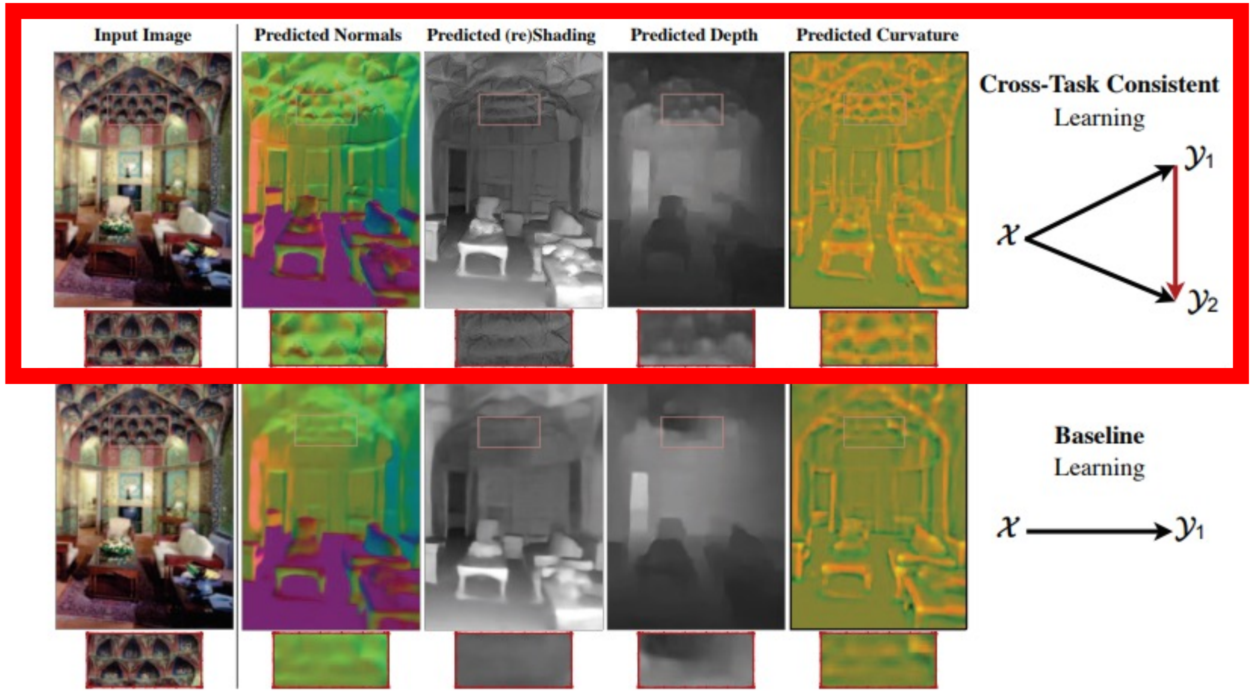
Related Work “Multitask Learning” in ComputerVision

- **End-to-End Multi-Task Learning with Attention**
  - Shikun Liu, Edward Johns, Andrew J. Davison
  - CVPR2019, PaperURL: https://arxiv.org/abs/1803.10704



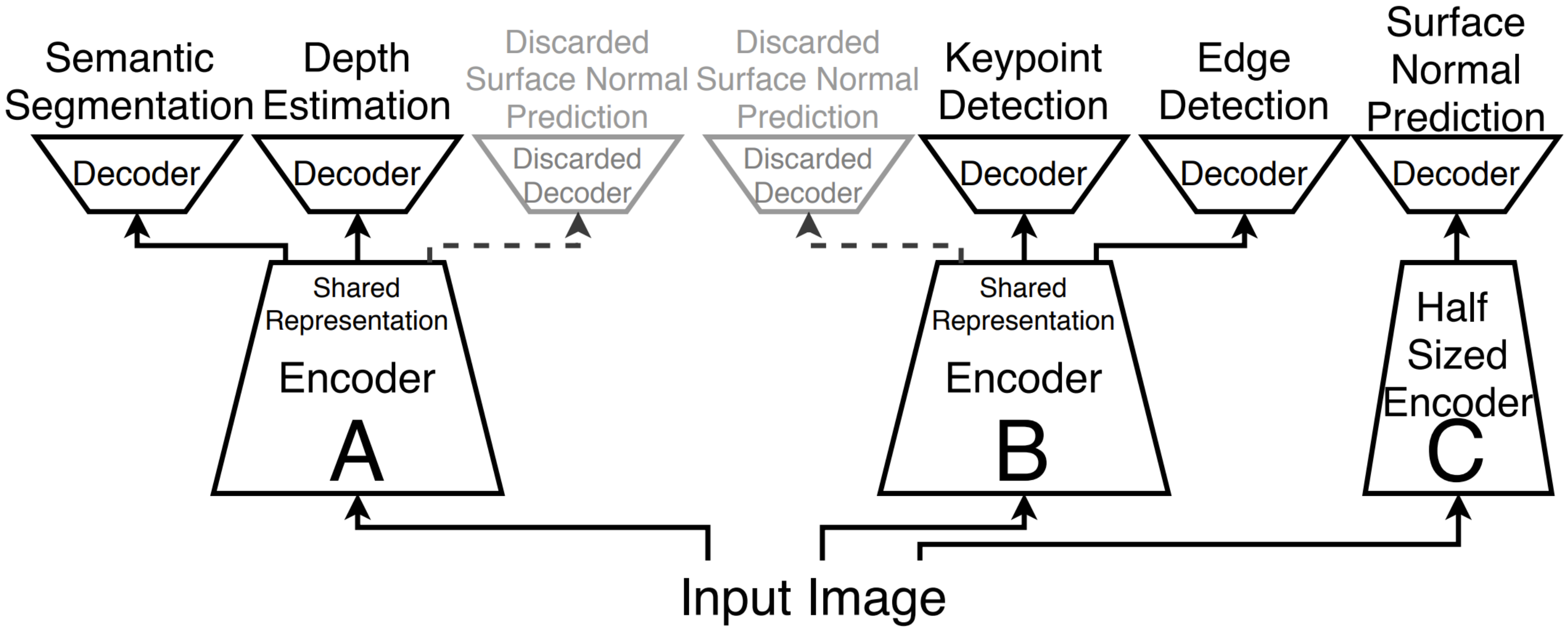
Related Work

- **Robust Learning Through Cross-Task Consistency**
  - Amir Zamir, Alexander Sax, Teresa Yeo, Oğuzhan Kar, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, Leonidas Guibas
  - CVPR2020, PaperURL: https://arxiv.org/abs/2006.04096



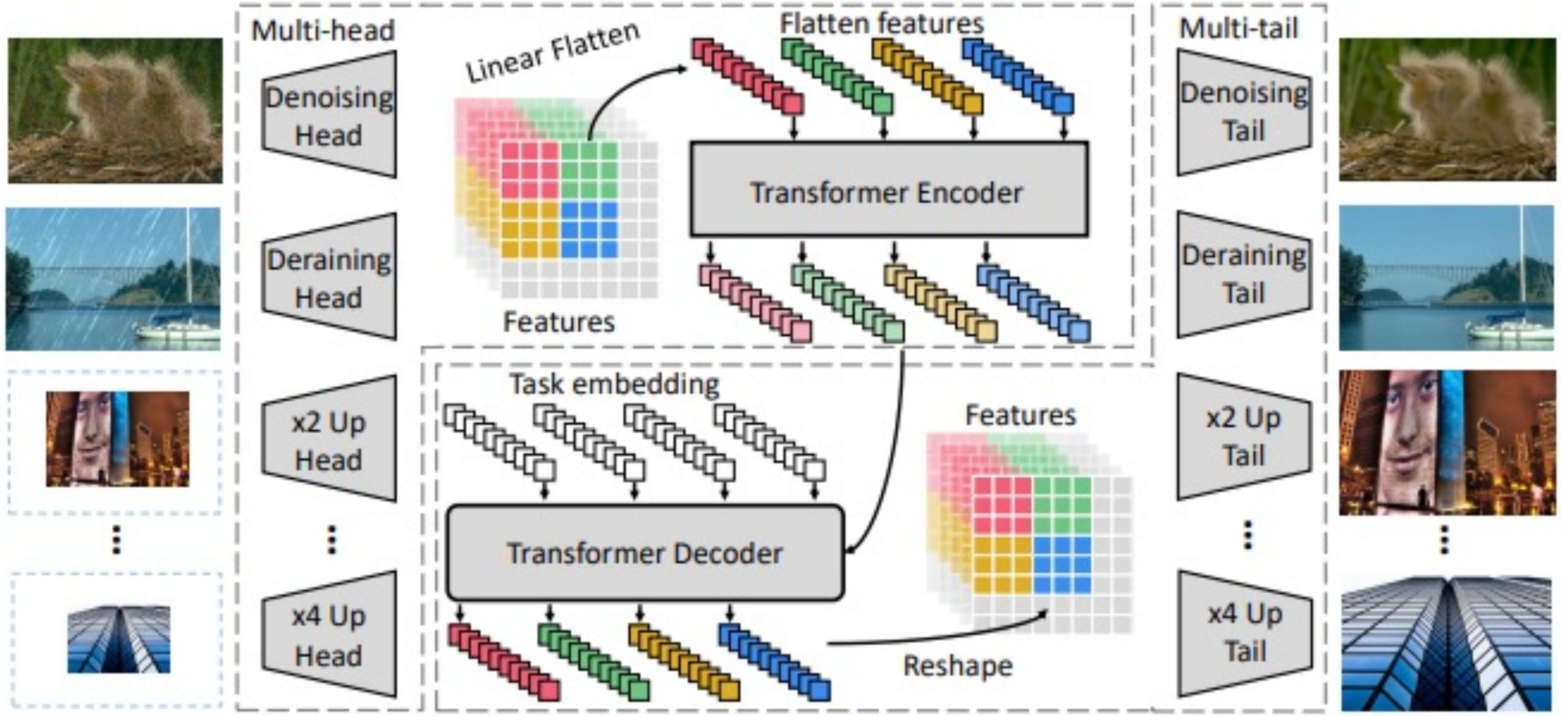
Related Work

- **Which Tasks Should Be Learned Together in Multi-task Learning?**
  - Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas,
  - Jitendra Malik, Silvio Savarese
  - ICML2020, PaperURL: https://arxiv.org/abs/1905.07553



Related Work

- **Pre-Trained Image Processing Transformer**
  - Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, Wen Gao
  - CVPR2021, URL:[GitHub - dongyan007/Pretrained-IPT-main-master]()



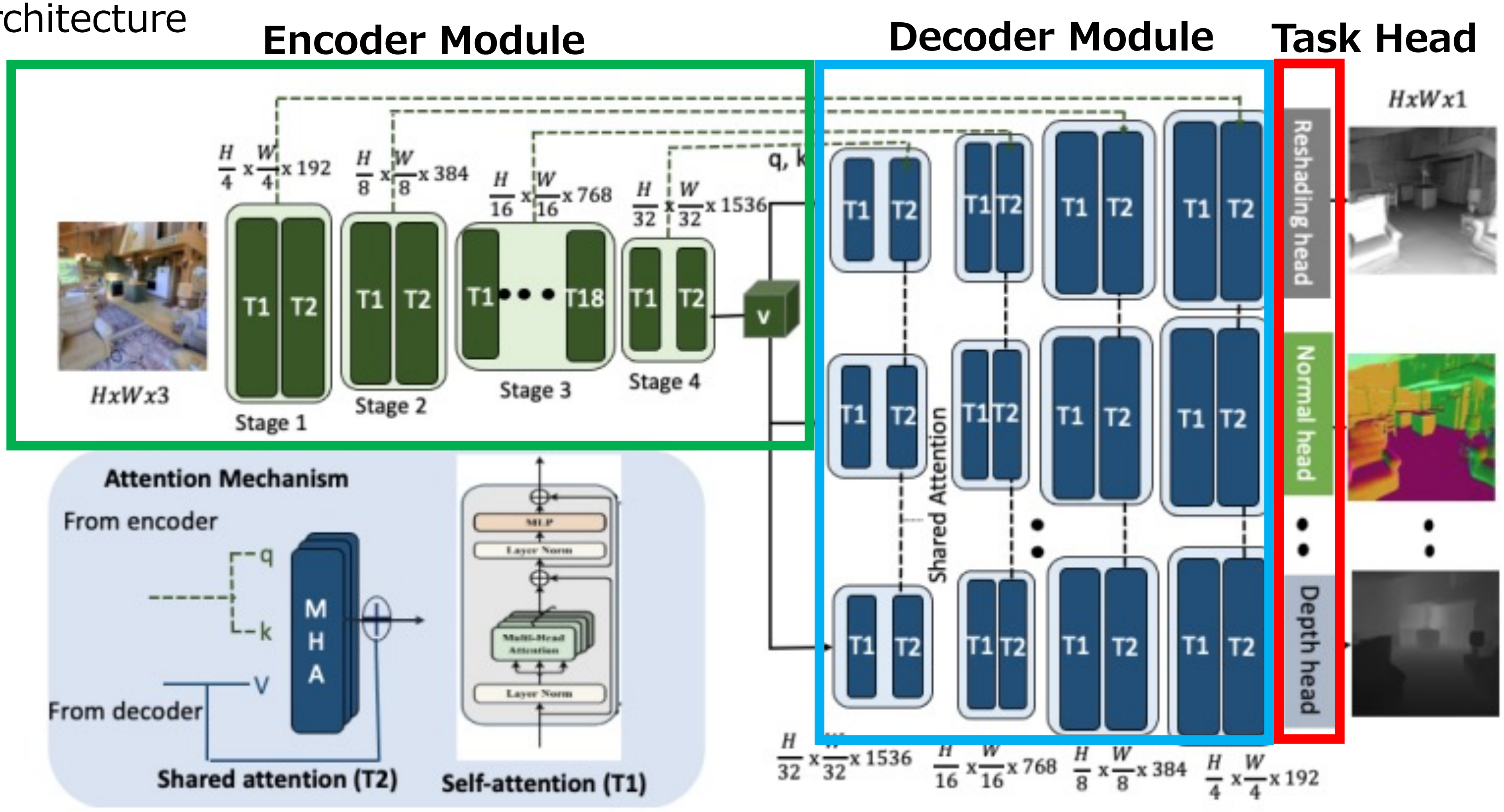
Related Work

- In previous studies, multitasking was trained with CNN-base and Transformer was not used.
- In addition, although multitask learning models with Transformer existed, they did not reflect the dependencies among tasks.

- Therefore, in this study, they attempted to improve the accuracy of multitask learning by introducing a shared attention mechanism.

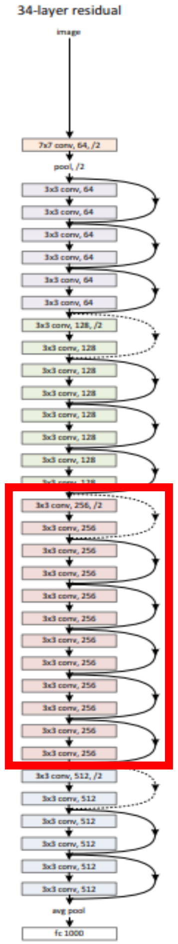
MulT: A Multitask Learning Transformer

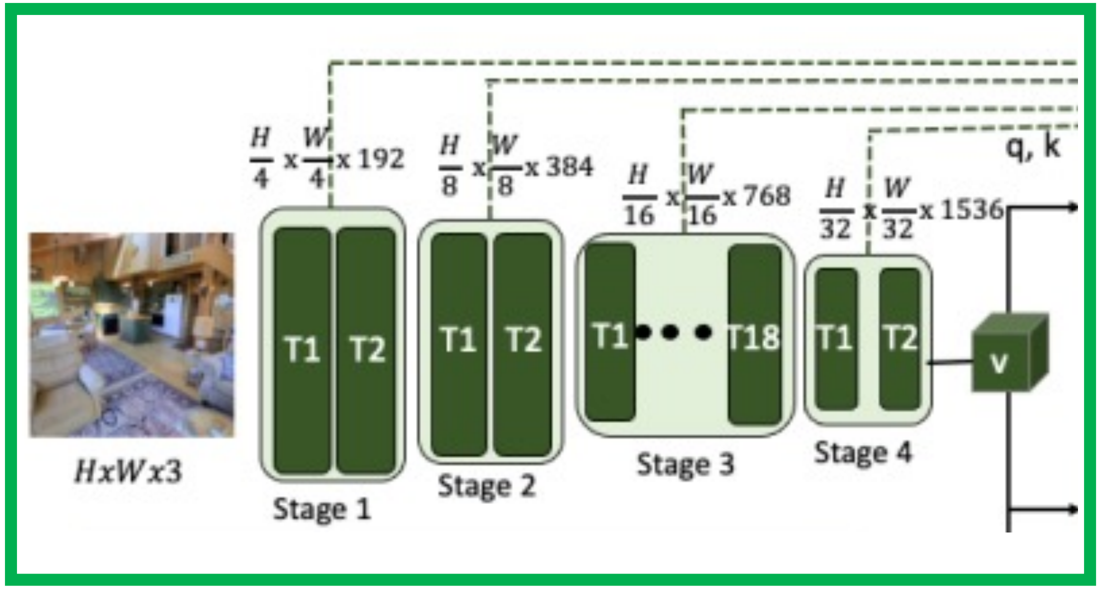
Model Architecture

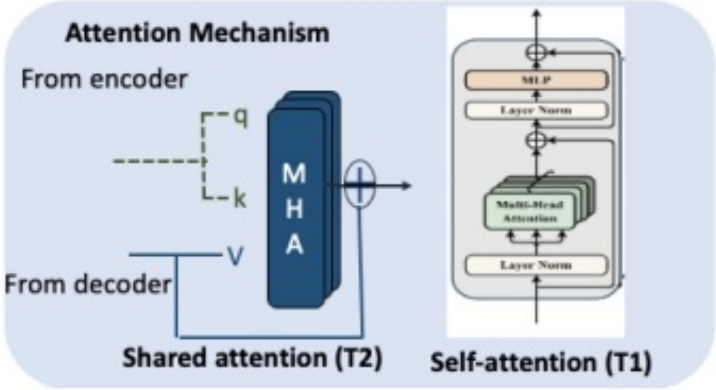


Encoder Module

ResNet [*]



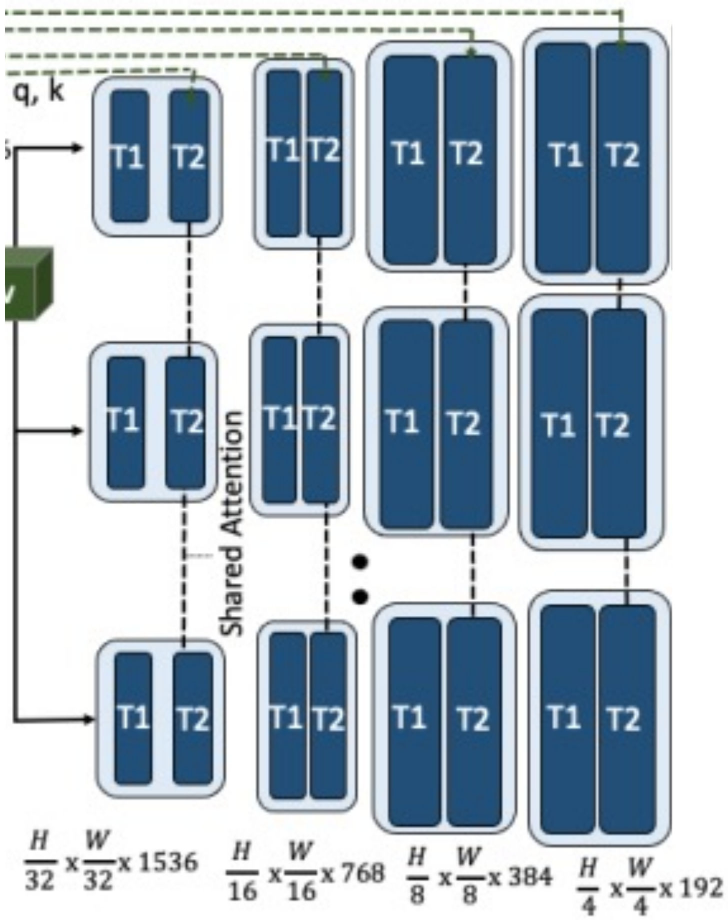


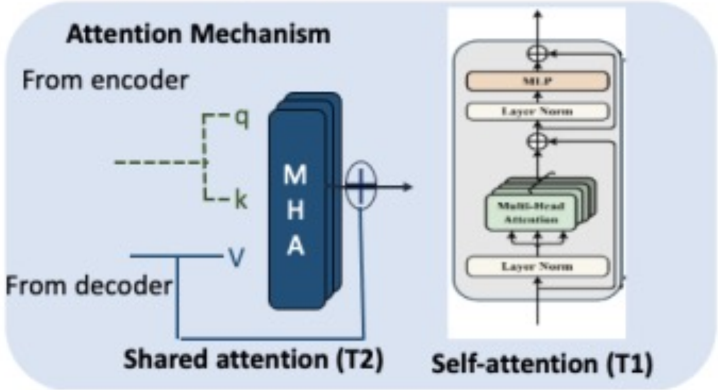


Key Points
- ✓ 4 Stage and Calculations at different resolutions
- ✓ q,k obtained at each stage are sent to the decoder

[*] Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, CVPR2016

Decoder Module

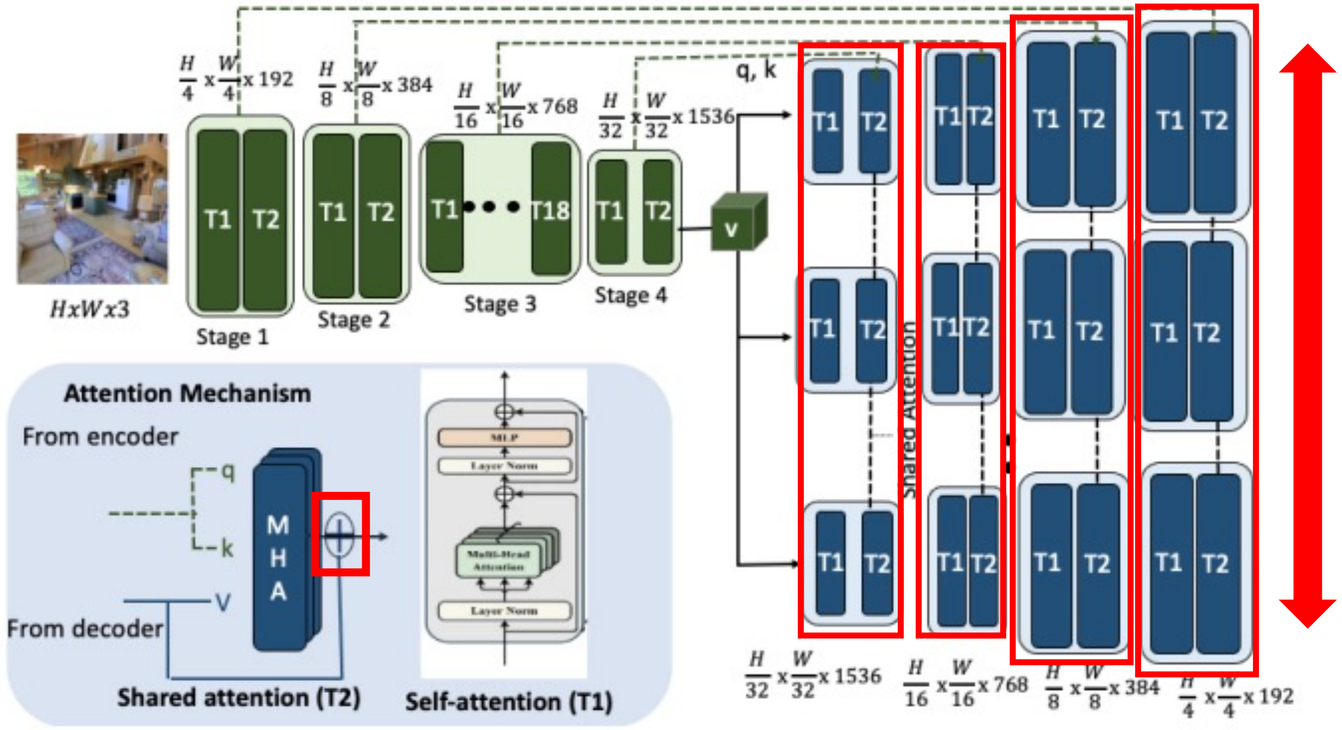




Key Points

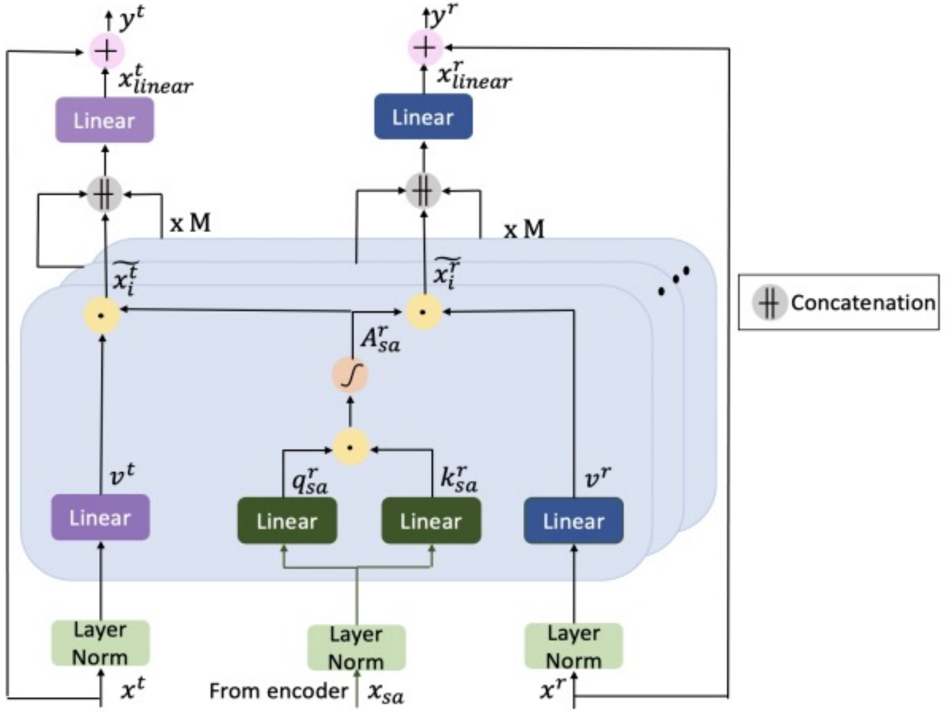
- ✓ 4 Stage and Calculations at different resolutions
- ✓ The feature map sent by skip-connection from the encoder matches the shape at the decoder.
- ✓ Shared Attention → Next Slide

Shared Attention

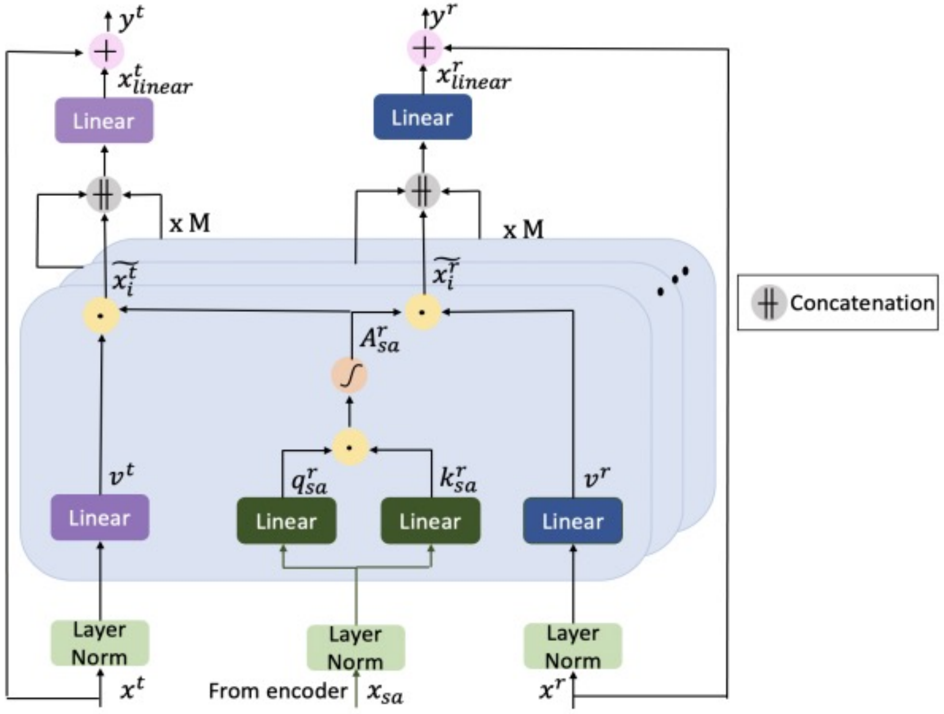


Shared Attention

✓ The same resolution features can be shared.



Shared Attention



Legend

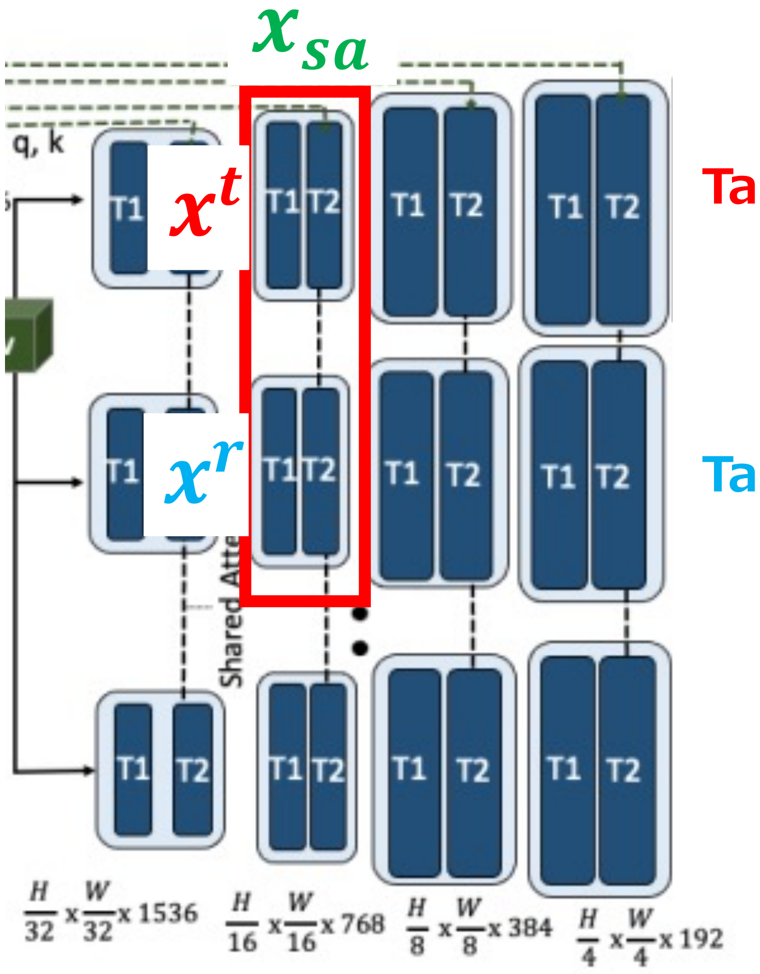
x^t: *upsampled output of the previous stage*
x_{sa}: *output of the encoding stage operating at the same resoluti*
t: *task*
q^r_{sa}: *query from* x_{sa} *by using the liner layer*
k^r_{sa}: *key from* x_{sa} *by using the liner layer*
r: *one paticular task*
v^t: *computed by* x^t *for task* t
C^r: *number of channels*
B^r: *relative position bias*
$\tilde{x}^t = A^r_{sa} v^t$: *self* $-$ *attention head*
$head^t_i(\tilde{x}^t_i, W^t_i) = \tilde{x}^t_i \cdot W^t_i$
W^t_i: *learnt attention weight for task* t *and* $\tilde{x}^t_i$ *is the* i *th channel*
i^{th}: *instance of the self* $-$ *attention, which is repeated* M *times*
x^t_{linear}: *linearly projecting the output of* $MHA^t(.,.)$
y^t: *last output*

$$A^r_{sa} = softmax\left(\frac{q^r_{sa} \cdot {k^r_{sa}}^T}{\sqrt{C^r_{qkv}}} + B^r\right)$$

$$MHA^t(\tilde{x}^t, W) = Concat(head^t_1, \ldots, head^t_M)W,$$
$$x^t_{linear} = MHA^t(.,.),$$
$$y^t = x^t + x^t_{linear}$$

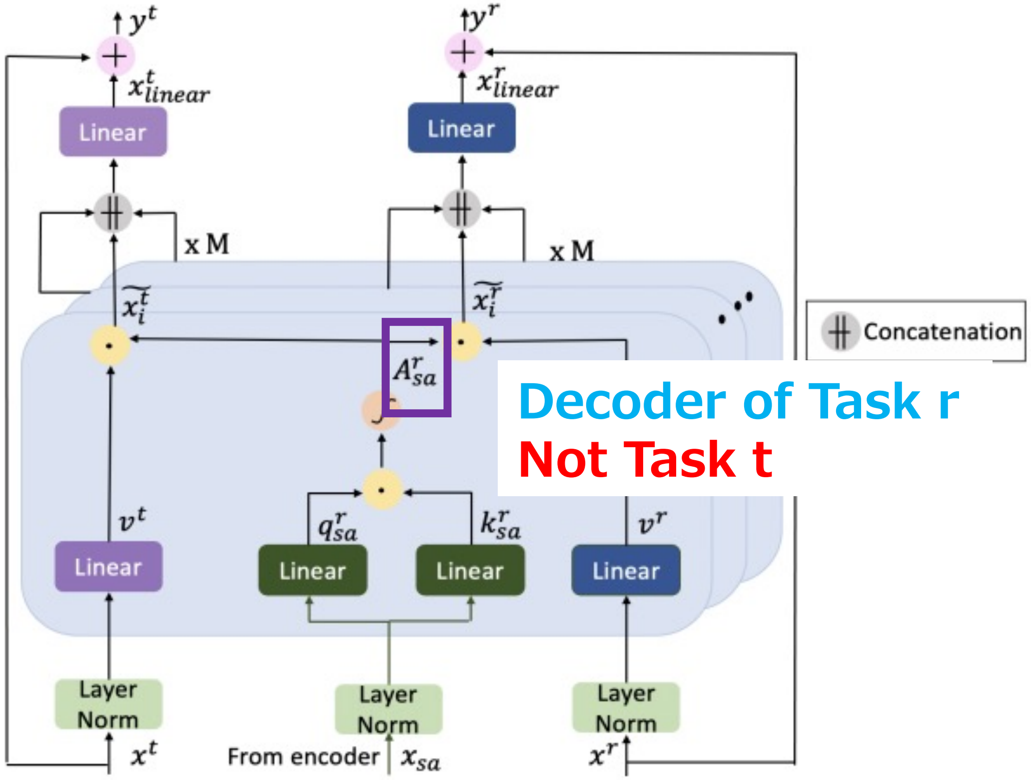
Shared Attention

$$A_{sa}^{r} = softmax\left(\frac{q_{sa}^{r} \cdot k_{sa}^{r\,T}}{\sqrt{C_{qkv}^{r}}} + B^{r}\right)$$



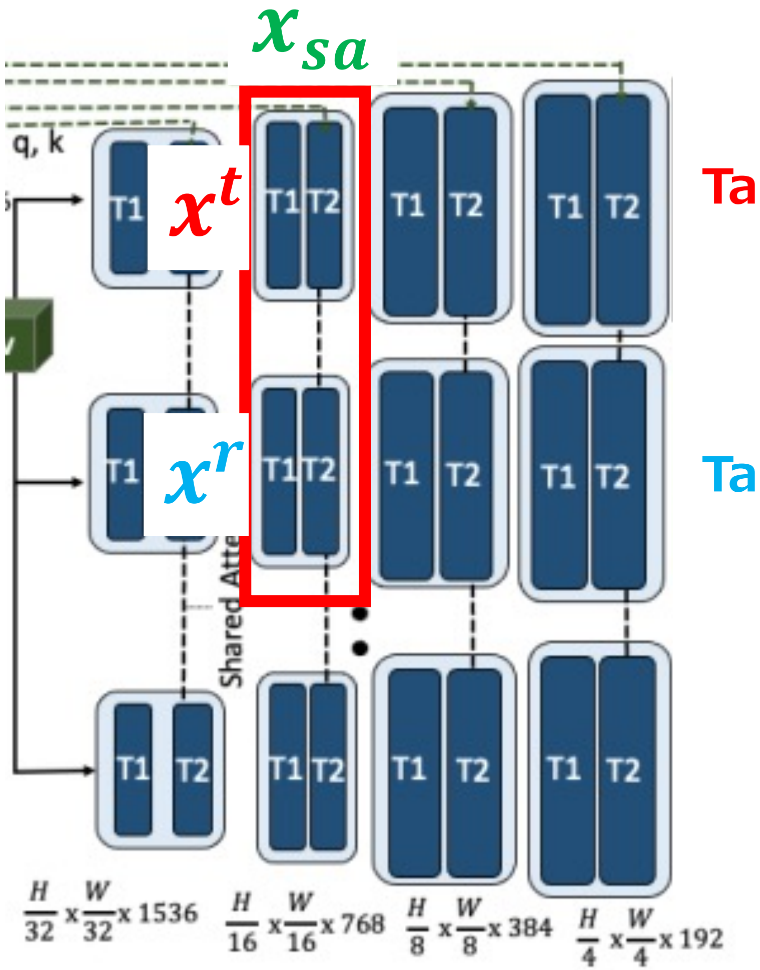
Task t: segmentation
x^t

Task r: Depth Estimation
x^r



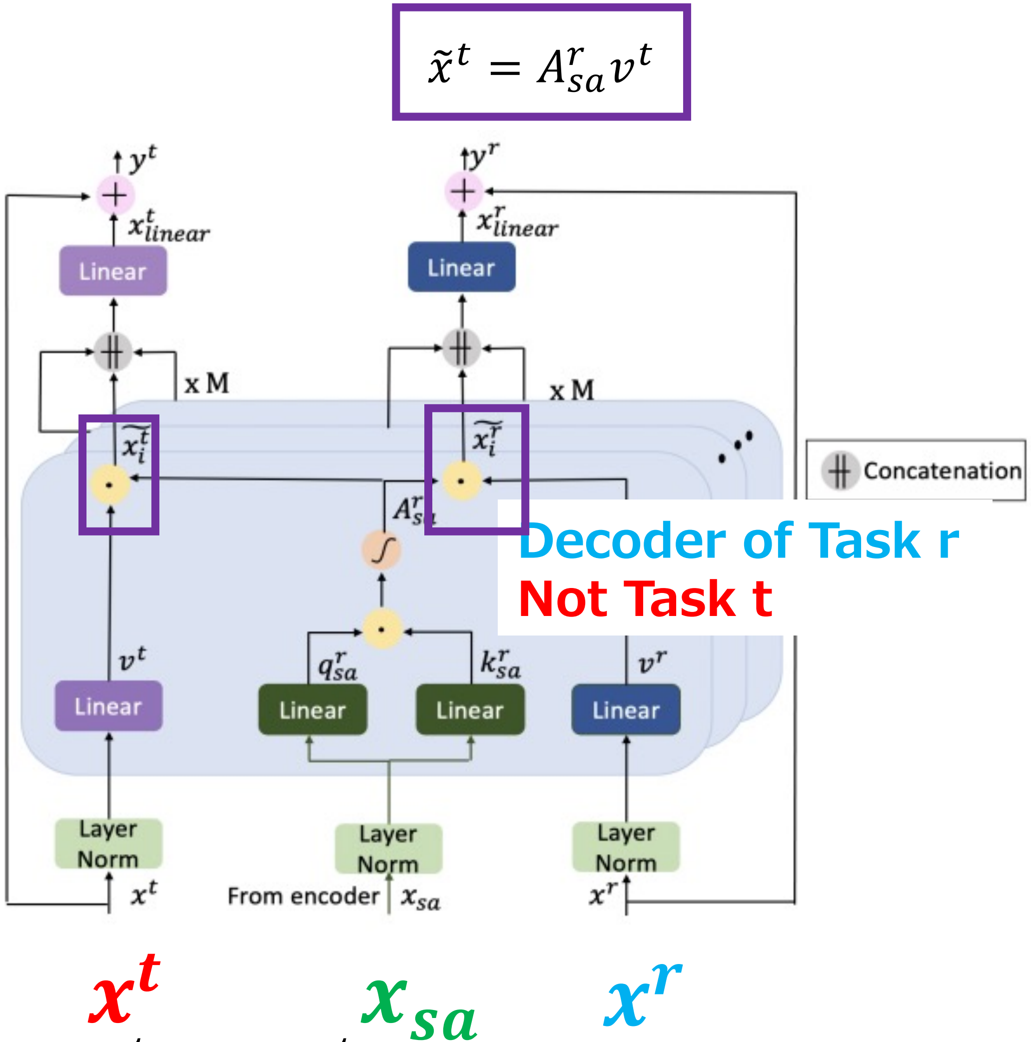
x^t x_{sa} x^r

Shared Attention



Task t: segmentation
x^t

Task r: Depth Estimation
x^r



$$MHA^t(\tilde{x}^t, W) = Concat(head_1^t, \dots, head_M^t)W,$$
$$x^t_{linear} = MHA^t(.,.),$$
$$y^t = x^t + x^t_{linear}$$

Task Heads and Loss

✓ The output from the Transformer decoder is input to task-specific heads, with each class head containing one linear layer that outputs a map of HxWx1.

✓ Losses also appear to be set up as task-specific losses to jointly train a single network.

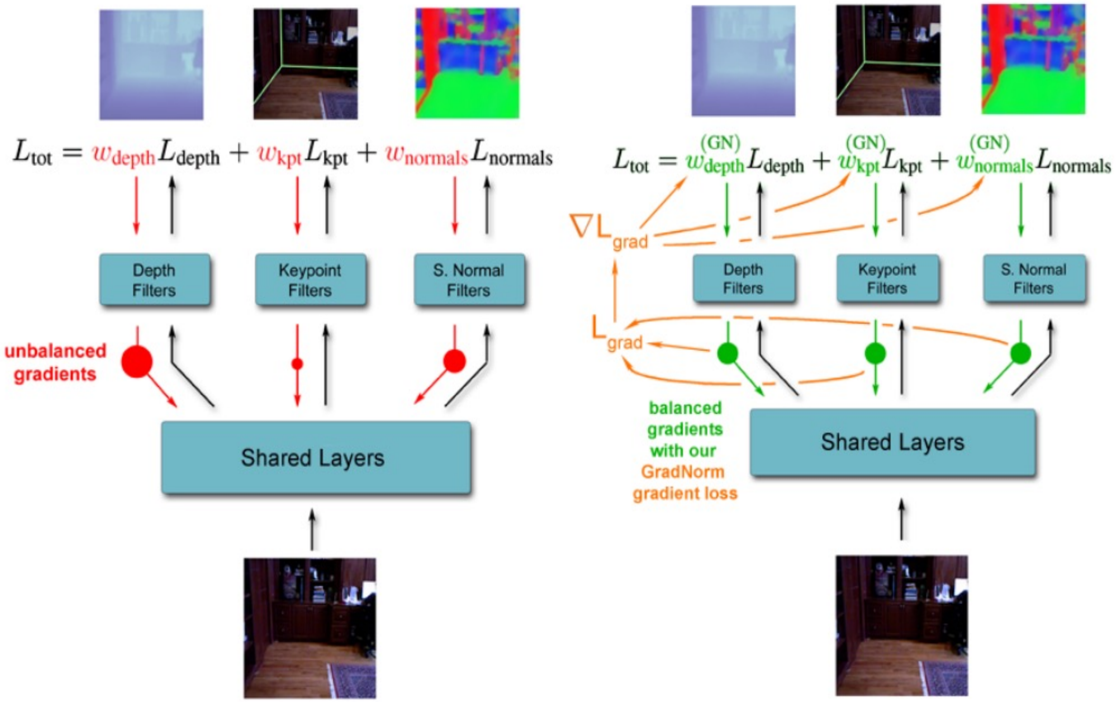
e.g.

Segmentation : cross entropy,
Depth estimation : rotation loss,
Others : L1 loss (MAE).

✓ The entire model is computed with all weighted losses using the GradNorm method.

Loss

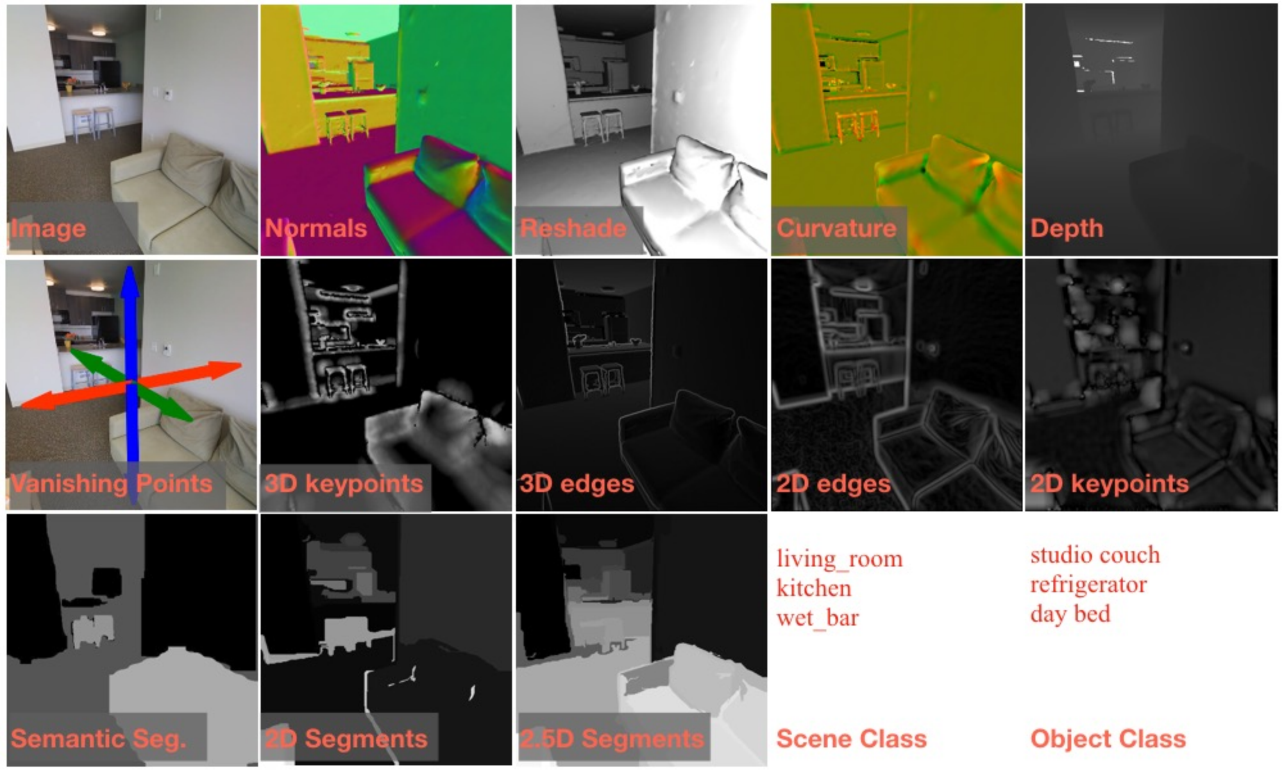
- GradNorm :
  - **GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks,**
  - ICLR2018,
  - Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, Andrew Rabinovich



✓ Weighted Loss
✓ Balanced Gradients

Experiment Details : Training Datasets

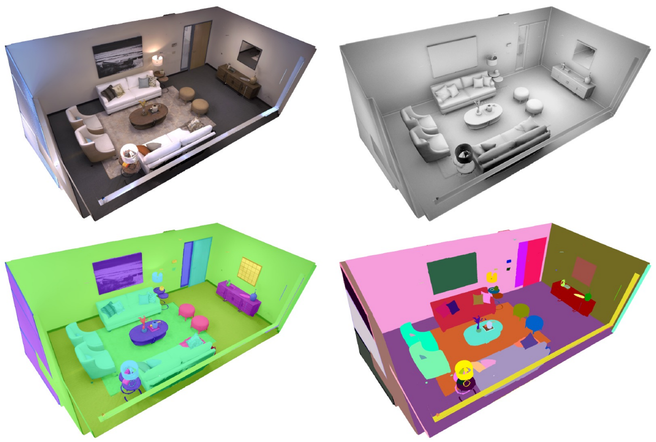
- Taskonomy [URL : https://github.com/StanfordVL/taskonomy/tree/master/data]



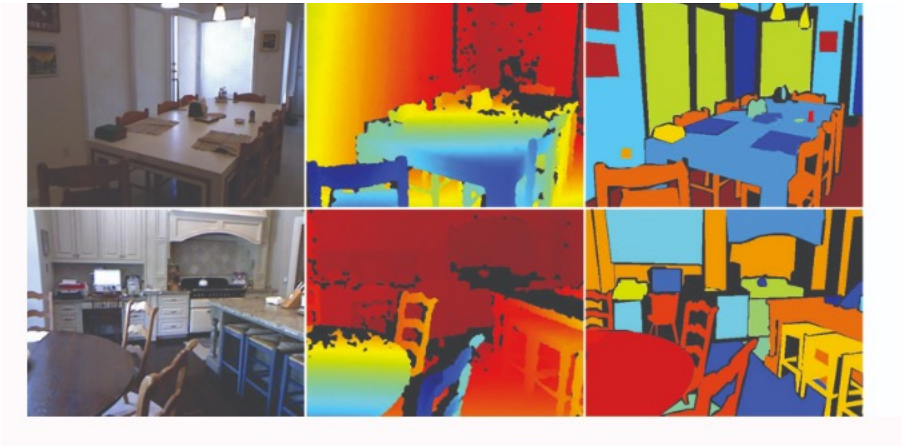
Data : over 4.5 million images from
over 500 buildings
Annotation : 15 tasks/image

Experiment Details : Test Datasets

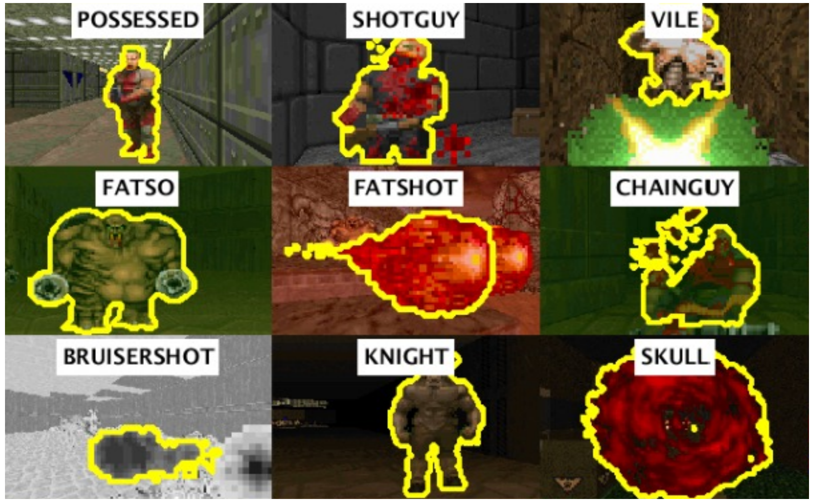
- Taskonomy
- Replica : https://github.com/facebookresearch/Replica-Dataset
  High resolution 1227 images and Ground Truth
  This dataset can evaluate more detail
- NYU : https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html
  1449 images of 464 different rooms
- CocoDoom : https://www.robots.ox.ac.uk/~vgg/research/researchdoom/cocodoom/
  synthetic images from the Doom video games



Replica



NYU



Cocodoom

Experiment Details : Training

- Train 6 tasks at the same time
  - Framework : PyTorch
  - Device : Nvidia V100-SXM2-32GBGPU
  - Batch size : 32
  - Lr : 5e-5
  - Scheduler : warmup cosine
  - Optimizer : weighted adam
  - Loss : CrossEntropy(S), RotationLoss(D), MAE(N,K,E,R)

S: Segmentation,
D: Depth Estimation,
N: Surface Normals,
K: 2D keypoints,
E: 2D texture edges,
R: Reshading

Results : Relative Performance

	Relative Performance On					
	$\mathcal{S}$	$\mathcal{D}$	$\mathcal{N}$	$\mathcal{K}$	$\mathcal{E}$	$\mathcal{R}$
$\mathcal{S}$	-	+3.83%	-1.42%	-1.33%	+33.9%	-0.80%
$\mathcal{D}$	+4.83%	-	+2.77%	-1.92%	+35.2%	+3.93%
$\mathcal{N}$	+11.3%	+8.35%	-	+91.2%	+77.1%	+9.09%
$\mathcal{K}$	+5.11%	+0.57%	-6.88%	-	+70.1%	+0.21%
$\mathcal{E}$	+6.09%	+4.33%	-0.73%	+4.75%	-	+5.11%
$\mathcal{R}$	+8.61%	+4.45%	+5.91%	+1.95%	+33.9%	-

MulT(multi-task) vs Swin(single)

Row : Another task used for training
Columns : Tasks to be tested

S: Segmentation,
D: Depth Estimation,
N: Surface Normals,
K: 2D keypoints,
E: 2D texture edges,
R: Reshading

Quantitative Results

	Relative Performance On					
	$\mathcal{S}$	$\mathcal{D}$	$\mathcal{N}$	$\mathcal{K}$	$\mathcal{E}$	$\mathcal{R}$
$\mathcal{S}$	-	+3.83%	-1.42%	-1.33%	+33.9%	-0.80%
$\mathcal{D}$	+4.83%	-	+2.77%	-1.92%	+35.2%	+3.93%
$\mathcal{N}$	+11.3%	+8.35%	-	+91.2%	+77.1%	+9.09%
$\mathcal{K}$	+5.11%	+0.57%	-6.88%	-	+70.1%	+0.21%
$\mathcal{E}$	+6.09%	+4.33%	-0.73%	+4.75%	-	+5.11%
$\mathcal{R}$	+8.61%	+4.45%	+5.91%	+1.95%	+33.9%	-

MulT(multi-task) vs Swin(single)

Row : Another task used for training
Columns : Tasks to be tested

S: Segmentation,
D: Depth Estimation,
N: Surface Normals,
K: 2D keypoints,
E: 2D texture edges,
R: Reshading

Quantitative Results

MulT(multi-task) vs Swin(single)

Row : Another task used for training
Columns : Tasks to be tested

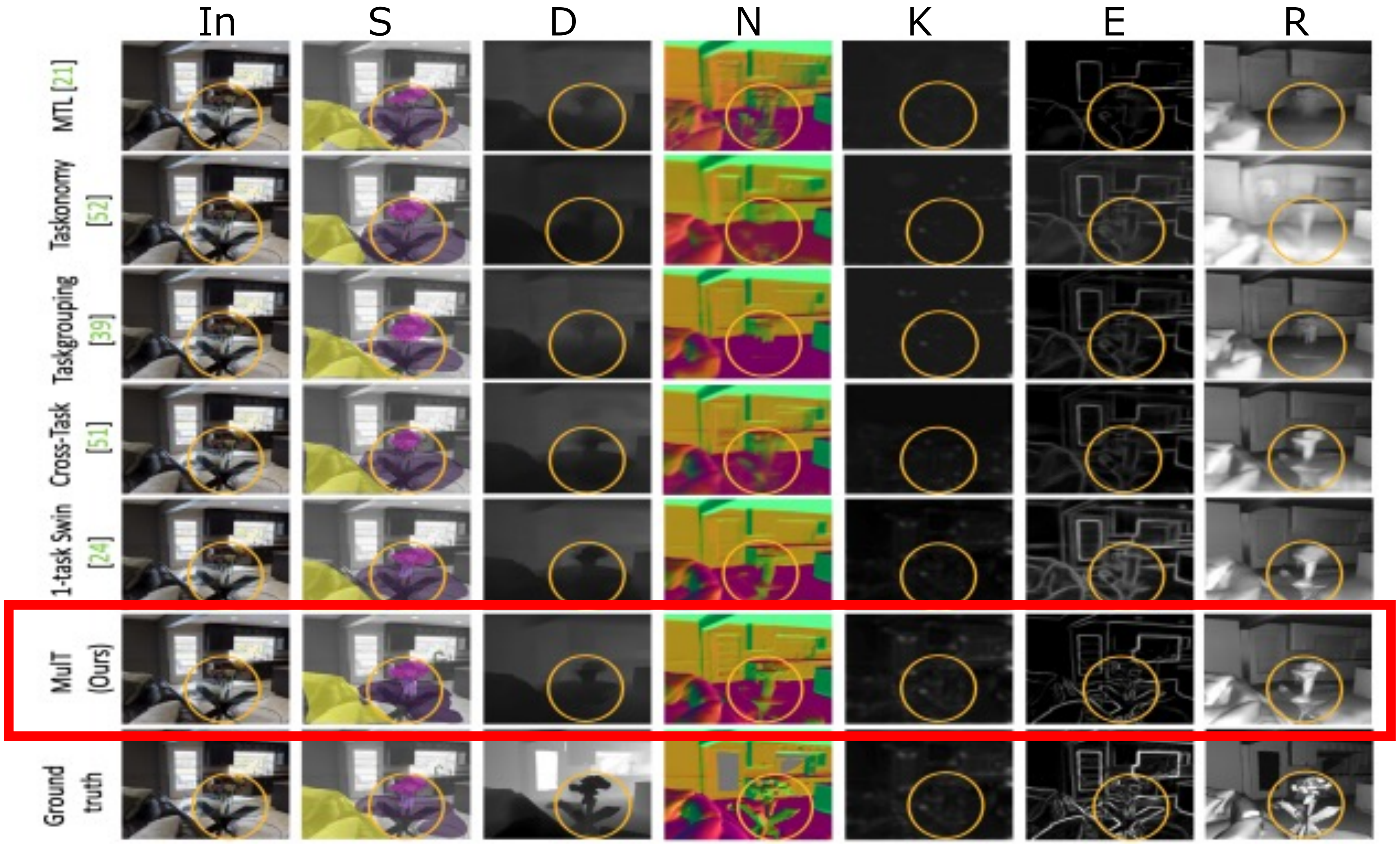
	Relative Performance On					
	$\mathcal{S}$	$\mathcal{D}$	$\mathcal{N}$	$\mathcal{K}$	$\mathcal{E}$	$\mathcal{R}$
$\mathcal{S}$	-	+3.83%	-1.42%	-1.33%	+33.9%	-0.80%
$\mathcal{D}$	+4.83%	-	+2.77%	-1.92%	+35.2%	+3.93%
$\mathcal{N}$	+11.3%	+8.35%	-	+91.2%	+77.1%	+9.09%
$\mathcal{K}$	+5.11%	+0.57%	-6.88%	-	+70.1%	+0.21%
$\mathcal{E}$	+6.09%	+4.33%	-0.73%	+4.75%	-	+5.11%
$\mathcal{R}$	+8.61%	+4.45%	+5.91%	+1.95%	+33.9%	-

		Relative Performance On				
		AutoEnc	Normals	Occ Edges	Reshading	Curvature
Trained With	AutoEnc	–	-3.23%	-2.66%	0.10%	-1.39%
	Normals	19.31%	–	3.16%	4.60%	1.95%
	Occ Edges	35.83%	-0.25%	–	1.15%	0.84%
	Reshading	-24.46%	3.71%	3.16%	–	1.88%
	Princ Curv	10.69%	2.61%	2.46%	3.15%	–
		10.34%	0.71%	1.53%	2.25%	0.82%

S: Segmentation,
D: Depth Estimation,
N: Surface Normals,
K: 2D keypoints,
E: 2D texture edges,
R: Reshading

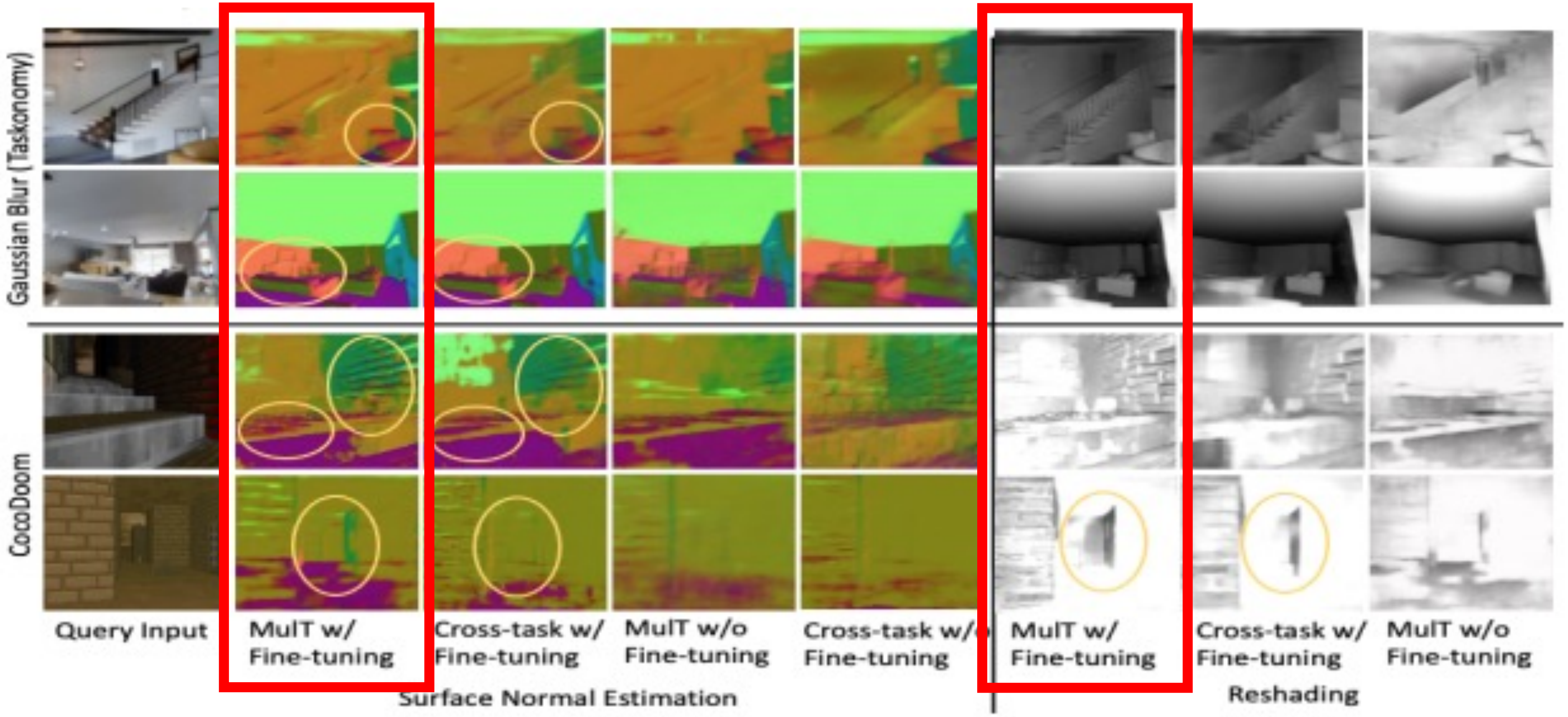
		Relative Performance On					
		SemSeg	Depth	Normals	Keypoints	Edges	Average
Trained With	SemSeg	–	1.91%	-6.00%	-9.91%	-21.93%	-8.98%
	Depth	-12.63%	–	2.95%	1.44%	-9.70%	-4.48%
	Normals	8.32%	15.38%	–	-1.35%	52.08%	18.61%
	Keypoints	-5.84%	-7.21%	-2.26%	–	55.63%	10.08%
	Edges	-5.62%	6.02%	-4.16%	-5.02%	–	-2.20%
		-3.95%	4.03%	-2.37%	-3.71%	19.02%	2.6%

Qualitative Results



In: Input image,
S: Segmentation,
D: Depth Estimation,
N: Surface Normals,
K: 2D keypoints,
E: 2D texture edges,
R: Reshading

Generalization To New Domains



Conclution

- This study proposed a single encoder-decoder model with Transformer tuned for multiple tasks.
- This model is able to solve six tasks simultaneously, and by sharing parameters, it demonstrated superior results for each task in a more compact manner.
- Furthermore, MulT outperforms CNN-based multitasking models and performs well not only on the original domain, but also on unknown domains.

Future Work

- Data dependency
  - Especially MulT is data intensive architecture
  - Require Large amount of data
- Paired data
  - Generalize for other tasks.
  - e.g. Non-surpervised learning
- Extend the combination of local and global attention to a more flexible mechanism.

For My Research

- I am about to work on a multi-task model that image synthesis and regression/classification.
- I chose this paper because I think it is meaningful to be able to do multiple tasks endo to end with a single model.

- Unfortunately, the code has not been updated, but I think that if I could replace the head part at the end for my task, I would be able to get good results with multitasking.