Supplementary Material

1. NearFarMix Augmentation

Figure 1 elegantly illustrates the operation of the proposed NearFarMix augmentation, delineating the four regions—near, pre-far, overlap, and exclusive—that contribute to the final augmented output. Despite the amalgamation of four regions, the pre-far, overlap, and exclusive regions primarily generate far regions, with overlap regions consequently subtracted. Importantly, the augmentation is designed for batch-wise application to enhance execution speed. The step-by-step implementation is comprehensively depicted in Algorithm 3.

Additionally, Fig. 3 and Fig. 2 provide additional examples between the proposed NearFarMix and DepthMix [2].

Algorithm 1 Batchwise Window Partition

$x \leftarrow Input$ ▷ Features
$h_w, w_w \leftarrow window\_size$ ▷ Size of each window

 # Partition features into local dense windows
$B, H, W, C = \text{shape}(x)$
$x = \text{reshape}(x, shape = [B, \frac{H}{h_w}, h_w, \frac{W}{w_w}, w_w, C])$
$x = \text{transpose}(x, permute\_axis = [0, 1, 3, 2, 4, 5])$
$x = \text{reshape}(x, shape = [B \times \frac{H \times W}{h_w \times w_w}, h_w, w_w, C])$

Algorithm 2 Batchwise Grid Partition

$x \leftarrow Input$ ▷ Features
$h_g, w_g \leftarrow grid\_size$ ▷ Size of each grid

 # Partition features into global sparse grids
$B, H, W, C = \text{shape}(x)$
$x = \text{reshape}(x, shape = [B, h_g, \frac{H}{h_g}, w_g, \frac{W}{w_g}, C])$
$x = \text{transpose}(x, permute\_axis = [0, 1, 3, 2, 4, 5])$
$x = \text{reshape}(x, shape = [B, h_g \times w_g, \frac{H \times W}{h_g \times w_g}, C])$
$x = \text{transpose}(x, permute\_axis = [0, 2, 1, 3])$
$x = \text{reshape}(x, shape = [B \times \frac{H \times W}{h_g \times w_g}, h_g, w_g, C])$

Algorithm 3 Batchwise NearFarMix Augmentation

$I_1 \leftarrow Images$ ▷ Input images
$D_1 \leftarrow Depths$ ▷ Input depths
$S_1 \leftarrow Semantics$ ▷ Input semantics
$\mathcal{U} \leftarrow random\_uniform$ ▷ Uniform distribution
$D_{min} \leftarrow 20(\text{KITTI})$ or $1.5(\text{NYUv2})$ ▷ Min depth
$D_{max} \leftarrow 60(\text{KITTI})$ or $6.5(\text{NYUv2})$ ▷ Max depth

 # Roll batch for fast shuffling
$I_2 = \text{roll}(I_1, shift = 1, axis = 0)$ ▷ Roll Images
$D_2 = \text{roll}(D_1, shift = 1, axis = 0)$ ▷ Roll Depths
$S_2 = \text{roll}(S_1, shift = 1, axis = 0)$ ▷ Roll Semantics

 # Depth threshold range for batch
$d_{min} = \max(\min(D_1, axis = (1, 2, 3)))$ ▷ Min depth
$d_{max} = \min(\max(D_1, axis = (1, 2, 3)))$ ▷ Max depth

 # Threshold for Near-Far region
$B, H, W, C = shape(I_1)$
$thr_{min} = max(D_{min}, d_{min})$ ▷ Clip min depth
$thr_{max} = min(D_{max}, d_{max})$ ▷ Clip max depth
$thrs = \mathcal{U}(shape = [B, 1, 1, 1],$ ▷ Random thresholds
$min = thr_{min},$
$max = thr_{max})$

 # Compute binary masks of regions for blending
$M_1 = D_1 <= thrs$ ▷ Broadcasted Near region mask
$M_2 = D_2 > thrs$ ▷ Broadcasted pre-Far region mask
$M_o = M_1 \odot M_2$ ▷ Overlap region mask
$M_e = (1 - M_1) \odot (1 - M_2)$ ▷ Exclusive region mask

 # Perform blending of regions
$I' = (I_1 \odot M_1)_{near} + ((I_2 \odot M_2) + (I_2 \odot M_e) - (I_2 \odot M_o))_{far}$ ▷ Augmented image
$D' = (D_1 \odot M_1)_{near} + ((D_2 \odot M_2) + (D_2 \odot M_e) - (D_2 \odot M_o))_{far}$ ▷ Augmented depth
$S' = (S_1 \odot M_1)_{near} + ((S_2 \odot M_2) + (S_2 \odot M_e) - (S_2 \odot M_o))_{far}$ ▷ Augmented semantics

2. Symbiotic Transformer

Transformers: Equation (1) presents the detailed mathematical expression for Symbiotic Transformer, which sym-

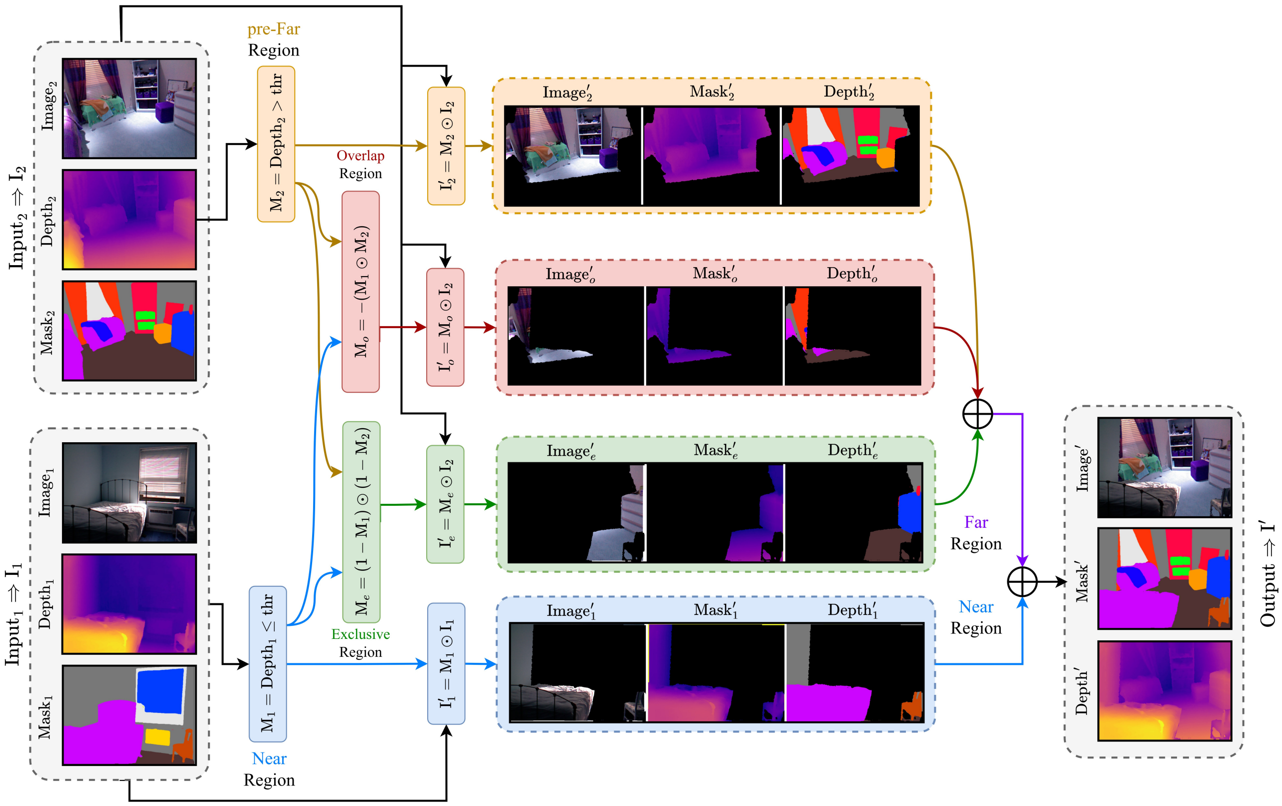

Figure 1. Proposed NearFarMix augmentation. The depth map undergoes thresholding to generate **Near** and **pre-Far** regions. Then, these regions are manipulated to produce **Overlap** and **Exclusive** regions. The **Far** region is then generated by combining the **pre-Far**, **Overlap**, and **Exclusive** regions, with the **Exclusive** region being subtracted and the remaining regions added. Finally, the **Near** and **Far** regions are combined to generate the augmented image.

biotically enhances both depth and semantics via local-global cross-attention. In the equation, $\mathbf{F_x^Q}$ represents query features of $\mathbf{x}$, $\mathbf{F_y^{KV}}$ denotes key-value features of $\mathbf{y}$, and $\mathbf{F_y}$ signifies the output features contextualized by $\mathbf{x}$. Specifically, for SGT, $\mathbf{x} = \mathbf{s}$ and $\mathbf{y} = \mathbf{d}$, while for DGT, $\mathbf{x} = \mathbf{d}$ and $\mathbf{y} = \mathbf{s}$. Moreover, $\text{DGT} = \text{LG-CAT}_{DG}$ and $\text{SGT} = \text{LG-CAT}_{SG}$ correspond to depth and semantics-guided local-global cross-attention transformers.

Cross Attentions: Under the hood, DGT and SGT employ semantics-guided cross attention (SG-CA) and depth-guided cross attention (DG-CA), respectively to contextualize features. SG-CA and DG-CA is mathematically elaborated in Eq. (2) and Eq. (3). In these equations, $\mathbf{W}$ represents the weight of the FFN layer, $\mathbf{Q_x}$ represents query features, $\mathbf{K_y}$ and $\mathbf{V_y}$ represent key and value features, $\mathcal{S}mx$ denotes softmax, and $\mathbf{d}$ is the query/key dimension. $\mathbf{B}$ represents relative positional bias, sampled similar to [1]. The Local-Global Cross-Attention Transformer (LG-CAT), employed by both SGT and DGT, can be implemented using Algorithm 4.

Partition Operation: Further, the algorithms for $\mathrm{WindowPartition}$ and $\mathrm{GridPartition}$ operations which are used in Block and Grid attention also slightly different from the methods of Max-ViT [4], are detailed in Algorithm 1 and Algorithm 2, respectively. It is noteworthy that Max-ViT implemented these operations using einops [3].

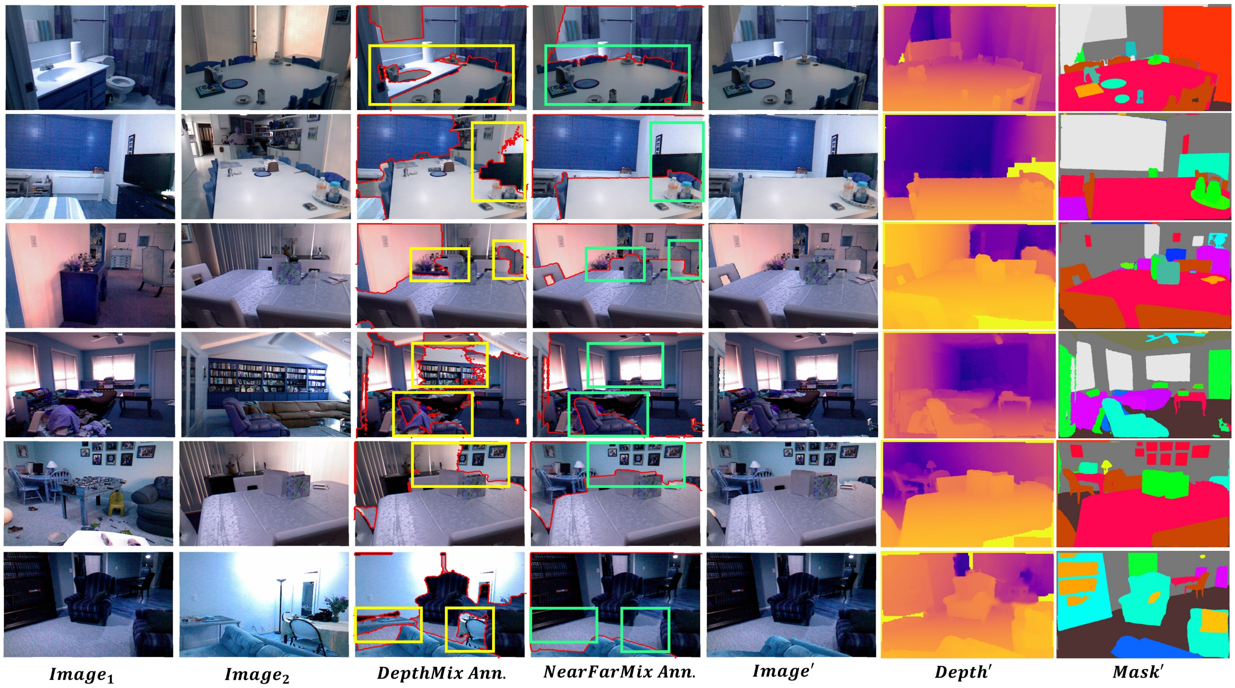

Figure 2. Additional qualitative comparisons between proposed NearFarMix and DepthMix augmentation on NYUv2 dataset.

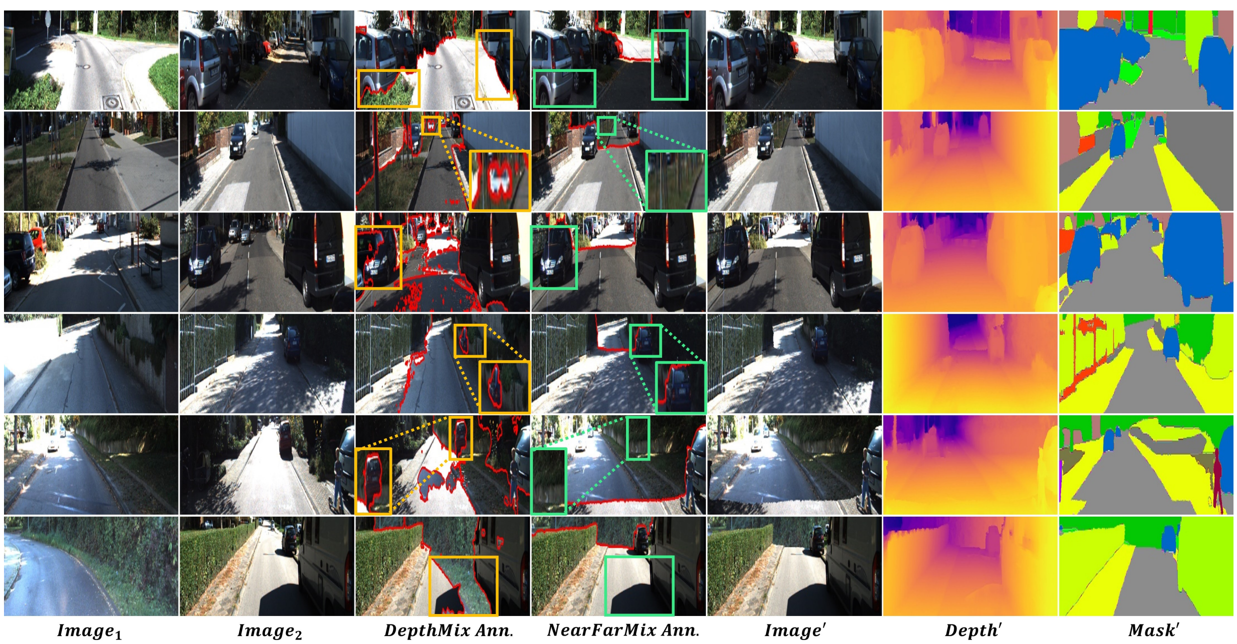

Figure 3. Additional qualitative comparisons between proposed NearFarMix and DepthMix augmentation on KITTI dataset.

$$\text{Sym-T}(\mathbf{F}'_{\mathbf{d}}, \mathbf{F}'_{\mathbf{s}}) = \begin{bmatrix} \mathbf{F}_{\mathbf{d}}^{\mathbf{Q}} = \mathbf{F}_{\mathbf{d}}^{\mathbf{KV}} = \mathbf{F}'_{\mathbf{d}} \\ \mathbf{F}_{\mathbf{s}}^{\mathbf{Q}} = \mathbf{F}_{\mathbf{s}}^{\mathbf{KV}} = \mathbf{F}'_{\mathbf{s}} \\ \mathbf{F}_{\mathbf{s}} \quad = \text{LG-CAT}_{DG}(\mathbf{F}_{\mathbf{d}}^{\mathbf{Q}}, \mathbf{F}_{\mathbf{s}}^{\mathbf{KV}}) \\ \mathbf{F}_{\mathbf{d}} \quad = \text{LG-CAT}_{SG}(\mathbf{F}_{\mathbf{s}}^{\mathbf{Q}}, \mathbf{F}_{\mathbf{d}}^{\mathbf{KV}}) \end{bmatrix} \tag{1}$$

$$\begin{aligned} \text{SG-CA} &= \text{CA}\left(\mathbf{Q}_{\mathbf{s}}, \mathbf{K}_{\mathbf{d}}, \mathbf{V}_{\mathbf{d}}\right) \\ &= \mathcal{S}mx\left(\frac{\mathbf{Q}_{\mathbf{s}}\mathbf{K}_{\mathbf{d}}{}^{T}}{\sqrt{\mathbf{d}}} + \mathbf{B}\right)\mathbf{V}_{\mathbf{d}} \\ &= \mathcal{S}mx\left(\frac{(\mathbf{F}_{\mathbf{s}}^{\mathbf{Q}} \cdot \mathbf{W}_{\mathbf{Q}}^{\mathbf{s}})(\mathbf{F}_{\mathbf{d}}^{\mathbf{KV}} \cdot \mathbf{W}_{\mathbf{K}}^{\mathbf{d}})^{T}}{\sqrt{\mathbf{d}}} + \mathbf{B}\right)(\mathbf{F}_{\mathbf{d}}^{\mathbf{KV}} \cdot \mathbf{W}_{\mathbf{V}}^{\mathbf{d}}) \end{aligned} \tag{2}$$

Algorithm 4 Local-Global Cross-Attention Transformer (LG-CAT)

$F_x^Q, F_y^{KV} \leftarrow inputs$ ▷ Input features
$x \leftarrow F_x^Q$ ▷ Query features of depth/semantics
$y \leftarrow F_y^{KV}$ ▷ Key-value features of semantics/depth
$i \leftarrow 0$ ▷ Initialize counter

while $i \neq 2$ **do**
  # Block Cross Attention
  $x_1 = \text{layer\_norm}(x)$
  $y_1 = \text{layer\_norm}(y)$
  $x_{1,q} = \text{FFN}(\text{window\_partition}(x_1))$ ▷ Query gen.
  $y_{1,k}, y_{1,v} = \text{FFN}(\text{window\_partition}(y_1))$ ▷ KeyValue
  $y_2 = \text{CA}(x_{1,q}, y_{1,k}, y_{1,v})$ ▷ Apply cross-attention
  $y_2 = y_1 + \text{window\_reverse}(\text{FFN}(y_2))$ ▷ Residual

  # Grid Cross Attention
  $y_2 = \text{layer\_norm}(y_2)$
  $x_{2,q} = \text{FFN}(\text{grid\_partition}(x_1))$ ▷ Query
  $y_{2,k}, y_{2,v} = \text{FFN}(\text{grid\_partition}(y_2))$ ▷ KeyValue
  $y_3 = \text{CA}(x_{2,q}, y_{2,k}, y_{2,v})$ ▷ Apply cross-attention
  $y_3 = y_2 + \text{grid\_reverse}(\text{FFN}(y_3))$ ▷ Residual
  $y = y_3$ ▷ Reset variable for loop
  $i = i + 1$ ▷ Increment counter
end while

 # FusedMBConv - Channel Attention
$\hat{y} = \text{DWConv3x3}(y)$ ▷ Depthwise convolution
$\hat{y} = \text{GELU}(\hat{y})$ ▷ Apply activation
$\hat{y} = \text{SE}(\hat{y})$ ▷ Squeeze-Excitation
$\hat{y} = \text{Conv1x1}(\hat{y})$ ▷ Convolution
$\hat{y} = y + \hat{y}$ ▷ Residual

$output \leftarrow \hat{y}$

$$\begin{aligned}
\text{DG-CA} &= \text{CA}\left(\mathbf{Q_d}, \mathbf{K_s}, \mathbf{V_s}\right) \\
&= \mathcal{S}mx\left(\frac{\mathbf{Q_d}\mathbf{K_s}^T}{\sqrt{\mathbf{d}}} + \mathbf{B}\right)\mathbf{V_s} \\
&= \mathcal{S}mx\left(\frac{(\mathbf{F_d^Q}\cdot\mathbf{W_Q^d})(\mathbf{F_s^{KV}}\cdot\mathbf{W_K^s})^T}{\sqrt{\mathbf{d}}} + \mathbf{B}\right)(\mathbf{F_s^{KV}}\cdot\mathbf{W_V^s})
\end{aligned} \tag{3}$$

3. Architecture Details

The architectural specifications, encompassing input, output, layer name, and layer details, are succinctly laid out in Table 1. Here, E and D represent the input/output of the encoder/decoder, while ST and N denote to the Symbiotic Transformer and the Neck.

References

[1] Ali Hatamizadeh, Hongxu Yin, Jan Kautz, and Pavlo Molchanov. Global context vision transformers. *arXiv preprint arXiv:2206.09959*, 2022. 2

[2] Lukas Hoyer, Dengxin Dai, Yuhua Chen, Adrian Koring, Suman Saha, and Luc Van Gool. Three ways to improve semantic segmentation with self-supervised depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11130–11140, 2021. 1

[3] Alex Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2021. 2

[4] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. *arXiv preprint arXiv:2204.01697*, 2022. 2

Table 1. Architectural Specifications of proposed method where h, w signify attention heads and window size; $Conv$ denotes 2D convolution with k, s, c as kernel size, stride size, and output channels; act and $norm$ represent activation and normalization types; sc indicates upscale size.

Input Size: $H \times W \times 3$				
Layer Name	Input	Output	Output Size	Architecture
Stem	$Image$	E_0	$\frac{H}{4} \times \frac{W}{2} \times 128$	Conv(c=128, k=3, s=2) Conv(c=128, k=3, s=1)
Encoder Stage 1	E_0	E_1	$\frac{H}{8} \times \frac{W}{4} \times 128$	[Max-ViT-Block(h=4, w=7)] $\times$ 2
Encoder Stage 2	E_1	E_2	$\frac{H}{16} \times \frac{W}{8} \times 256$	[Max-ViT-Block(h=8, w=7)] $\times$ 6
Encoder Stage 3	E_2	E_3	$\frac{H}{32} \times \frac{W}{16} \times 512$	[Max-ViT-Block(h=16, w=7)] $\times$ 14
Encoder Stage 4	E_3	E_4/ D_4	$\frac{H}{32} \times \frac{W}{32} \times 1024$	[Max-ViT-Block(h=16, w=7)] $\times$ 2
Decoder Stage 3	(D_4, E_3)	D_3	$\frac{H}{32} \times \frac{W}{32} \times 512$	Upsample(sc=2) Concat([E_3, D_4], axis=-1) Conv(c=512, k=3, s=1, norm='layer', act='gelu')
Decoder Stage 3	(D_3, E_2)	D_2	$\frac{H}{16} \times \frac{W}{16} \times 256$	Upsample(sc=2) Concat([E_2, D_3], axis=-1) Conv(c=256, k=3, s=1, norm='layer', act='gelu')
Decoder Stage 1	(D_2, E_1)	D_1	$\frac{H}{4} \times \frac{W}{4} \times 128$	Upsample(sc=2) Concat([E_1, D_2], axis=-1) Conv(c=128, k=3, s=1, norm='layer', act='gelu')
Neck	D_1	(N_d, N_s)	$(\frac{H}{4} \times \frac{W}{4} \times 150,$ $\frac{H}{4} \times \frac{W}{4} \times 150)$	([Conv(c=150, k=3, s=1, norm='layer', act='gelu')] $\times$ 2, [Conv(c=150, k=3, s=1, norm='layer', act='gelu')] $\times$ 2)
Symbiotic Transformer	(N_d, N_s)	(ST_d, ST_s)	$(\frac{H}{4} \times \frac{W}{4} \times 150,$ $\frac{H}{4} \times \frac{W}{4} \times 150)$	[Block-Cross-Attention(h=4, w=7) Grid-Cross-Attention(h=4, w=7)] $\times$ 2 [FusedMBConv] $\times$ 1
Head	(ST_d, ST_s)	$(Depth, Semantics)$	$(H \times W \times 1,$ $H \times W \times 150)$	([Conv(k=3, s=1, act='sigmoid') Upsample(sc=4)] $\times$ 2, [Conv(k=3, s=1, act='softmax') Upsample(sc=4)] $\times$ 2)
($Depth$: $H \times W \times 1$, $Semantics$: $H \times W \times 150$)				