Hyperdimensional Uncertainty Quantification for Multimodal Uncertainty Fusion in Autonomous Vehicles Perception

Supplementary Material

6. Similarity to Uncertainty/Expressivity to Uncertainty Quantifiability

The projection function ϕ is an encoding from $\mathbb{R}^n \rightarrow \mathbb{R}^d$ that can be mathematically expressed as:

$$\phi(z) = \Phi \cdot z \tag{10}$$

where $\Phi \in \mathbb{R}^{d \times n}$ is the projection matrix and the output hypervector space **H** is an **inner-product space**.

According to [57], the initialization of Φ can limit the learnability of a VSA system. They proved this by defining VSA systems by their **expressivity** as follows:

Definition 1 *A VSA system can express a similarity matrix $\mathcal{M} \in \mathbb{R}^{N \times N}$ if for any $\epsilon > 0$, there exists a $d \in \mathbb{N}$ and d-dimensional hypervectors $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_N$ such that $|\mathcal{M}_{i,j} - \delta(\mathcal{H}_i, \mathcal{H}_j)| \leq \epsilon$*

Since $\mathcal{M}$ represents the knowledge of all sample relations in the **H** space, the expressivity depends on whether ϕ can capture the similarities of $\mathcal{M}$ accurately. Given the knowledge of $\mathcal{M}$, **we derive an expression for the *uncertainty quantifiability* of a VSA system as follows:**

Corollary 1 *A VSA can express an uncertainty similarity matrix $\mathcal{U} \in \mathbb{R}^{L \times N}$ if for any $\eta > 0$, there exists a $d \in \mathbb{N}$ and d-dimensional hypervectors $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_N$ such that $|\mathcal{U}_{i,j} - \delta(\mathcal{H}_i, \mathcal{H}_j)| \leq \eta$.*

In this Corollary, the rows of the uncertainty similarity matrix $\mathcal{U}$ are the prototypes $\mathcal{H}^l_m$ (Equation 1) and the columns are the similarities of each hypervectors $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_n$ to each $\mathcal{H}^l_m$ (Equation 2).

Expressivity thus directly impacts a VSA's ability to accurately quantify uncertainty. [3] [57] showed that classical initializations using the kernel trick [43] has limited expressiveness. Instead, Random Fourier Features (RFF) can, in expectation, exactly achieve M or some approximation of M.

However, the above assumes that the **orthogonality** between hypervectors is maintained by ϕ. When assessing whether some $\mathcal{H}_z \in \mathcal{H}^l_m$ and the encodings are perfectly orthogonal, we get the expression $|\delta(\mathcal{H}_z, \mathcal{H}^l_m)| = I\mathbb{1}(\mathcal{H}_z \in \mathcal{H}^l_m)$ where $\mathbb{1}$ is an indicator that evaluates to one if true and zero otherwise, $I = \min_{z \in \mathcal{Z}} ||\phi(z)||^2$ when δ is the dot-product or $I = 1$ when δ is the cosine similarity.

[3] Obtaining $\mathcal{M}$ involves solving an intractable linear programming problem of size exponential in N, making it unrealistic both from a computation and memory perspective for arbitrarily large datasets.

According to [48], when orthogonality is not maintained, it can cause interference in the hypervector encoding. They characterizes this as the **incoherence** which limits the expressivity of ϕ as follows:

Definition 2 *For $\mu \geq 0$, ϕ is μ-incoherent if for all distinct $z, z' \in \mathcal{Z}$ we have*

$$|\delta(\phi(z), \phi(z'))| \leq \mu I \tag{11}$$

When the encoding $\phi(z)$ and $\phi(z')$ are **not perfectly orthogonal** ($\mu = 0$), the interference causes a Δ "cross-talk" such that $\delta(\mathcal{H}_z, \mathcal{H}^l_m) = I\mathbb{1}(\mathcal{H}_z \in \mathcal{H}^l_m) + \Delta$. **By combining Corollary 1 & Definition 2 we propose a new definition for the *uncertainty quantifiability of a μ-incoherent ϕ* as follows:**

Definition 3 *A μ-incorherent VSA system can express an uncertainty matrix $\mathcal{U} \in \mathbb{R}^{L \times N}$ if for any $\eta, \mu > 0$, there exists a $d \in \mathbb{N}$ and d-dimensional hypervectors $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_N$ s.t. $|\mathcal{U}_{i,j}| \leq \mu I + \eta$*

Thus, to reliably quantify uncertainty, we must ensure the contribution of μ is small. Given that the loss of orthogonality contributes to incoherence, the instinct is to improve ϕ by enforcing orthogonality onto the initialization of the projection matrix Φ.

7. Proof of Equation

Given the following inequalities:

$$|\mathcal{U}_{i,j} - \delta(\mathcal{H}_i, \mathcal{H}_j)| \leq \eta \tag{1}$$
$$|\delta(\mathcal{H}_i, \mathcal{H}_j)| \leq \mu L \tag{2}$$

Step 1. Rewrite (1) & (2) as follows:

$$-\eta \leq \mathcal{U}_{i,j} - \delta(\mathcal{H}_i, \mathcal{H}_j) \leq \eta \tag{3}$$
$$-\mu L \leq \delta(\mathcal{H}_i, \mathcal{H}_j) \leq \mu L \tag{4}$$

Step 2. Add $\delta(\mathcal{H}_i, \mathcal{H}_j)$ to both sides of (1):

$$\delta(\mathcal{H}_i, \mathcal{H}_j) - \eta \leq \mathcal{U}_{i,j} \leq \delta(\mathcal{H}_i, \mathcal{H}_j) + \eta \tag{5}$$

Step 3. Define upper and lower bounds for (4) and (5):
Upper Bounds:

$$\delta(\mathcal{H}_i, \mathcal{H}_j) \leq \mu L \tag{6}$$
$$\mathcal{U}_{i,j} \leq \delta(\mathcal{H}_i, \mathcal{H}_j) + \eta \tag{7}$$

Lower Bounds:

$$-\mu L \leq \delta(\mathcal{H}_i, \mathcal{H}_j) \tag{8}$$
$$\delta(\mathcal{H}_i, \mathcal{H}_j) - \eta \leq \mathcal{U}_{i,j} \tag{9}$$

Step 4. Substitute terms and rewrite bounds:
Upper Bounds:

$$\mathcal{U}_{i,j} \leq \mu L + \eta \tag{10}$$

Lower Bounds:

$$-\mu L - \eta \leq \mathcal{U}_{i,j} \tag{11}$$

Step 5. Rewrite inequality, end of proof:

$$-(\mu L + \eta) \leq \mathcal{U}_{i,j} \leq \mu L + \eta = |\mathcal{U}_{i,j}| \leq \mu L + \eta \tag{12}$$

8. FLOPs Computation Approximation

We approximate the number of Floating-Point Operations for each method by deriving the major contributing operations as follows:

Matrix Multiplication FLOPs Approximation: Projection encoding uses the dot product operation which involves matrix multiplication and addition. We assume an input size of $In = 1 \times C$, a projection matrix size of $Proj = C \times D$, and an expected output size of $Out = B \times D$. For each element $Out_{i,j}$ there are C multiplications. For each element $Out_{i,j}$ there are $C - 1$ additions (because we need to add C products together, which requires $C - 1$ additions). Where B is the batch size (we are assuming B=1 for the sake of simplicity), C is the channel size, and D is the output dimension/hyperdimension. Thus, for the entire output Out with $B \times D$ elements:

$$\begin{aligned}
\text{Total Multiplications} &= B \cdot D \cdot C && (13)\\
\text{Total Additions} &= B \cdot D \cdot (C-1) && (14)\\
\text{Total FLOPs} &= B \cdot D \cdot C && (15)\\
&\quad + B \cdot D \cdot (C-1) \\
&= 2 \cdot B \cdot D \cdot C - B \cdot D && (16)\\
&= B \cdot D(2 \cdot C - 1) && (17)\\
&\approx 2 \cdot B \cdot D \cdot C && (18)
\end{aligned}$$

Einstein Summation FLOPs Approximation: is a general case of matrix multiplication where we used it for our Channel-Projection method. Specifically, Einsum("BC,CD->BCD", A, B), which reduces the computation to an outer product, obtains Out $= B \times C \times D$ with 0 addition FLOPs:

$$\begin{aligned}
\text{Total Multiplications} &= B \cdot D \cdot C && (19)\\
\text{Total Additions} &= 0 && (20)\\
\text{Total FLOPs} &= B \cdot D \cdot C && (21)
\end{aligned}$$

Cosine Similarity FLOPs Approximation: $cosine\_similarity(u, v) = \frac{u \cdot v}{||u|| ||v||}$ involves dot product along with division and two Euclidean norm operations. We assume an input u of size $B \times D$ and the comparison v is of size $1 \times D$. Since $B = 1$, the dot product is between two vectors, thus involving D multiplications and $D - 1$ additions. For Euclidean Norm $||u|| = \sqrt{\sum_{d=1}^{D} u_d^2}$, squaring involves D multiplications, summing involves $D - 1$ additions, and we assume square root is 1 operation, thus the total FLOPs for cosine similarity is:

$$\begin{aligned}
\text{Dot Product FLOPs} &= D + (D-1) && (22)\\
&= 2 \cdot D - 1 && (23)\\
\text{Euclidean Norm FLOPs} &= D + (D-1) + 1 && (24)\\
&= 2 \cdot D && (25)\\
\text{Division + Norm Mult. FLOPs} &= 2 && (26)\\
\text{Total FLOPs} &= (2 \cdot D + 2 \cdot D && (27)\\
&\quad + 2 \cdot D + 2) \\
&= 6 \cdot D + 1 \approx 6 \cdot D && (28)
\end{aligned}$$

Entropy FLOPs Approximation: The deterministic entropy equation $H(P) = -\sum_i P(i) log P(i)$ involves addition from summation, multiplication, and logarithm. We assume that P has dimensions $B \cdot C \cdot H \cdot W$, and thus the number of addition, multiplication, and logarithmic operations depends on the number of elements which is $B \cdot C \cdot H \cdot W$. Therefore, the total number of FLOPs for Entropy is:

$$\begin{aligned}
\text{Total elements} &= B \cdot C \cdot H \cdot W && (29)\\
\text{Mult. + Log FLOPs} &= 1 + 1 = 2 && (30)\\
\text{Sum FLOPs} &= B \cdot C \cdot H \cdot W && (31)\\
\text{Total FLOPs} &= 2 \cdot \text{Tot. elems.} && (32)\\
&\quad + \text{Sum FLOPs} \\
&= 3 \cdot B \cdot C \cdot H \cdot W && (33)
\end{aligned}$$

We note that predictive entropy and mutual information are more complex operations that induce more FLOPs (refer to [49]), for simplicity, we assume the same FLOPs as the deterministic entropy.

From here, we demonstrate the aiMotive FLOPs computations in Table 8 as an example with $B = 1$. The BEVFusion model from aiMotive generates RGB features (BEVDepth) and Lidar+Radar (VoxelNet) features where the total feature dimensions together are $C \times H \times W = 336 \times 64 \times 512$ (RGB = 80 channels and Lidar+Radar = 256).

InfMCD: involves 10 forward passes to generate feature outputs (realistically all previous computation layer FLOPs in the model should be accounted for), followed by predictive entropy, mutual information, and deterministic entropy computations (we assume 3× deterministic entropy FLOPs)

$$\text{Total elements} = 336 \times 64 \times 512 = 11,010,048 \quad (34)$$

$$\text{Total FLOPs} = 10 \text{ outputs} \times 3 \times \text{ Entropy FLOPs} \quad (35)$$

$$= 30 \times 3 \times 11,010,048 \quad (36)$$

$$= 990,904,320 = 990.90 \text{ MFLOPs} \quad (37)$$

InfNoise: involves 10 forward passes to generate Gaussian noise added features (realistically FLOPs for generating Gaussian noise and adding noise to the input of a specified layer along with FLOPs for obtaining the new layer outputs should be accounted for), followed by predictive entropy, mutual information, and deterministic entropy computations (we assume $3\times$ deterministic entropy FLOPs). This results in the same approximated FLOPs as InfMCD.

LDU: involves comparing the cosine_similarity of the features with the learned prototypes along with a Conv2d for uncertainty estimation (ignored due to negligible cost). Since LDU allows arbitrary prototypes, we set it to $L = 4$, the same number as our method based on the number of aiMotive scenarios. Additionally, LDU's prototypes are defined with the shape $1 \times L \times C \times H \times W$.

$$\text{Vector } u \text{ dimension} = 336 \times 64 \times 512 \quad (38)$$

$$= 11,010,048 \quad (39)$$

$$\text{Total FLOPs} = L \times \text{Cos\_Sim FLOPs} \quad (40)$$

$$= 4 \times 6 \times u \quad (41)$$

$$= 24 \times 11,010,048 \quad (42)$$

$$= 264,241,152 \quad (43)$$

$$= 264.24 \text{ MFLOPs} \quad (44)$$

HyperDUM: involves the feature projection and cosine similarity computation, where we set $d = 10k$ as real-valued hyperdimensional prototypes and $L = 4$. Particularly, for the Einsum Matrix multiplication, the

$$\text{Vector } u \text{ dim} = 1 \times d = 10k \quad (45)$$

$$\text{Tot. FLOPs w/o Chan. Proj.} = \text{MatMul FLOPs} + L \times \text{Cos\_Sim FLOPs} \quad (46)$$

$$= 2 \times d \times 336 + 4 \times 6 \times u \quad (47)$$

$$= 2 \times 10k \times 336 + 24 \times 10k \quad (48)$$

$$= 6.96 \text{ MFLOPs} \quad (49)$$

$$\text{Vector } u \text{ Chan. Proj. dim} = 1 \times c \times d = 3.36M \quad (50)$$

$$\text{Tot. FLOPs w/ Chan. Proj.} = \text{Einsum FLOPs} + L \times \text{Cos\_Sim FLOPs} \quad (51)$$

$$= d \times c + 4 \times 6 \times u \quad (52)$$

$$= 10k \times 336 + 24 \times 3.36M \quad (53)$$

$$= 84.00 \text{ MFLOPs} \quad (54)$$

HyperDUM FLOPs breakdown by component for aiMotive: 4x (patches) spatial projection=4*6.72M=26.88M, 4x spatial similarity=4*240K=960K, channel projection=3.36M, channel similarity=80.64M (**72% of all costs**). Uncertainty weights=(kern_h*kern_w*in_c+1)*(out_h*out_w*out_c) =(4*1*336+1)*(1*1*336)=452K. (Computation computed for Conv layer described in Architectures figure, approx. same for all methods, thus ignored).

9. Architectures

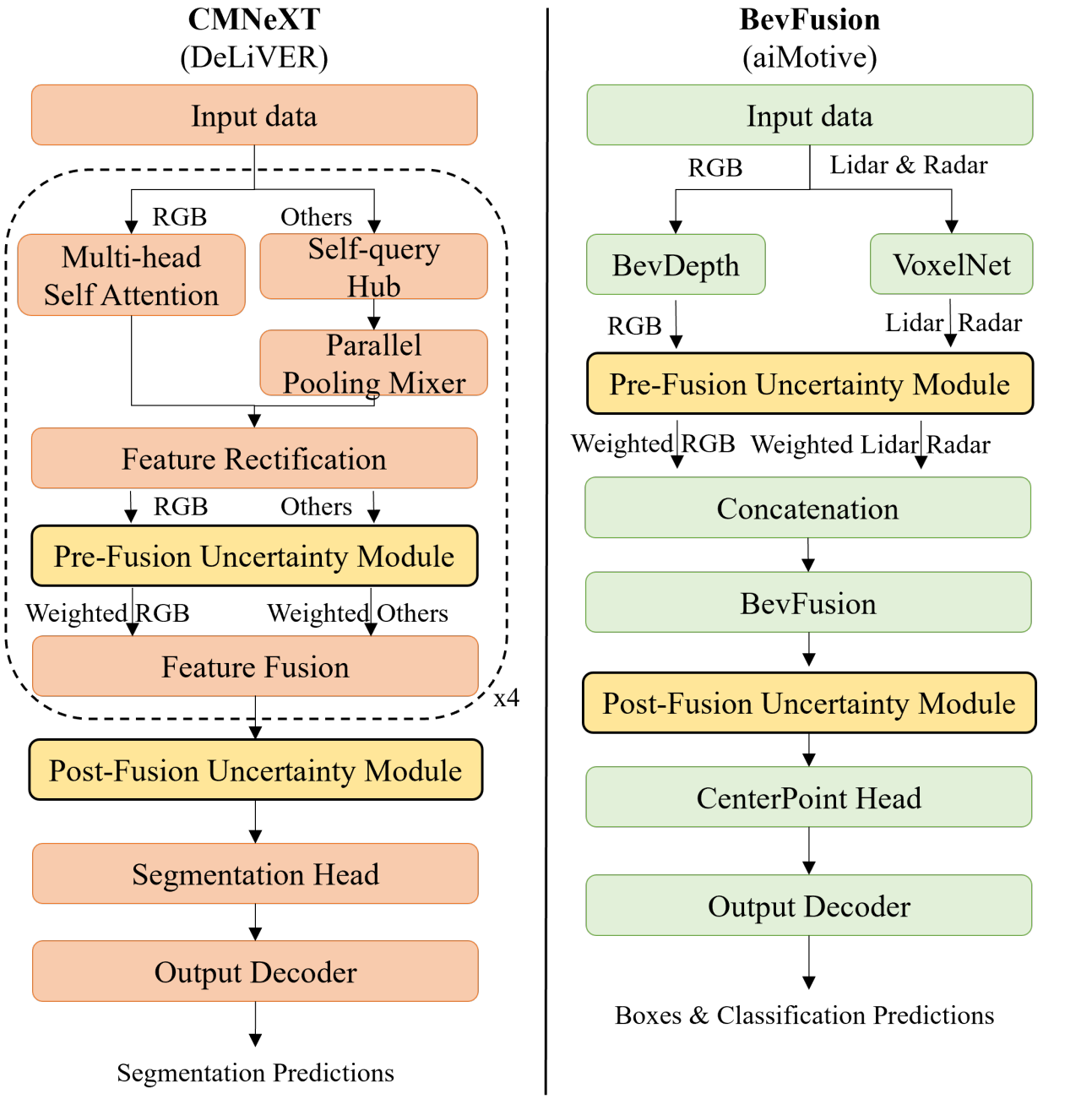

Figure 5. Architectures diagram showing where we insert the uncertainty module for pre and post fusion methods. The uncertainty weighting is a single Conv layer, Input: (B,M,P,C)/(B,M,P,1), kernel=(P,1), stride=(P,1), Output: (B,M,1,C)/(B,M,1,1) for channel/patch weights respectively. B=Batch, M=Modality, P=Prototype, C=Channel. Channel/**Patch** weights are multiplied uniformly across all spatial/**channel** dimensions per channel/**patch**. The weighting module performs dimension matching automatically.

10. Additional Experiments

UQ Method	Cloudy	Foggy	Night	Rainy	Sunny	MB	OE	UE	LJ	EL	Mean
CMNeXt [58]	53.05	54.06	50.63	54.26	51.88	49.25	49.40	47.08	52.28	53.62	53.00
InfMCD [34]	53.67	54.83	51.42	54.72	**52.37**	49.67	51.25	51.18	53.02	**54.02**	53.41
InfNoise [34]	53.25	54.37	50.85	54.52	51.95	49.39	51.08	50.86	52.83	53.37	53.19
PostNet [5]	53.38	54.10	51.13	54.39	51.89	49.48	51.15	50.99	52.68	53.20	53.15
LDU [10]	53.66	54.40	51.41	54.46	51.94	49.38	51.32	51.04	52.92	53.31	53.37
HyperDUM	**53.77**	**54.91**	**51.52**	**54.75**	**52.37**	**49.69**	**51.38**	**51.42**	**53.17**	53.78	**53.69**

Table 9. DeLiVER test set with adverse weather and corner cases: Lidar-Jitter (**LJ**), Event Low-resolution (**EL**). (Metrics: mIoU)

UQ Method	Highway	Urban	Night	Rain	Mean (Δ)
HyperDUM	**72.23/69.58**	**64.77/64.48**	**76.69/75.15**	**44.78/45.48**	**66.70/66.00**
– w/o Patch (4x) Proj.	70.74/68.55	64.35/64.15	75.53/73.83	40.09/41.69	65.67/65.23 (-1.03/-0.77)
– w/o Chan Proj.	70.62/68.94	62.03/59.90	75.11/74.04	34.01/36.21	64.43/64.25 (-1.24/-0.98)

Table 10. aiMotive ablation under diverse scenes. (Metrics: all-point AP/11-point interpolation AP)

UQ Method	MB	OE	UE	LF	Mean (Δ)
HyperDUM	**64.39/63.99**	**66.16/65.39**	**64.18/63.91**	**65.89/65.22**	**65.16/64.62**
– w/o Patch (4x) Proj.	63.58/63.35	65.26/64.62	63.38/63.40	64.85/64.42	64.27/63.95 (-0.89/-0.67)
– w/o Chan Proj.	62.65/62.43	64.76/63.16	62.29/62.22	63.60/63.02	63.47/62.83 (-0.80/-0.88)

Table 11. aiMotive ablation under corner cases. (Metrics: all-point AP/11-point interpolation AP)

UQ Method	Cloudy	Foggy	Night	Rainy	Sunny	MB	OE	UE	LJ	EL	Mean (Δ)
HyperDUM	**53.77**	**54.91**	**51.52**	**54.75**	**52.37**	**49.69**	**51.38**	**51.42**	**53.17**	**53.78**	**53.69**
– w/o Patch (4x) Proj.	53.22	54.60	51.16	54.69	52.07	49.69	51.25	51.16	53.13	53.69	53.37 (-0.32)
– w/o Chan Proj.	53.27	54.44	50.87	54.31	51.83	49.47	51.02	50.54	52.65	53.47	53.15 (-0.54)

Table 12. DeLiVER test set ablation with adverse weather and corner cases. (Metrics: mIoU)

10.1. #Prototypes & Held-Out Scenarios

Experiment	UQ Method	Cloudy	Foggy	Night	Rainy	Sunny	Mean
Baseline	CMNeXt	68.70	65.66	62.46	67.50	66.57	66.30
# Prototypes: 3	*HyperDUM*	69.50	66.00	63.35	68.28	67.17	66.94
# Prototypes: 10	*HyperDUM*	**70.04**	**66.48**	**64.07**	**68.85**	**67.67**	**67.53**
Held-Out Scenarios	InfMCD	68.61	65.93	62.37	67.12	66.90	66.29
	InfNoise	**69.59**	65.66	62.93	68.13	**66.95**	66.75
	PostNet	69.10	65.91	62.37	68.10	66.52	66.50
	LDU	69.46	65.36	62.49	67.68	66.89	66.48
	HyperDUM	69.27	**66.08**	**63.75**	**68.15**	66.81	**66.91**

Table 13. Varied prototypes and held-out experiments (DeLiVER).

Table 13 aims to evaluate the impact on performance of *HyperDUM* under two non-ideal conditions. 1) What would happen if we have poorly-defined scenarios, facilitating prototype definition under non-ideal conditions, i.e. fewer prototypes (e.g., 2 prototypes for 4/5 scenarios) and more prototypes for extremely specific data attributes. 2) What would happen if *HyperDUM* is trained without a scenario in the dataset, prototypes are computed only for the available scenarios (training data), and then the model is evaluated with samples from the held-out scenario? To test these scenarios for fewer prototypes (protos) we merged: $cloud\_fog$, $night\_rain$, sun. For more protos, we split by maps (1,2,3,6): $cloud_{1\_2}$, $cloud_{3\_6}$, fog_{1_2}, fog_{3_6}, $\cdots$. Table 13 shows that fewer protos by **careless merging** achieve suboptimal performance, while fine-grained protos improve performance. Merging different underlying uncertainty attributes (cloud/fog) introduces ambiguity (what is a cat/dog hybrid?). Protos should be formed by samples that share all underlying uncertainty attributes. **Praticality:** Well-curated datasets, i.e., nuScenes, Waymo, KITTI, etc. all capture meta-data facilitating prototype definition. With no/limited meta-data, one can apply unsupervised methods (i.e. clustering) to identify prototypes or generate new prototypes using diffusion models. **Held-out Scenarios:** Average performance drops as expected, but compared to others, *HyperDUM* maintains competitive performance. Particularly in more uncertain (fog/night/rain) held-out scenarios.

10.2. Uncertainty Visualization

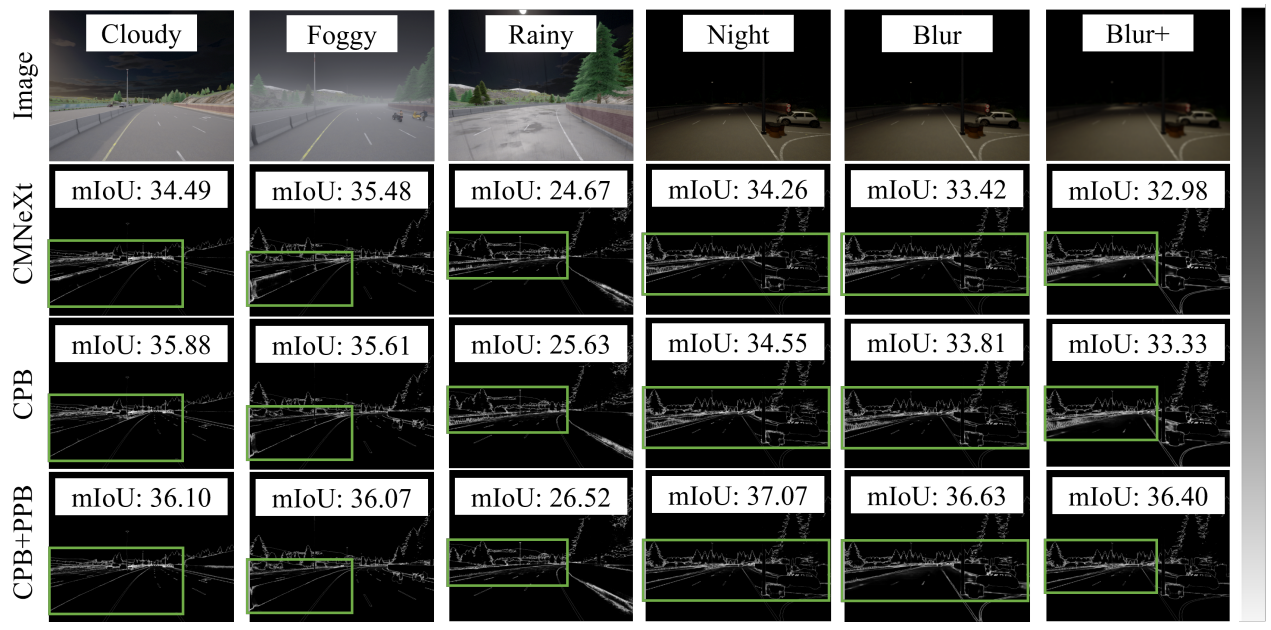

Figure 6. Channel/Patch Projection & Bundling (CPB/PPB) effects visualization on prediction uncertainty map (DeLiVER).

Figure 6 ablates the channel and spatial uncertainty weighting performance through the prediction uncertainty map across scenarios and noise effects (Gaussian Blur 7x7 and 25x25). Generally, channel sharpens and spatial refines the uncertainty maps in-detail. Additionally, we see that CPB and PPB together improve the overall robustness to Gaussian blur noise when compared to the baseline model.

10.3. Fine-Tuning Details

Hyper-parameter	aiMotive	DeLiVER
Architecture	BEVFusion	CMNeXt
backbone	ResNet	CMNeXt-B2
learning rate	1e-3/64	6e-5
batch size	1	1
epochs	200	200
EarlyStopping (epochs)	10	10
weight decay	1e-7	0.01
Scheduler	MultiStepLR	warmuppolylr
Prototypes	4	5
Forwards (InfMCD & InfNoise)	10	10
Projection Type	Ortho. Rand. Fourier Proj.	Ortho. Rand. Fourier Proj.
Channel Hyperdimension	10000	5000
Patch Hyperdimension	10000	5000
Number of Patches	4	4

Table 14. Hyper-parameter configuration used for aiMotive and DeLiVER Fine-tuning. The projection function uses Random Fourier Features (rfflearn python) similar to [57] which has no learned parameters. Prototype formation uses the train set (100%) but can be a subset (importance selection), with **only one pass through train data** instead of multi-epochs.

10.4. Artifical Noise Injection Paramters

Noise Injection	Configuration Setting
Over Exposure	rescale_intensity(data, in_range=(0, int(np.max(data)/2)), out_range=(0,255)).astype(uint8)
Under Exposure	rescale_intensity(data, in_range=(int(np.max(data)/2), int(np.max(data))), out_range=(0,255)).astype(uint8)
Motion Blur	GaussianBlur(kernel_size=(25,25), sigma=16)
Foggification	BetaRandomization(beta=1e-5)

Table 15. Configuration for aiMotive artificial noise injections. GaussianBlur from torchvision.transforms, rescale_intensity from skimage.exposure, for foggification refer to [3]. These injections are only performed on the test set during evaluation and never seen during training/validation.