Fundamentals

Computer Vision: The inverse algorithmic process of inferring world/sensor properties from observations. Light source → optical medium → optical sensor + object → observation → CV algorithm

Dynamic Driving Task (DDT): Lateral/longitudinal control, object/event detection, recognition, classification, response (OEDR), maneuver planning, enhancing conspicuity via lighting, horn, signaling. NOT trip planning. ⇝ Perception, Prediction, Planning, Control

Driving automation System: Hardware/software performing parts of DDT.

Automated Driving System (ADS): Hardware/software performing whole DDT.

Operational Design Domain (ODD): Specific conditions for ADS function.

Autonomous Car (ADS-DV): Driverless operation within ODD.

SAE levels of automation: 0 ⇝ none, 1 ⇝ driver assistance, 2 ⇝ partial driving automation, 3 ⇝ "eyes off" but human fallback, 4 ⇝ high driving automation with limited ODD, 5 ⇝ full, unlimited ODD

Sensors

External: Camera, LiDAR, Radar, Event Camera, GPS/IMU

Internal: Camera, Thermometer

Light Detection and Ranging (LiDAR): active, near-infrared, direct range via time-of-flight.

Pulsed LiDAR: Precision ≈ 1.5cm. Range depends on signal-to-noise ratio, today ≈ 150m. Max range $\propto \frac{1}{\text{emission frequency}}$

$$P_R(R) = C_A \int_0^{2R/c} P_T(t)\, \mathrm{H}\left(R - \frac{ct}{2}\right) dt$$

Point Cloud acquisition by scanning spherical coordinates

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r\cos\delta\cos\varepsilon \\ r\sin\delta\cos\varepsilon \\ r\sin\varepsilon \end{pmatrix}$$

Motorized Optomechanical Scanner or MEMS
↳ 360° horizontal FOV, but low frame rate and mechanical wear
↳ small size, no wear, higher rate but <120° FOV.

Challenges: Motion Distortion, non-Lambertian surfaces, optical attenuation, back-scattering, interference

Camera:

Pinhole model: $\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z_c} K \overbrace{[R|t]}^{\text{extrinsics}} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}$ for intrinsics $K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$

Radar: 70–80 GHz spectrum robust to fog, rain, snowfall.

FMCW scanning radar has $P_{rec} = \frac{P_{trans} G^2 \lambda^2 A_e}{16\pi^2 r^2 L}$

Modulate frequency and look at intermediate frequency signal

Event Camera: Asynchronous events at single pixels with low latency

$\Delta L(x_k, t_k) := L(x_k, t_k) - L(x_k, t_k - \Delta t_k) \gtrless \pm C$ irradiance change

Deep Neural Networks

Activation of neuron j in layer ℓ: $z_{\ell,j} = w_{0,j}^{(\ell)} + \sum_{i=1}^{d_{\ell-1}} w_{i,j}^{(\ell)} g(z_{\ell-1,i})$

where width of layer ℓ is d_ℓ and g is nonlinear activation funct:

Sigmoid $g(z) = \frac{1}{1+e^{-z}}$ ∫ $g'(z) = g(z)(1-g(z))$

Tanh $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ ∩ $g'(z) = 1 - g(z)^2$

ReLU $g(z) = \max(0, z)$ ⎍ $g'(z) = [z > 0]$

Binary Classification: Predict label $y \in \{-1, 1\}$ from logit $NN(x) =: z$.

Assuming $\log \widehat{P}(y|x) = yz$ yields $P(y|x) = \frac{\exp(yz)}{\exp(-z)+\exp(z)} = \text{Sigmoid}(yz)$.

Minimize $-\log P(y|x) = \log(1 + \exp(-yz)) = \text{Softplus}(-yz)$.

Multiway Classification: Predict label $y \in [n]$ from logits $NN(x) =: z$.

Assuming $\log \widehat{P}(y=i|x) = z_i$ yields $P(y=i|x) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} = \text{softmax}(z)_i$.

Minimize $-\log P(y=i|x) = \log\left(\sum_j \exp(z_j)\right) - z_i$

Supervised learning: Minimize empirical risk $J(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(f_\theta(x_i), y_i)$, usually via SGD. Batch size n yields standard error $SD/\sqrt{n}$. Updates with momentum $v \leftarrow \alpha v - (1-\alpha)\eta_t \nabla_\theta J_B(\theta)$, $\theta \leftarrow \theta + v$ and learning rate $\eta_t = \eta_0/(1+kt)$ (time) or $\eta_t = \eta_0 \exp(-kt)$.

Vanishing/exploding gradient Solutions:

Weights are initialized randomly around zero with larger means deeper in the network to avoid vanishing/exploding gradients (Kaiming).

Input is normalized: For channel j, $\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$.

Pre-activation batch normalization:

Learn γ and β: For batch B, $\hat{x}_i = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta$

Prepare for inference: $\gamma \leftarrow \frac{\gamma}{\sqrt{\text{Var}[x] + \varepsilon}}$, $\beta \leftarrow \beta - \frac{\gamma E}{\sqrt{\text{Var}[x] + \varepsilon}}$

Regularization: Early stopping when validation loss is low, weight-based penalty, data augmentation (rotation, shearing, warping, cropping, photometric)

Convolutional Neural Networks

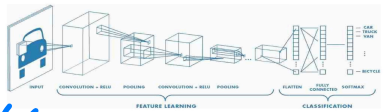
Convolutional Layer: A kernel has size $d_{\ell-1} \times m_\ell \times n_\ell$ and produces a 2D feature map by summing the per-channel convolution. Then, add d_ℓ biases to the result. So, $d_\ell (d_{\ell-1} m_\ell n_\ell + 1)$ parameters.

$$z_{\ell,c}[p,q] = \sum_{k=1}^{d_{\ell-1}} \sum_{i=1}^{m_\ell} \sum_{j=1}^{n_\ell} w_{k,c}^{(\ell)}[i,j]\, g(z_{\ell-1,k}[sp-1+r(i-1), sq-1+r(j-1)]) + b_c$$

stride s_ℓ increases receptive field of subsequent neurons non-uniformly, dilation r_ℓ uniformly sparsely. $H_\ell = \left\lfloor \frac{H_{\ell-1} + 2p_{\ell-1} - r_\ell m_\ell}{s_\ell} \right\rfloor + 1$

Pooling Layer: Slide window over image and get min/max/mean.

AlexNet Architecture:



ResNet adds skip connections.

Semantic Segmentation

For inputs $X \in \mathbb{R}^{H \times W \times 3}$ compute $\hat{Y} \in \mathbb{R}^{H \times W \times C}$ where $Y[h,w,c]$ is the softmax-probability of $x[i,j]$ belonging to class c. Optimize logits by minimizing $L_{CE}(\hat{Y}, Y) = -\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} Y[h,w,c] \log(\hat{Y}[h,w,c])$.

One-hot encode Y.

Mean Intersection over Union: $\text{mIoU}(\hat{Y}, Y) = \frac{1}{C} \sum_{c=1}^{C} \frac{\sum\sum [\hat{Y}[h,w]=c \wedge Y[h,w]=c]}{\sum\sum [\hat{Y}[h,w]=c \vee Y[h,w]=c]}$

Evaluation: Cityscapes, Mapillary Vistas, Foggy Cityscapes, ACDC, GTAE

Bad Approaches: Sliding window (expensive, no global context), CNN without striding, dilation, or pooling (overfitting)

Encoder-Decoder via Upscaling:

Transpose convolution (flip signs) OR Unpooling (via switch variables)

UNet: Encoder-Decoder with upscaling and concatenation of encoder features in decoder at the same size.

Transformer: Allow model to learn receptive field.

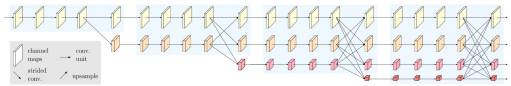
Attention $(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

Multihead $(Q,K,V) = \text{Concat}\left(\text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\right)W^O$

Complexity is $O(BN^2D)$ in time and $O(BN^2 + BND)$ in space

Swin Transformer: Shifted Window Attention as drop-in replacement for convolutional layers. Cross-window connections arise from offsetting windows in deeper layers.

HRNet: Parallel multi-resolution convolutions



Depth Estimation

Monocular from shading, texture, focus (, motion) can be framed as regression task. Use ℓ_1, ℓ_2 loss or $\text{BerHu}(x) = \begin{cases} |x| & \text{for } |x| \le c \\ \frac{x^2+c^2}{2c} & \text{otherwise} \end{cases}$.

Error metrics are rel. difference or RMS $= \sqrt{\frac{1}{T}\sum_{t \in A} |y_t - y_t^*|^2}$.

Deep Ordinal Regression: Discretize depth at points $t_0, t_1, \ldots, t_{K+1}$ and predict $\mathcal{Y}^k = P[\hat{\ell} > k]$. Pixel-wise loss is $\sum_{k=0}^{L-1} (\log(\mathcal{Y}^k) + \sum_{k=L}^{K-1} \log(1 - \mathcal{Y}^k))$.

Inferred depth is $\hat{d} = \frac{t_{\hat{\ell}} + t_{\hat{\ell}+1}}{2}$ for $\hat{\ell} = \sum_{k=0}^{K-1} [\mathcal{Y}^k > 0.5]$.

Plane Coefficient Representation: $z^{-1} = \underbrace{\frac{-a}{f_x d}}_{\hat{\alpha}} u + \underbrace{\frac{-b}{f_y d}}_{\hat{\beta}} v + \underbrace{\frac{1}{d}\left(\frac{a}{f_x} u_0 + \frac{b}{f_y} v_0 - c\right)}_{\hat{\gamma}}$

Normalize by dividing with $\rho = \sqrt{\hat{\alpha}^2 + \hat{\beta}^2 + \hat{\gamma}^2}$. Then, learn $(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\rho})$.

P3 Depth: Learn offset δ s.t. $p+\delta$ have the same plane coefficients.

Stereo Depth from disparity: $u' = u + b$. Then, $x_w = b\frac{u}{u-u'}$, $y_w = b\frac{f_x}{f_y}\frac{v}{u-u'}$, $z_w = b f_x \frac{1}{u-u'}$. Increasing b and f increases depth resolution but reduces overlapping FOV. Algorithms match and aggregate cost, optimize, and refine disparity of image patches. Siamese network shares weights, then computes 4D cost volume (h, w, disparity+1, 2·features). Regress disparity by linear combination of softmax bucket scores.

Unsupervised stereo depth: Train monocular estimator and verify by warping and comparing to true stereo image (left-right consistency) OR jointly optimize pose-from-motion and depth prediction from video.

Object Detection

Detect bounding boxes (or masks for instance segmentation) and classes.

Common data sets: PASCAL VOC (small), COCO (large)

Average Precision (AP): Precision $p = \frac{TP}{TP+FP}$, Recall $r = \frac{TP}{TP+FN}$,

$AP = \int_0^1 p_{int}(r)\, dr$ where $p_{int}(r) = \max_{\tilde{r} \ge r} p(\tilde{r})$ is decreasing stepwise.

$mAP = \frac{1}{C} \sum_{c \in} \frac{1}{|\text{IOUs}|} \sum_{\text{IOU}} AP(\text{IOU})$. Pascal VOC uses 11 point avg.

Slow R-CNN: Region Proposal → Crop & Warp → CNN → {Linear classifier → Task; Box regressor → Task} heads

SPP-net: Large CNN Region Proposal → RoI Pooling → Small CNN → Task heads

Multi-Task loss: $L(p, u, t^u, v) = \underbrace{-\log p_u}_{\text{Lcls}} + \underbrace{\lambda [u \ge 1]}_{\text{not BG}} \underbrace{\sum_{i \in \{x,y,w,h\}} \text{Huber}(t_i^u - v_i)}_{\text{Lloc}}$

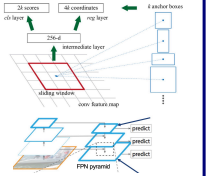
where $\text{Huber}(x) = \begin{cases} 0.5x^2 & \text{for } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$

Faster R-CNN: Large CNN → Region Proposal → RoI Pooling → MLP → {Task heads}

RPN is a small MLP that is applied sliding-window style over k crops to predict objectness & box offsets.

Label objectness truth $p_i^* = \begin{cases} 1 \text{ if } \exists \text{ box. IoU} > 0.7 \\ 0 \text{ if } \forall \text{ box. IoU} < 0.3 \\ \text{None otherwise} \end{cases}$

$L_{RPN}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(A_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$



Feature Pyramid Network (FPN): Append prediction head to U-Net decoder layers.

Direct set Prediction for Object Detection: Instead of non-maximum suppression, compute bipartite matching via Hungarian algorithm.

Detection Transformer (DETR): Encoder-decoder attention and N object queries which are processed one-by-one by a Feed-Forward Network.

Instance Segmentation

Mask R-CNN: Detect-then-segment, so ... → RoIAlign ⇉ MLP → Classifier, Box regressor; FCN → Segmentation loss.
RoIAlign performs interpolation. Masks are upscaled and unstretched.
⊖ low-res, expensive inference, near-duplicate predictions

Spatial Mappings: Proposal-free method.
CNN ⇉ seed CNN ~ per-class seed map → Sample
⇉ Instance CNN ~ ∇-map for size and pixel offsets for instance → clusters
$\phi_k(e_i) = \exp(-\|e_i - c_k\|^2 / (2\sigma_k^2)^2)$

Panoptic Segmentation: Countable instances AND uncountable classes without overlap. Predict $Y_i \in \{1, \dots, C\} \times \mathbb{R}^2, \dots, \mathbb{R}^2, \phi\}^{H \times W}$

Panoptic Quality: Each GT segment $PQ = \underbrace{\frac{\sum_{TP} IoU(A,B)}{|TP|}}_{\text{recognition quality}} \cdot \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}(|FP| + |FN|)}}_{\text{segmentation quality}}$
can have ≤1 predicted segment with IoU > 0.5. The unique matching yields segmentation quality F1-recognition quality

MaskFormer: Variant of DETR with parallel pixel-level module.
⊖ Slow, not SOTA in instance OR semantic, trained per-task.

OneFormer: Provide textual queries for the model to search for: Segmentation ⇒ ["car", "person", "road"], Panoptic: 8 · ["car"] + ["road"].

3D Sensing

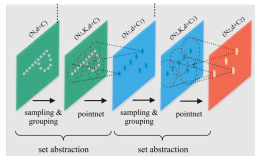
Multi-view CNN: Render to 2D and run CNN per-image. Then, pool shape descriptors with another CNN. ⊖ linear in number of views, unclear how noisy / incomplete renders are handled.

Volumetric CNN: Work on voxel representation. ⊖ information lost during Voxelization, Sparsity increases with grid resolution.
Alternative: Oct Trees for recursive partitioning.

PointNet: Process scattered, unordered, noisy point clouds.
Input Transform → MLP → Feature Transform → MLP → Max Pooling → Classifier (pointwise); ⇉ Concat → Segmentation Head
Transforms are learned matrices with the goal of finding a canonical representation for geometric invariance. Permutation invariance from MaxPool.
⊖ No local context for each point, global features depend on coordinates.

PointNet++: Iterative furthest Point sampling → Grouping via ball query → PointNet → ... ⇉ Segmentation, Classifier
$N \times (d+C) \to N' \times d$ centroids $\to N' \times K \times (d+C) \to N' \times (d+C')$

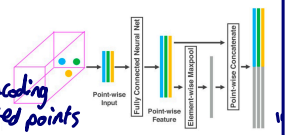
For segmentation, upscale with skip connection in interpolate layer. K-nearest neighbors' features are weighted by distance: $w_i = \frac{1}{\|x_i - x_c\|_p}$



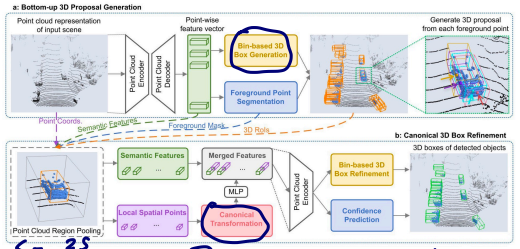
3D Object Detection

Birds Eye View (BEV): 2D top-down lidar.

VoxelNet: Object proposals from voxels.
Partition → Group & Sample → Stacked Feature Encoding on centroid-augmented points → 3D-CNN (with BatchNorm & ReLU) → RPN
Trained via Faster-RCNN loss. For regression, use $\Delta x = \frac{x^g - x^a}{d^a}, \Delta l = \log(\frac{l^g}{l^a}), \Delta\theta = \theta^g - \theta^a$



PointRCNN: 2-Stage detector with 3D proposal, then refinement.
Bin-based 3D box generation: Establish a grid around each foreground point. Classify bin-x, bin-y, bin-θ and regress $z, h, w, l, \Delta x, \Delta y, \Delta\theta$.
$bin_x^{(p)} = \left\lfloor \frac{x^p - x^{(p)} + S}{\delta} \right\rfloor$ with $\delta = \frac{2S}{\#bins}$
$res_x^{(p)} = \frac{1}{c}(x^p - x^{(p)} + S - (bin_x^{(p)} \delta + \frac{\delta}{2}))$.
The per-point loss for bins is $\mathcal{L}_{cls} + \mathcal{L}_{reg}$ } ⇒ more robust coarse localization of center point.

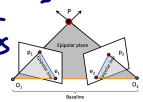


3D Localization & Reconstruction

Iterative Closest Point (ICP): Obtain $[\hat{R}|\hat{t}]$ and assignments $\hat{\phi}: [n] \to [n]$ such that $E(R,t,\phi) = \sum_i \| R p_S^{(\phi(i))} + t - p_S^{(i)} \|_2^2$ is minimized. Solved iteratively by guessing, computing ϕ by closeness, then updating $[R|t]$ until convergence.
closed-form $\hat{t} = \mu_S' - \hat{R}\mu_{S,\phi}$ (align centroids), optimal rotation by aligning principal 3D directions of variation. (Split $R = UV^T$ and maximize trace $(U^T W V)$ where $W = \sum_i (p_S^{(\phi(i))} - \mu_{S\phi})(p_S^{(i)} - \mu_S)^T = \hat{U}\Sigma\hat{V}^T$. Eliminating reflections yields $\hat{R} = \hat{V}\begin{bmatrix} 1 & & \\ & \ddots & \\ & & \det(\hat{U}\hat{V}^T) \end{bmatrix}\hat{U}^T$.) ⊖ sensitive to outliers, expensive

Visual Odometry: Estimate relative 3D Euclidean motion of a camera from its captured images. Motion from I_{k-1} to I_k is $[R_{k,k-1} | t_{k,k-1}]$.

2-frame structure from motion: Epipoles e_1 and e_2 are the image-plane intersections of the baseline offset $\overrightarrow{O_1O_2}$. Potential matches p_2 to p_1 must lie on the epipolar line $\overrightarrow{e_2 p_2}$:
$p_2^{(2)} (t \times p_2^{(-1)}) = (p_2^{(2)})^T [t]_\times R p_1^{(2)} \stackrel{!}{=} 0$

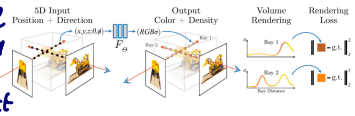


Computing the essential matrix $[t]_\times R$ needs 5 point correspondences at minimum. The 8-point algorithm yields 4 solutions which are filtered by triangulation.

RANSAC: Randomly select minimal correspondences. Estimate structure from motion. For all other points, compute distance to induced epipolar line. Store inliers. Repeat N times. Finally, compute with the largest set of inliers.

SuperGlue Feature Matching: Compute pairwise feature similarity matrix via Attentional Aggregation. Then run Sinkhorn Algorithm to refine into soft assignment matrix where $\sum_{j=1}^{M} p_{ij} = 1$.

Neural Radiance Fields (NeRFs): Learn volume rendering densities from posed images via MLPs. Observed color for ray $r(t) = o + td$ is $C(r) = \int T(t)\sigma(r(t))\, c(r(t), d)\, dt$. Predict via stratified quadrature: $\hat{C}(r) = \sum_i T_i (1 - \exp(-\sigma_i(t_{i+1} - t_i)))c_i$.
Here, $T(t) = \exp(-\int^t \sigma(r(s))ds)$ is the accumulated transmittance and c color.
To improve resolution, use positional embeddings in >5D space for input.



NeRF in the wild: Robust learning of static densities and colors from images taken on different days by learning per-image transient embeddings
Block NeRF: Partition based on visibility.

Domain Adaptation

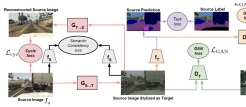
Domain shift: Formally, images I and ground truths Y are sampled from domains S and t: $(I_S, Y_S) \sim p_S(I, Y)$ and $(I_t, Y_t) \sim p_t(I, Y)$. When $p_S \neq p_t$, we have domain shift. E.g., weather conditions, sensor setup, real vs. synthetic, object semantics changing in different countries. (low → high)

Weakly Supervised DA: Label correspondences between source and target inputs, assuming similar outputs.

Source-Free DA (pre-test), Online DA (test-time), Domain Generalization (pre-training)

Input-level DA: Foggy Cityscapes (synthetic fog based on depth estimation), LiDAR snowfall, image translation with CycleGANs, Fourier Domain Adaptation.

CyCADA: Adversarial loss at feature-level to make differentiation from target domain impossible.



$L_{GAN,feat} = \mathbb{E}_{I_t \sim X_t}[\log D_{feat}(P_T(I_t))] + \mathbb{E}_{I_S \sim X_S}[\log(1 - D_{feat}(P_T(G_{S \to T}(I_S))))]$

CISS: Contrastive learning of Siamese networks with shared feature.
CMA: Patch-wise learning with contrastive embeddings.

Output-level DA:
AdaptSegNet: Optimize for quality and indistinguishability in an adversarial setting. ⊖ Bias towards easily transferable classes.
DACS: Mix images from two domains together based on their masks, hoping to eliminate implicit training of per-domain submodels. (e.g. put Mexican Taxis on Berlin streets)

Probabilistic Warping: Learn deterministic warps to find correspondences.

Test-time adaptation via TENT: Optimize batch norm statistics and parameters to better reflect the test-set by minimizing entropy.

Multi-Modal Detection

Sensor Fusion: Late (fuse results), early (fuse inputs), deep (fuse features), asymmetric, weak.

Frustum PointNet: Use camera to constrain 3D search space.

MV3D: Generate proposals from BEV for front-view LiDAR, BEV, and camera. Then, deeply fuse sensors.

PointPainting: Color point clouds from camera image pre-input.

TransFusion: Use attention to let model decide where to look up camera.

CLOC: Late fusion of camera-lidar object candidates.

CRF-Net: Camera-radar fusion by faking height with pillars.

Entropy-based fusion vs. MWC-Attention: Both quadratic in time, former might fail with snow.

Misc. Improvements

Visual Grounding: Given a scene and a verbal query, detect the referred object in the scene. 3DVG-T does verbo-visual fusion with transformers. ⊖ Spatial specifications ambiguous ("left of x").
Solved via multi-view encoding (MVT-3DVG) or learned camera embeddings (Concrete Net)

Anomaly Segmentation: Find areas of epistemic (model-related) uncertainty, not aleatory (data-related) uncertainty, by doing Monte-Carlo integration using dropout (Find large $-\frac{1}{c}\sum_c(\frac{1}{T}\sum_t y_c^t) \log(\frac{1}{T}\sum_t y_c^t) - \frac{1}{T}\sum_{c,t} y_c^t \log y_c^t$) OR image resynthesis with discrepancy network.

Vehicle-to-Vehicle Communication for Perception (V2V Net): Maximize informativeness facing unknown time delays and limited bandwidth

Object Tracking & Path Prediction

Tracking by detection: In both frames, detect objects. For each box, compute features. Associate boxes by feature similarity.

MOT metrics: Mostly tracked trajectories (>80% of frames), fragments (>80% of trajectories), ID switches
$MOTA = 1 - \frac{FN + FP + IDSW}{GT} \in (-\infty, 1]$

MOTR: End-to-end, DETR inspired prediction of object sequences.

CNN-based path prediction: Encode past frame features as images and perform sensor fusion with them.

RNN-based path prediction: Use Long Short-Term Memory encoders.