Self-driving cars need to drive billions of kms before they can surpass human driving capabilities. The quality of those kilometres also matters – it must involve complex driving scenarios, including pedestrians, drivers, other participants, different weather conditions etc. If we relied on supervised learning alone, where we manually label all those sensor data, it'd take zillion more years for us to reach Level 5 autonomy.

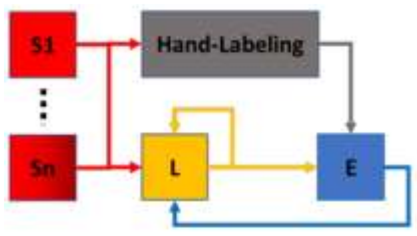
True, we have nuScenes, Waymo, Argoverse and many other datasets, focussed on Self-driving cars. But, even their size (~100 hour) is significantly lesser than what modern deep learning algorithms demand.

The cost associated with annotation and verification of such complex dataset is also very high. There are multiple cameras, lidar point clouds, and in some cases radar, which need to be labelled w.r.t space and time, and for multiple applications – Object Detection, instance segmentation, Semantic segmentation etc. It's almost impossible to scale certain tasks like Point Cloud segmentation, without some kind of intelligence / automation.

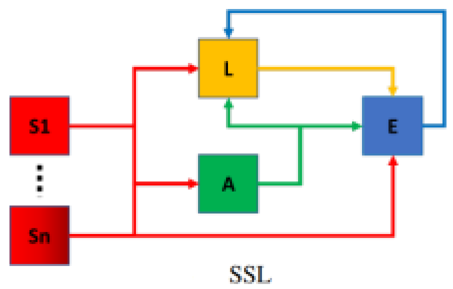
This is where Non supervised learning methods play a crucial role. There are giant strides made in Unsupervised, Self-Supervised, Semi-supervised, Weakly supervised learning methods. Each of these methods have its own set of advantages and disadvantages. We'll focus on Self-Supervised Learning (SSL) in this post

Self-Supervised Learning

SSL is method where model / algorithms are trained without manually annotated labels. ***It's different from Unsupervised learning, where there are no labels to begin with.*** In Self-supervised learning, training data is generated by exploiting the inherent structure or relationships within the data, enhancing their adaptability to diverse and real-world driving conditions.



Supervised learning



SSL

- S1 = Reference sensor
- Sn = 'N' the sensor
- A = Analytical method
- L = Learning model
- E = Evaluation

Image reference

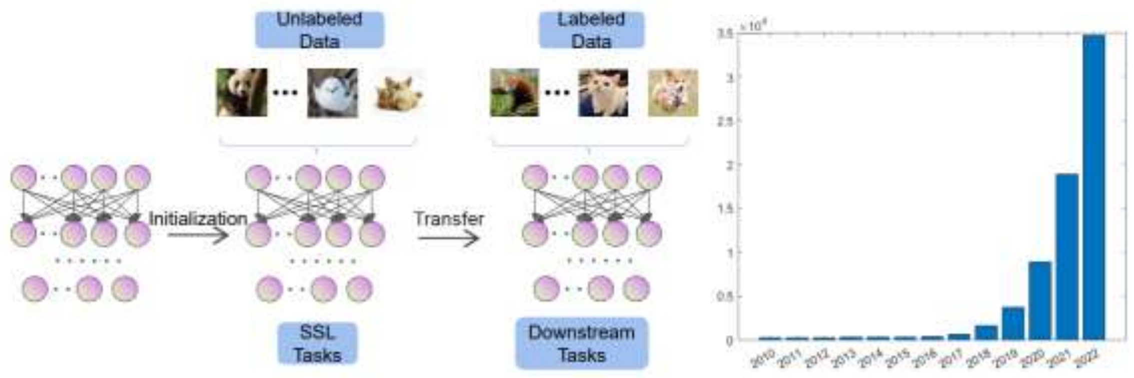


Image reference

Usually, the network is made to learn for a pretext task ("pretraining"), on a large dataset, without any manual annotations. In pretext task, part of data is withheld and network has to predict it. Features / representations learned on the pretext task are subsequently used for different downstream tasks, usually, where some annotations are available.

One of the initial "pretext" tasks was to predict the rotation of image. We can generate a total of four "image-rotation label" pairs from any given image, just by rotating the image in multiples of 90 degree. We can train any model to predict the "rotation class" from the input image.

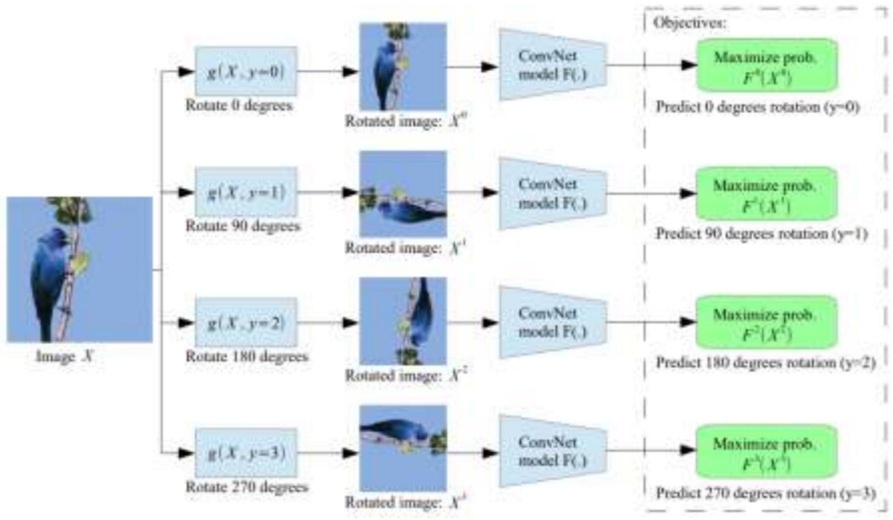


Image X
$g(X, y=0)$
Rotate 0 degrees
$g(X, y=1)$
Rotate 90 degrees
$g(X, y=2)$
Rotate 180 degrees
$g(X, y=3)$
Rotate 270 degrees
Rotated image: X^0
Rotated image: X^1
Rotated image: X^2
Rotated image: X^3
ConvNet model F(.)
Objectives:
Maximize prob. $F^0(X^0)$
Predict 0 degrees rotation (y=0)
Maximize prob. $F^1(X^1)$
Predict 90 degrees rotation (y=1)
Maximize prob. $F^2(X^2)$
Predict 180 degrees rotation (y=2)
Maximize prob. $F^3(X^3)$
Predict 270 degrees rotation (y=3)
!

Image reference

A model, trained such Classification task, can then be fine tuned for the actual task, like Object detection / Segmentation. The dataset size in the downstream task would be much lower, compared to the original "pretext" task

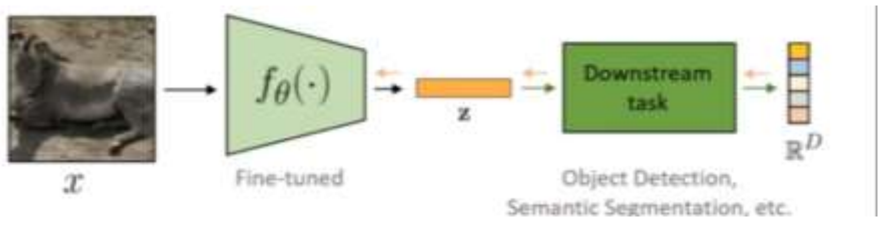


Image reference

Keeping with recent trends, Self-Supervised learning tasted success in NLP (Natural Language Processing), before gaining traction in the Computer Vision space. Following are the primary reasons for great success of LLMs (Large Language Models).

- Scalable Network Architecture: Transformers rule the world
- Learn from large volumes of internet data (Self-supervised pre-training on "unlabelled" data)

SOTA models are focussing on leveraging same to Computer vision (CV) models

NOTE: Recent works, propose training directly for downstream task of interest

Advantages

- Self-supervised models can be more efficient than supervised counterparts

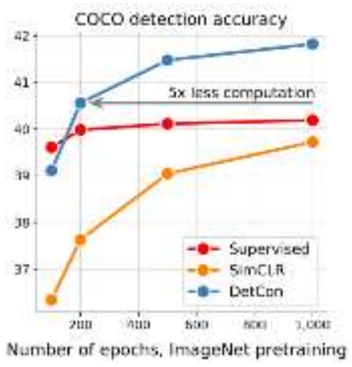


Image reference

- Data efficient image recognition with contrastive predictive coding

Need for SSL

- Collecting and labelling data is costly, particularly in context of Autonomous driving
- Datasets evolving continuously – environment changes, data distributions changing (long tail problem). So, need to efficiently adapt to changing data
- Sensor specs change (e.g. increasing camera resolution, Halogen vs LED headlamps)
- Some tasks are very difficult / impossible to manually annotate for real-life datasets. Eg: depth estimation, we can project calibration lidar point cloud to images, but still sparse, even KITTI, Cityscapes datasets use this approach).

Opportunities for SSL

- Availability of large, complex and repetitive scenes
- Importance of time and multi-modality
- Knowledge transfer over large space and time spans
- Making models more robust. Some scenes, unseen during the training, can appear frequently in the context of the autonomous vehicle. For instance, an accident on the road can change drastically the appearance and the location of potential obstacles. Thus, even if it is possible to predict when the model does not know what it observes, it may be interesting to confirm it through an analytical process and to adapt the learning model to this novel situation

We’ll briefly look at different types of Self-supervised learning, before going into detail about how they can be applied to Self-driving car applications

Types Self-Supervised Learning

SSL variations can be summarized as follows:

1. Predict any part of the input from any other part
2. Predict the future from the past
3. Predict the invisible from the visible
4. Predict any occluded, masked, or corrupted part from all available parts

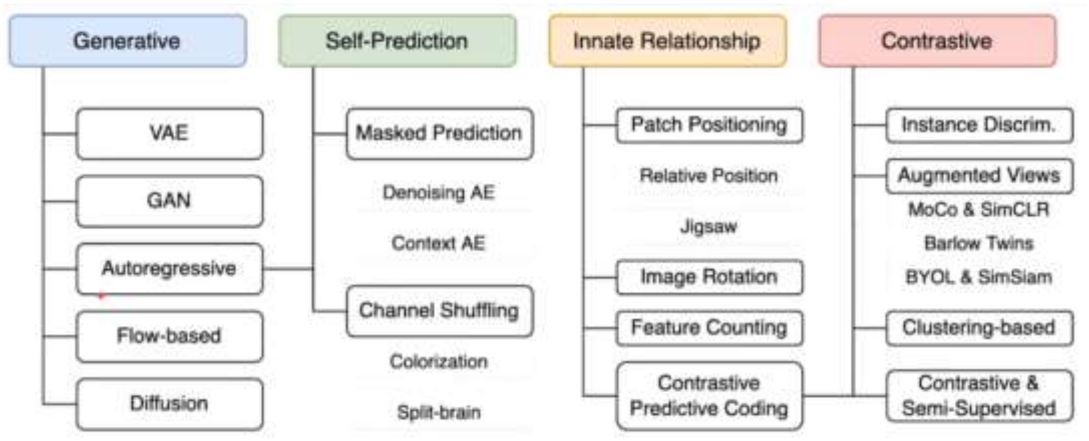


Image reference

Context-based / Innate relationship

Context-based methods rely on the inherent contextual relationships among the provided examples, encompassing aspects such as spatial structures and the preservation of both local and global consistency

θ=?
Rotation
Jigsaw
Colorization

Image reference

Common pretext tasks include

- **Rotation prediction**
- **Jigsaw:** Images are fragmented into discrete patches, and their positions are randomly rearranged, with the objective of reconstructing the original order
- **Colorization:** Taking RGB colour space as reference, with ‘R’ channel as input, the model has to predict the ‘GB’ channels. The *L2* norm is commonly used as loss function

Contrastive Learning

Contrastive learning involves training a model to bring similar instances closer and dissimilar instances apart in a feature space.

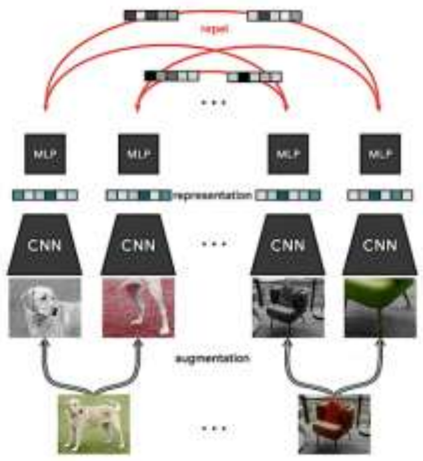


Image reference

This can enhance discriminative features for object detection.

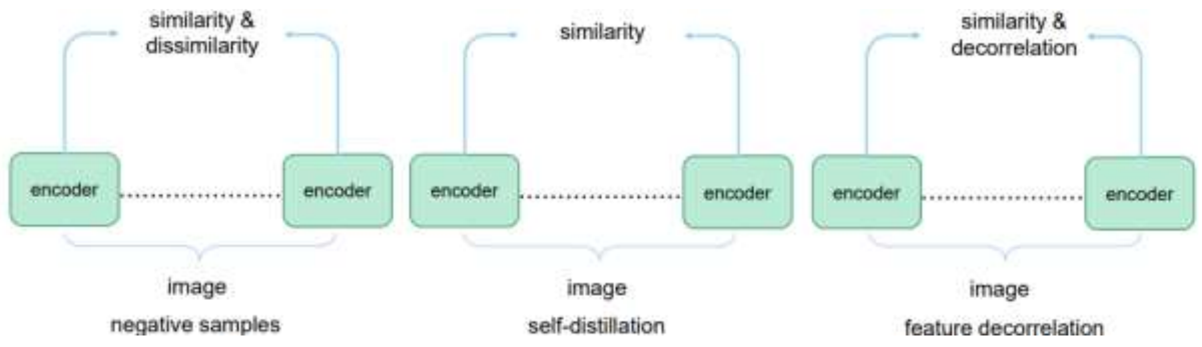


Image reference (A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends)

Earliest CL models were Negative-sample based. Views originating from the same instance are treated as positive examples for an anchor sample, while views from different instances serve as negative examples. The underlying principle is to promote proximity between positive examples and maximize the separation between negative examples within the latent space.

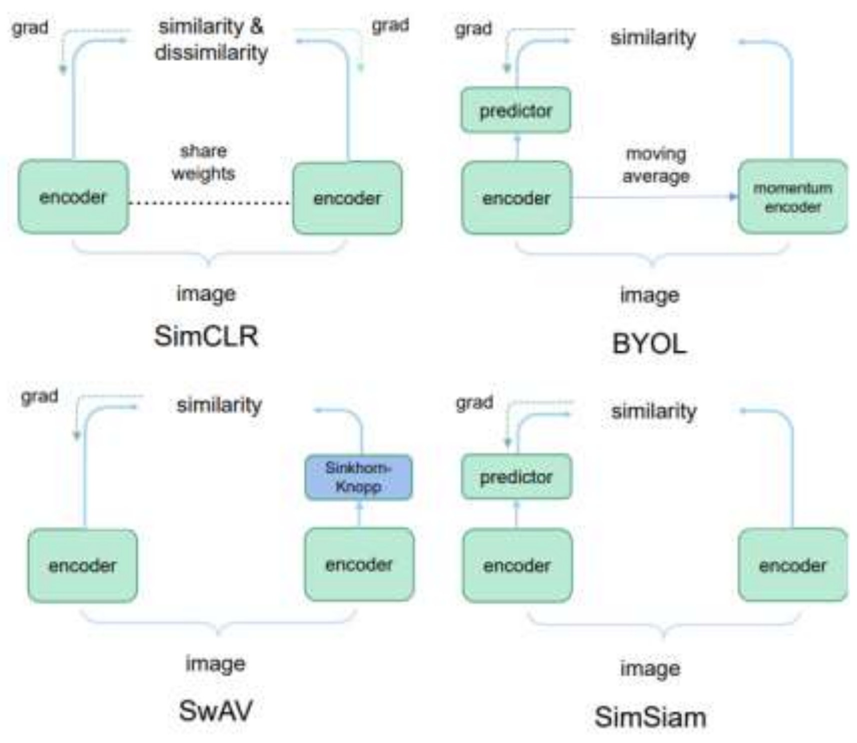
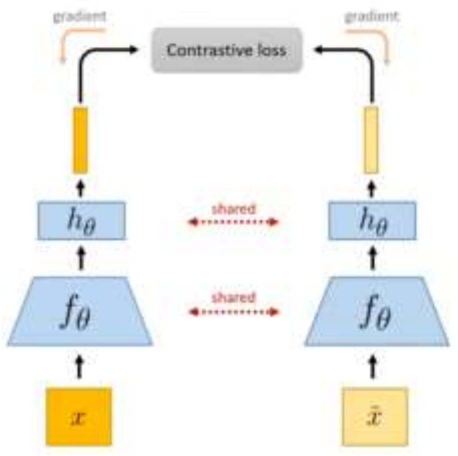


Image reference (A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends)

Negative Sampling based CL

SimCLR

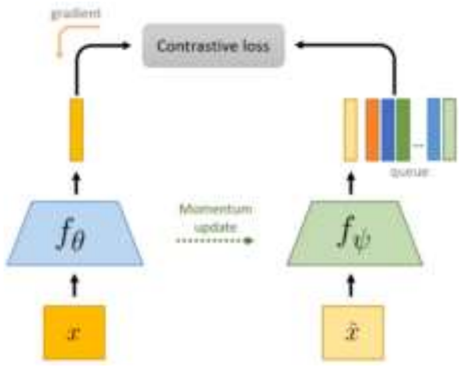


- fΘ = Resnet50, hΘ = MLP layer
- Given a mini-batch containing ‘N’ images, an augmented image is generated for each sample, resulting in total of ‘2N’ images
- For every image, its augmented image is considered a positive sample and remaining images as negative samples (Mini-batch sampling strategy).
- InfoNCE Loss function is computed over positive samples

$$l_{i,j} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} 1_{[k\neq i]} \exp(sim(z_i, z_k)/\tau)},$$

-
- Large mini-batches required to ensure enough negative samples

Moco

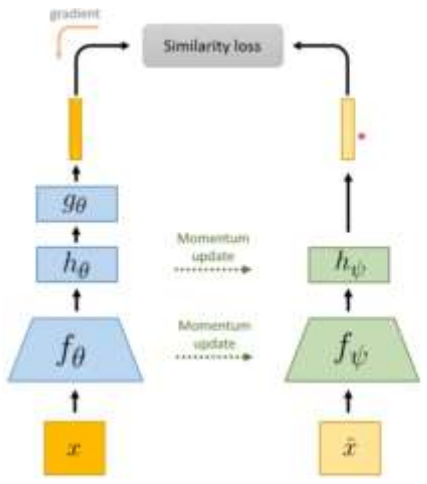


- MoCo builds upon SimCLR.
- Two instances of same model are used. One model is used to predict the features, while the other model is updated using exponential moving average (momentum encoder model)
- The features generated by the momentum encoder model are stored in a feature queue as negative examples. These negative examples do not undergo gradient updates during backpropagation
- This avoids the problem of mini-batches

Self-distillation methods

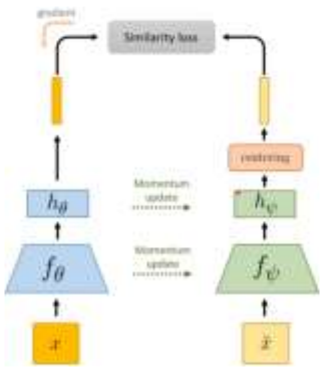
Self-distillation methods use Siamese network setup to learn features from data directly. It avoids use of negative samples

BYOL



- Teacher-student setup, where student is made to learn teacher's features
- The Teacher is updated using EMA of the student.
- MSE loss function

DINO

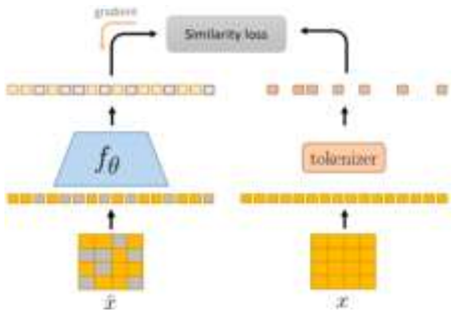


- Removed prediction head. Post-processing of teacher outputs to avoid feature collapse (if we’re to predict one-hot encoded features, can cause problem -> alternative representation)
- Introduces ViT (Vision transformers) as feature backbone
- Cross Entropy loss function

MIM (Masked Image models)

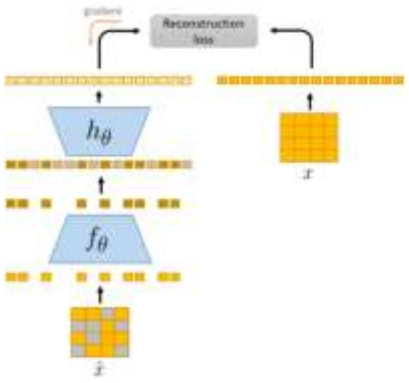
MIM leverages co-occurrences of patches as supervision signals. Objective is to make model robust to input noise.

BEiT



- Mimics practices from NLP domain (BERT)
- Some patches of image are masked and model (ViT) has to predict discrete visual token (~features) for entire image (including the masked ones)
- Pretrained tokenizer (DALL-E) is used to generate reference tokens
- Cross entropy loss function over masked tokens

MAE



- Simplified MIM pipeline without tokenizer, data augmentation
- Encoder operates only on visible patches
- Light weight Decoder (removed after pre-training) predicts entire image, based on encoded features. Predicted image is compared with the original “unmasked” image
- Can handle aggressive masking (up to 75%)
- Shines when tuned on downstream task
- Swin Transformer, ViT are all pretrained using self-supervised learning

CL models tend to be data hungry and vulnerable to overfitting issues, whereas MIM encounter data-filling challenges and exhibit inferior data scaling capabilities when compared to contrastive models.

Other approaches

GAN

- Generative Adversarial Networks are class of unsupervised learning methods, that generate new data samples based on the learned distribution of the training data
- SS-GAN (Self-supervised GAN) incorporates rotation invariance into GAN’s prediction process by integrating rotation prediction task during training

$$L_G = -V(G, D) - \alpha E_{x \sim p_G} E_{r \sim R}[\log Q_D(R = r|x^r)],$$
$$L_D = V(G, D) - \beta E_{x \sim p_{data}} E_{r \sim R}[\log Q_D(R = r|x^r)],$$

-
- Image reference (A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends)
- Input images are rotated by ‘r’ degrees and the Q (R =r | x^r) is discriminator predicting the rotation, based off the input image. By integrating it into loss function, model becomes more robust

Multi-Modality learning

- Text associated with an article used as supervision signal for images – Topic modelling

- CLIP leverages CL-style pre-training task to predict the correspondence between captions and images. Benefiting from the CL paradigm, CLIP is capable of training models from scratch on an extensive dataset comprising 400 million image-text pairs collected from the internet.

Birds Eye View Segmentation

- BEV (Bird's Eye View) semantic maps are very useful, but difficult to annotate. Perfect recipe for Self-supervised Learning. SkyEye is a self-supervised approach for generating a BEV semantic map using a single monocular image from the frontal view (FV). The model leverages the easily available FV semantic annotations of video sequences.

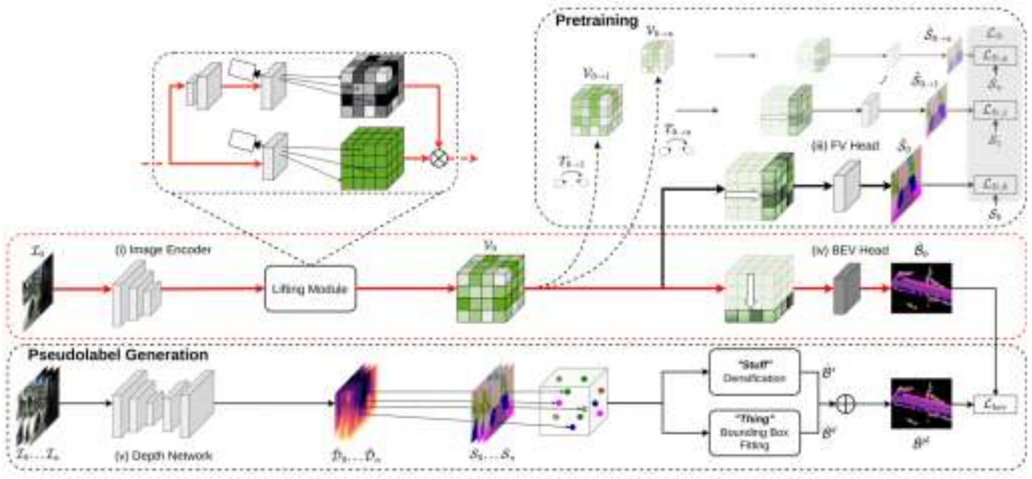
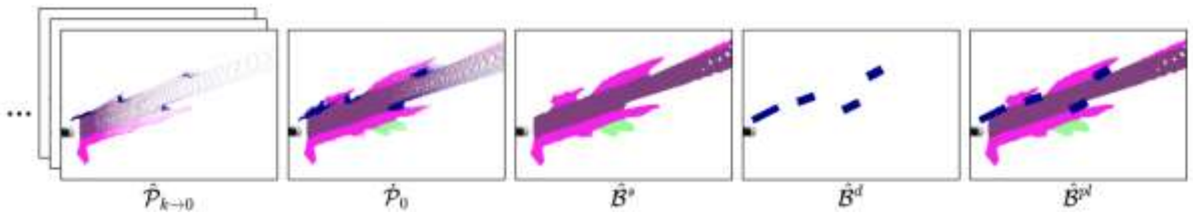


Image reference

- Core idea is to generate a 3D voxel grid that serves as a joint feature representation for both FV and BEV segmentation tasks, allowing us to use FV semantic labels to supervise BEV semantic learning.
- Sky Eye consists of 5 major components:
  - A Image encoder to generate 2D image features
  - A lifting module to generate the 3D voxel grid using a learned depth distribution
  - FV semantic head to generate the FV semantic predictions for implicit supervision
  - A BEV semantic head to generate the BEV semantic map
  - An independent self-supervised depth network to generate the BEV pseudolabels.
- **Implicit Supervision**
  - Given a Frontal View (FV) image at time t0, the goal is to predict the Front View Semantic maps for timesteps t0-tn.
  - First, a pre-trained EfficientDet-D3 encoder is used to generate feature maps at 4 different scales, which are combined using multi-scale fusion (reference : EfficientPS).
  - Lifting module lifts the 2D features to 3D voxel grid (V0)
  - The 3D voxel grid goes through perspective transform using camera intrinsics and processed along the depth dimension to generate FV semantic map (S0)
  - The Voxel grid (V0) is used to generate future voxel grids (V1-Vn) using vehicle pose at those timestamps. The future FV semantic maps (S1-Sn) are generated from the corresponding Voxel grids
  - FV Loss function is the weighted Cross Entropy b/w the predicted Semantic maps and actual FV semantic labels. w = step based linear decay from current timestamp to future timestamp

$$\mathcal{L}_{fv} = \sum_{i=0}^{n} L_{fv,i} = \sum_{i=0}^{n} w_i CE(\hat{\mathcal{S}}_{0\rightarrow i}, \mathcal{S}_i),$$

- **Explicit Supervision**
  - An independent Depth Estimation network is trained via Self-supervision on the same dataset (reference: [Digging into SS Monocular depth estimation])
  - Sparse semantic point clouds (P0-Pn) are generated using the predicted depth map , FV semantic label, camera intrinsics for t0-tn. NOTE: The point clouds are projected to BEV for each timestep
  - The future sparse point clouds are projected backwards in time using vehicle pose and accumulated to get a denser point cloud at time t0
  - A separate module densifies the point cloud and generates dense semantic labels for static objects (Bs)
  - The non-static points from previous step go through DBSCAN clustering + RANSAC algorithm to get BEV of bounding boxes for dynamic objects (Bd)
  - Bd are overlaid on top of Bs to get the **Bpl (**BEV pseudo label)



  - The 3D voxel grid from 1st step is projected height-wise, then passed through a BEV segmentation head to get the BEV semantic predictions (**B**)

$$\mathcal{L}_{bev} = CE(\hat{\mathcal{B}}^{pl}, \hat{\mathcal{B}}).$$

- The total loss = L_fv + L_bev
- Evaluated on KITTI 360 and Waymo 360 datasets and performs on par, with fully supervised models

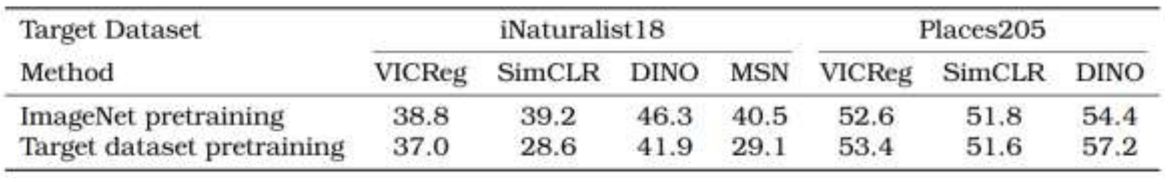
Table 1. Evaluation of BEV semantic mapping on the KITTI-360 dataset. All metrics are reported in [%].

Method	BEV GT	Road	Sidewalk	Building	Terrain	Person	2-Wheeler	Car	Truck	mIoU
IPM [25]	✗	53.03	24.90	15.19	32.31	0.20	0.36	11.59	1.90	17.44
TIIM [33]	✓	63.08	28.66	13.70	25.94	0.56	**6.45**	33.31	8.52	22.53
VED [24]	✓	65.97	35.41	37.28	34.34	0.13	0.07	23.83	8.89	25.74
VPN [30]	✓	69.90	34.31	33.65	40.17	0.56	2.26	27.76	6.10	26.84
PON [32]	✓	67.98	31.13	29.81	34.28	2.28	2.16	37.99	8.10	26.72
PoBEV [6]	✓	70.14	35.23	34.68	40.72	2.85	5.63	**39.77**	**14.38**	30.42
SkyEye (Ours)	✗	**71.39**	**37.62**	**37.48**	**44.38**	**4.73**	4.72	32.73	10.84	**30.49**

- Home page: http://skyeye.cs.uni-freiburg.de/
- Code: https://github.com/robot-learning-freiburg/SkyEye

Pointers for using Self-Supervised learning

- Data Augmentation (DA) is very important in the context of SSL. The type of DA to be used, depends on the downstream task and the targeted dataset. For example, simpler augmentations such as Color jitter, crops improve performance on classification tasks, while not so much in dense prediction tasks like Object detection / Semantic segmentation. Multi-crop, Mixup, Cutmix are more complex DA strategies used, but this increases pre-processing time considerably
- Selection of Data Augmentation methods is more important for contrastive learning methods (SimCLR, MocoVx variants) than Masked Image modelling methods.
- SSL performs best when you have limited labelled data, and its efficiency decreases when percentage of labelled data increases. Eg: If you have a labelled dataset of 1M samples, it might be more preferable to start with Supervised learning methods, compared to SSL methods.
- SSL performs poorly, when trained on unbalanced datasets. One of SOTA techniques introduce a regularisation term to improve performance
- CL (Contrastive Learning) methods have shown better performance, when trained on specific dataset, but generalize poorly, when evaluated on other datasets. (i.e) CL methods are more prone to overfitting, compared to MIM methods.
- Whenever possible, make use of "multi-modality" in the dataset. Eg: CLIP (Contrastive Language Image Pretraining) is a CL method, trained in self-supervised manner on image-caption pairs. When given an image, the model can describe the context.
- Barriers to Self-Supervised learning
    - **High computational cost**. As SSL operates without any manual labels, performance is related to size of dataset. This requires considerable compute capabilities.
    - Compared to Supervised learning, SSL methods are less standard. More transparency is required on SSL methods, models, performance evaluation on downstream tasks, fair comparison methods etc.
    - Lack of common taxonomy / vocabulary. SSL cookbook by FAIR is an effort to standardise the above concerns, and is highly recommended for people looking to venture into using SSL approaches
- Data quality is more important for SSL than supervised methods. Most methods developed so far involve pretraining on curated datasets like the ImageNet-1k, followed by limited supervised training on downstream task + downstream dataset. Recent research is exploring the effect of pretraining on the downstream dataset directly.
-



Target Dataset	iNaturalist18				Places205		
Method	VICReg	SimCLR	DINO	MSN	VICReg	SimCLR	DINO
ImageNet pretraining	38.8	39.2	46.3	40.5	52.6	51.8	54.4
Target dataset pretraining	37.0	28.6	41.9	29.1	53.4	51.6	57.2

Image reference

- In the above table, SimCLR and MSN approaches perform poorly on iNaturalist18 dataset, while other methods perform equally well. This shows that certain methods are more sensitive to the pretraining dataset than others
- **Role of projector**
    - Projector / Predictor is very important for SSL

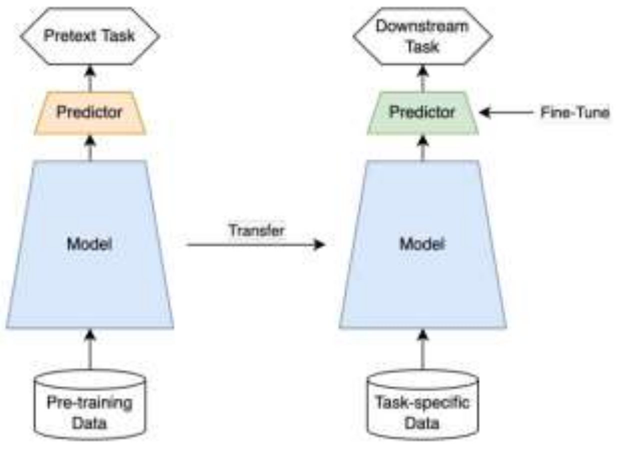


Image reference

  - With a typical SSL approach, only the encoder model is carried forward to downstream tasks and the Predictor (often 2-3 MLP layers) are fine tuned on target dataset.
  - Presence / Absence of projector, projector output size, the layer at which model is trimmed (referred to as "probing") greatly impacts the performances
- **Evaluation strategy**
  - SSL approaches are primarily trained for classification in "unlabelled" dataset and evaluated on downstream tasks. Following methods are commonly used to evaluate performance of different methods
  - **KNN**
    - Compute features from frozen encoder for all images in training dataset
    - For each evaluation sample, 'k' nearest neighbours are found based on distance b/w features. Majority vote of the neighbours is used as predicted class
    - Simplest method, fast to deploy without much hyperparameters
  - **Linear probing**
    - Single Linear layer on top of frozen encoder, optimized on labeled dataset
    - Since the capabilities of the layer itself is low, accuracy mainly depends on discriminative capability of the underlying model
  - **MLP probing**
    - 2 /3 Layer MLP network, on top of frozen encoder, which can adapt to more non-linear representations than a simple Linear network
    - Possibility of overfitting
  - **Fine-tuning**
    - The entire network is retrained on the downstream task, which makes it the most computationally

Vision Transformers (ViTs) have been preferred over CNNs for SSL in dense prediction tasks such as Object Detection and Semantic Segmentation. CNNs can achieve competitive performance, if augmentation, parameters are carefully tuned. MIM approaches have shown better performance, due to simpler architecture with fewer hyperparameters. The inherent masking + prediction nature enables the model to learn local features, which are more robust to input noise.

Challenges in using SSL for AD

- All current approaches are designed, verified using ImageNet (carefully curated object-centric dataset) whereas AD datasets are very different, mostly scene-centric with long tail class distribution (sky and road classes occupy large % of pixels)
- Add ImageNet sample image + BDD100K sample image
- AD datasets are comparatively boring with lot of repetitive scenes
- Important because performance of pretext tasks is highly dependent on the data properties

Unsupervised learning: Think hard about the model, use whatever data fits

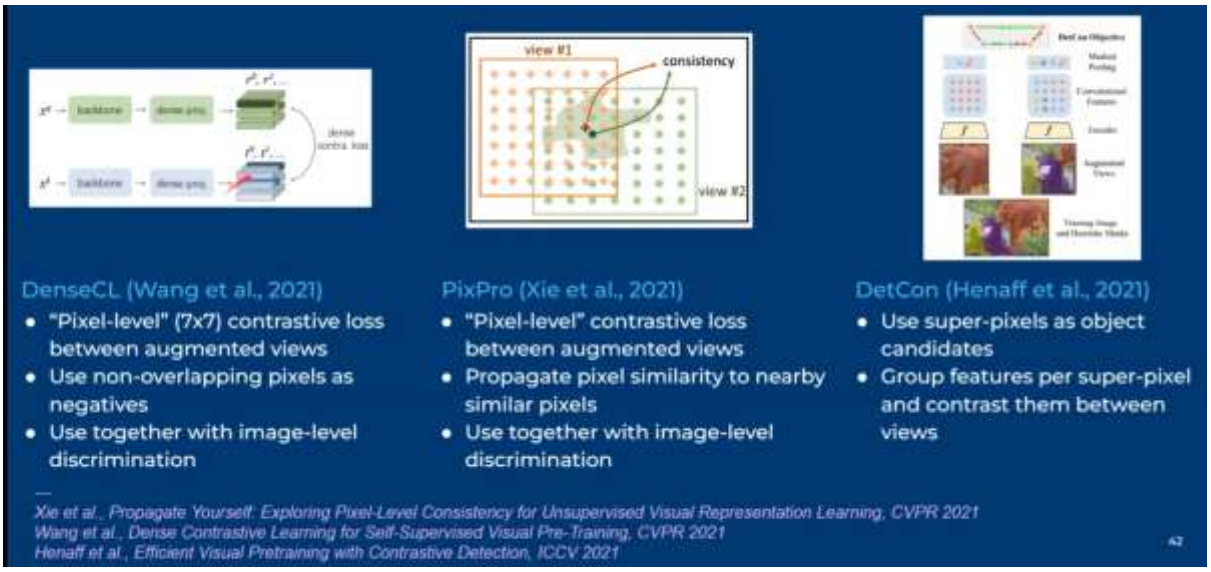
Self-supervised learning: Think hard about the data, use whatever model fits

Pretraining on uncurated dataset

- Divide and contrast: Self supervised learning from uncurated dataset, SEER
- Pretrained on Flickr (100M), Instagram (1B), google images (300M) dataset
- Perform close to supervised learning model (R50 = 76.5)

Beyond one object per image assumption

- [Dense Contrastive Learning for Self-Supervised Visual Pre-Training]
- [Propagate Yourself: Exploring Pixel-Level Consistency for Unsupervised Visual Representation Learning]
- [DetCon - Efficient Visual Pretraining with Contrastive Detection]


backbone
dense proj
dense contra. loss
view #1
consistency
view #2

DenseCL (Wang et al., 2021)
- "Pixel-level" (7x7) contrastive loss between augmented views
- Use non-overlapping pixels as negatives
- Use together with image-level discrimination

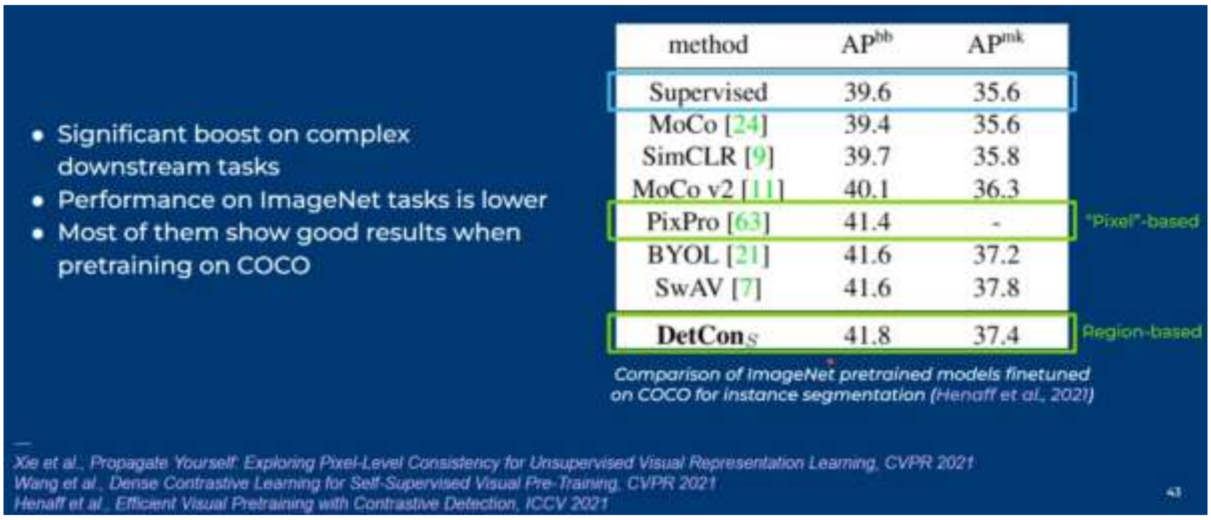
PixPro (Xie et al., 2021)
- "Pixel-level" contrastive loss between augmented views
- Propagate pixel similarity to nearby similar pixels
- Use together with image-level discrimination

DetCon (Henaff et al., 2021)
- Use super-pixels as object candidates
- Group features per super-pixel and contrast them between views

Xie et al., Propagate Yourself: Exploring Pixel-Level Consistency for Unsupervised Visual Representation Learning, CVPR 2021
Wang et al., Dense Contrastive Learning for Self-Supervised Visual Pre-Training, CVPR 2021
Henaff et al., Efficient Visual Pretraining with Contrastive Detection, ICCV 2021

42

- Most show improvements when pretrained on COCO dataset
- Performance on ImageNet is lower



- Significant boost on complex downstream tasks
- Performance on ImageNet tasks is lower
- Most of them show good results when pretraining on COCO

method	AP^{bb}	AP^{mk}	
Supervised	39.6	35.6	
MoCo [24]	39.4	35.6	
SimCLR [9]	39.7	35.8	
MoCo v2 [11]	40.1	36.3	
PixPro [63]	41.4	-	"Pixel"-based
BYOL [21]	41.6	37.2	
SwAV [7]	41.6	37.8	
DetCon$_S$	41.8	37.4	Region-based

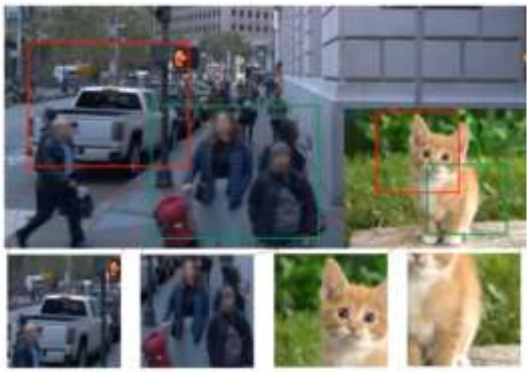
Comparison of ImageNet pretrained models finetuned on COCO for instance segmentation (Henaff et al., 2021)

Xie et al., Propagate Yourself: Exploring Pixel-Level Consistency for Unsupervised Visual Representation Learning, CVPR 2021
Wang et al., Dense Contrastive Learning for Self-Supervised Visual Pre-Training, CVPR 2021
Henaff et al., Efficient Visual Pretraining with Contrastive Detection, ICCV 2021

43

Pretrain on AD dataset

- Random crops on AD datasets not trivial, compared to ImageNet
- High resolution images and contain multiple object instances
- Different random crops could have different semantic meanings. So, classifying as positive / negative examples is difficult
- MultiSiam (Self supervised multi instance learning for AD)



Exploiting other AD sensors and supervisory signals for SSL

- Annotation free supervision signals in form of synchronized sensors
- Projecting Lidar points to camera frame (pre-calibrated) to get sparse pixel-depth mapping (3D point – 2D pixel correspondences)
- Time dimension can be used as supervisory signal

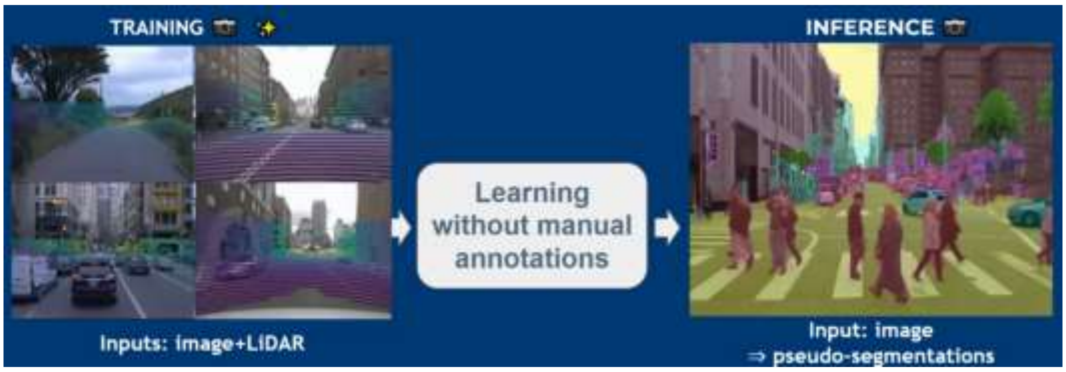
Image based SSL

- Exploiting videos to learn better image representations – FlowE
- Add architecture
    - Single frame approach
        - Take two consecutive frames (input: affine transforms of single image)
        - Separate Teacher-student approach
        - The transform b/w two views are applied on output of teacher network
        - Pixel wise BYOL loss function b/w student and teacher predicted features
        - Pixel correspondences: know transform b/w views
    - Video approach
        - Use 2 consecutive frames (~1sec apart)

        - Pixel correspondences, calculated using optical flow b/w frames
        - Affine transforms can be applied on input frames, apart from optical flow
    - Affine transforms + optical flow (Data augmentation) -> 25% increase in mIoU (semantic segmentation) and mAP (mean Average precision)
- Apart from pre-training task, downstream task can also be self-supervised, unsupervised object detection or semantic segmentation
- Use pretrained models (DINO / VIT) features representations (separate background) to pretrain models
    - LOST Unsupervised Object localization with pretrained SSL features
    - STEGO – Unsupervised segmentation using pretrained SSL features
- In AD, we can also use the synchronized Lidar point cloud + camera images for unsupervised learning of OD + sem.seg

Drive & Segment

- Drive&Segment: Unsupervised Semantic Segmentation of Urban Scenes via Cross-modal Distillation



- Training involves learning from synchronized lidar-image data
- During inference, we predict pseudo-segmentations, meaning, the predicted classes don't have explicit meaning, but are grouped together. Cars in blue, people in red etc

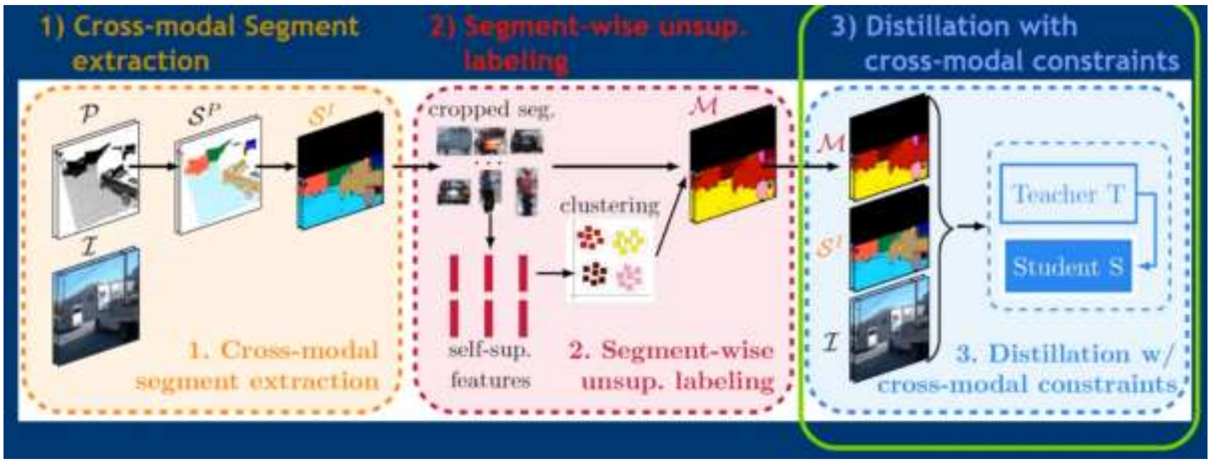

1) Cross-modal Segment extraction
2) Segment-wise unsup. labeling
3) Distillation with cross-modal constraints
$\mathcal{P}$
$\mathcal{S}^P$
$\mathcal{S}^I$
$\mathcal{I}$
cropped seg.
$\mathcal{M}$
clustering
self-sup. features
Teacher T
Student S
1. Cross-modal segment extraction
2. Segment-wise unsup. labeling
3. Distillation w/ cross-modal constraints

Image reference

- 1st step
    - cluster them using heuristic approaches
        - remove ground plane
        - k-means clustering using 3D coordinates

    - project segmented lidar points on images
- 2nd step:
    - Crop each segment from segmented images
    - Pass through SSL pretrained models to extract features
    - Features -> k means clustering to get clustered labels
    - Creates pseudo segmentation maps
- 3rd step:
    - Learn sem.seg from outputs of both 1st and 3nd step (lidar projected segmented points, pseudo segmentation maps)
    - Teacher student model
- Architectures
    - R18 + FPN (RN)
    - Segmenter + ViT-S backbone (SG)

Unsupervised OD with Lidar clues

- Unsupervised Object Detection with LiDAR Clues

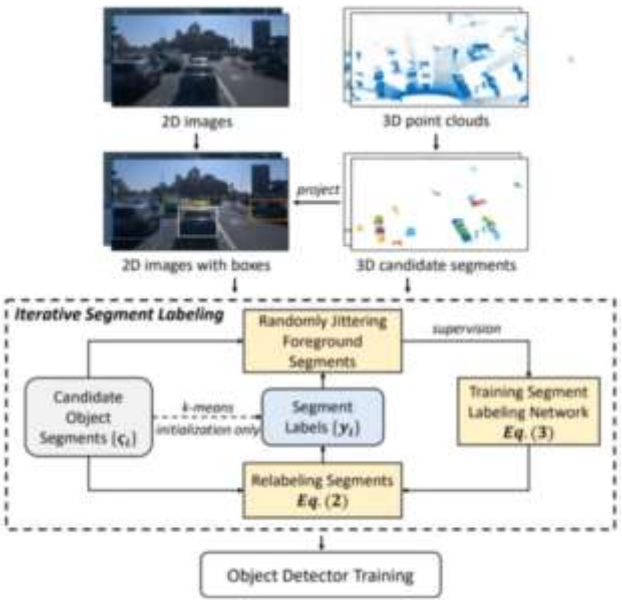

2D images
3D point clouds
project
2D images with boxes
3D candidate segments
Iterative Segment Labeling
Randomly Jittering Foreground Segments
supervision
Candidate Object Segments {c_i}
k-means initialization only
Segment Labels {y_i}
Training Segment Labeling Network Eq. (3)
Relabeling Segments Eq. (2)
Object Detector Training

Image reference

- Use Lidar + SSL features to generate pseudo annotations for Unsupervised OD (separate labelling network)

Lidar based SSL

- Lidar data, useful for 3D perception tasks
- Labelling Lidar data is hard, expensive
- Also, unlike images where we have ImageNet (and other datasets), we don’t have pretrained backbones for lidar data. So, generally supervised training on downstream task directly

Lidar-only SSL

DepthContrast

- Depth Contrast: Self-Supervised Pretraining on 3DPM Images for Mining Material Classification
- Scene level contrastive SSL
- Point network and voxel network are jointly pretrained on original 3D point cloud and Augmented views

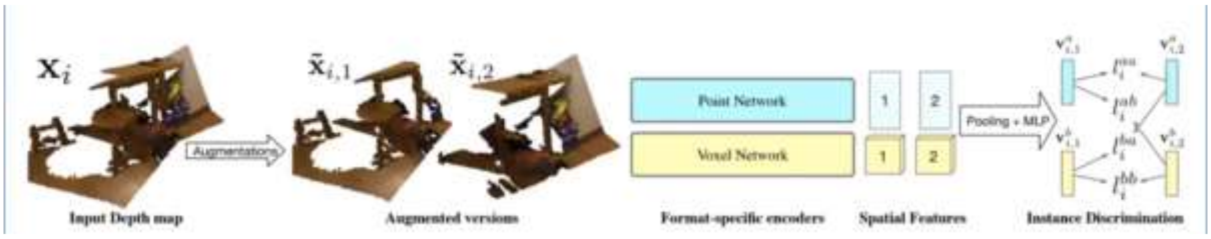


Image reference

- Reduces need for labelled datasets (for same performance) by 50%

PointContrast

- PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding

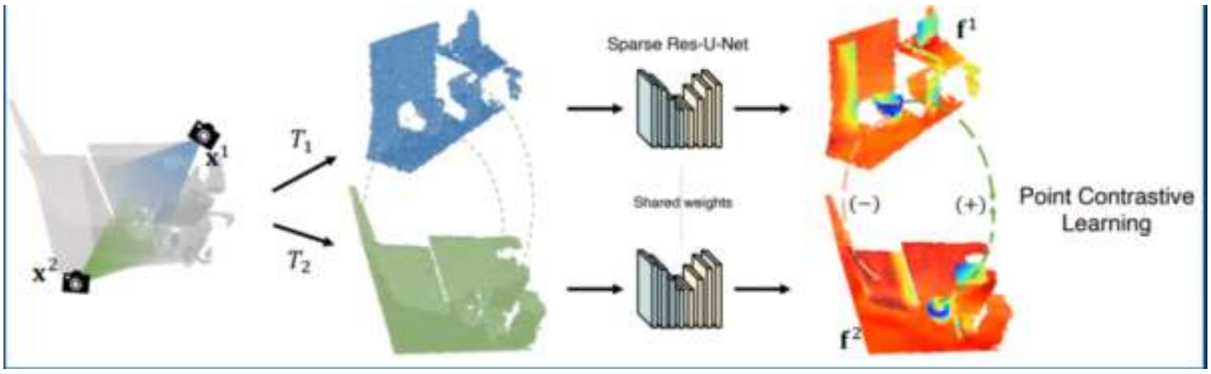


Image reference

- Point wise contrastive loss b/w 2 3D views of the same scene

ProposalContrast

- ProposalContrast: Unsupervised Pre-training for LiDAR-based 3D Object Detection
- Extract point cloud proposals
- Remove ground plane, cluster using Farthest Sampling method, generate proposals from 2 views using encoder network (shared b/w two views)
- Projector network to reduce dimension and contrastive loss at end

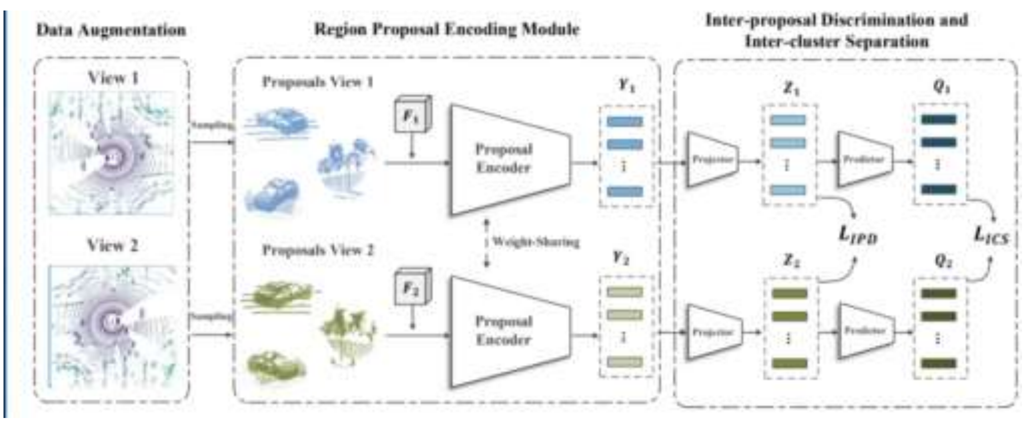


Image reference

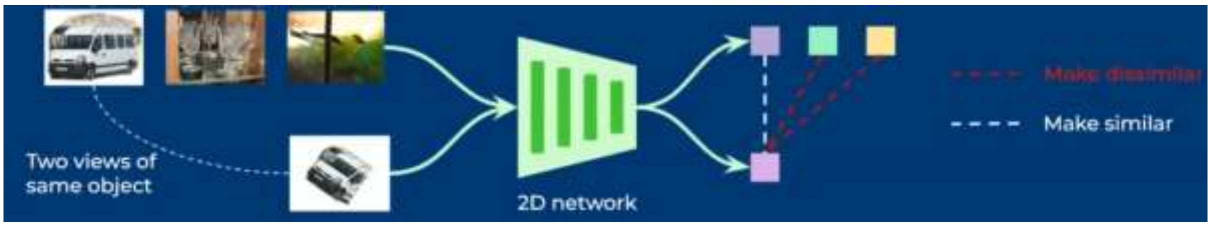
- Better representations for 3D OD

Lidar-Image based SSL

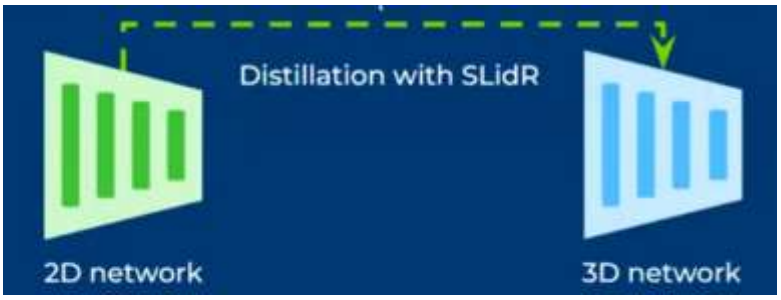
- To help in 3D OD and Semantic segmentation, make use of synchronized Lidar-Image data

Image to Lidar Distillation

- Image-to-Lidar Self-Supervised Distillation for Autonomous Driving Data
- 2D network pretrained like MoCO, SimCLR using SSL (e.g. on ImageNet)



- Knowledge Distillation from 2D network to 3D network using SLidR



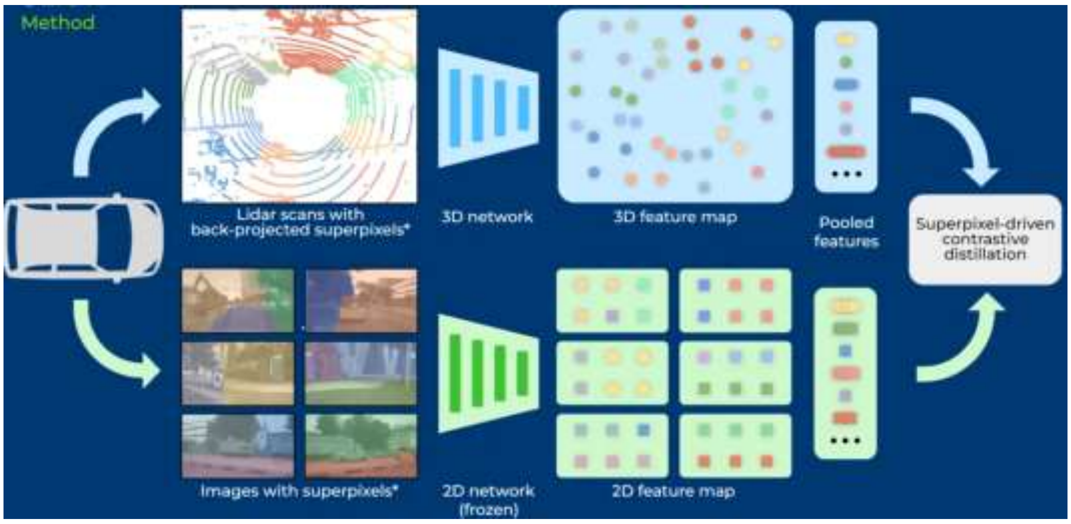


Image reference

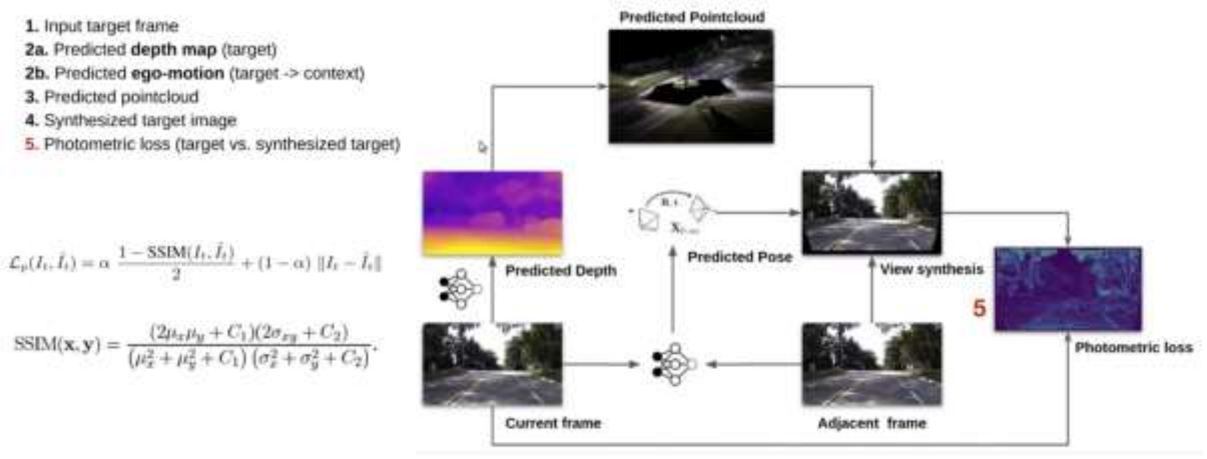
- Pixels are grouped together as superpixels
- Lidar annotation is more costly, and SSL methods on Lidar is gaining popularity

Monocular Depth Estimation SSL

- Videos are not raw data -they contain geometry information
- Geometric principles can be used as proxy supervision for certain tasks
    - Depth, ego-motion, key points, optical flow, scene flow

Methodology

- Two networks – one to predict depth and another to predict the pose transformation
- Generate point cloud by lifting the predicted depth map and combined with predicted pose, predict the original frame
- Photometric loss b/w the original frame and the synthesized frame

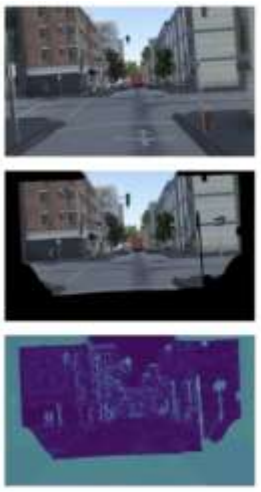
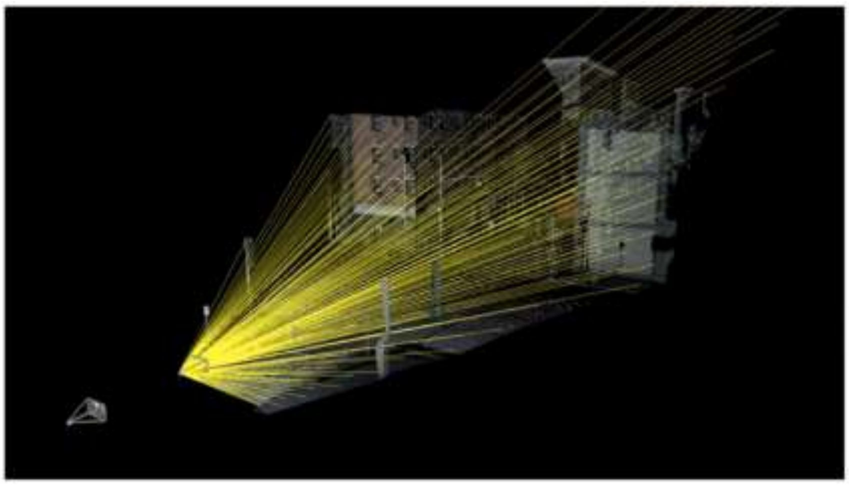

1. Input target frame
2a. Predicted depth map (target)
2b. Predicted ego-motion (target -> context)
3. Predicted pointcloud
4. Synthesized target image
5. Photometric loss (target vs. synthesized target)
Predicted Pointcloud
Predicted Depth
Predicted Pose
View synthesis
5
Photometric loss
Current frame
Adjacent frame

$$\mathcal{L}_p(I_t, \hat{I}_t) = \alpha \frac{1 - \mathrm{SSIM}(I_t, \hat{I}_t)}{2} + (1-\alpha)\, \|I_t - \hat{I}_t\|$$

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}.$$

Image reference

- 3d point cloud is obtained by lifting 2d features using depth information
- Project the 3d point cloud onto new frame (t+1). Some areas will be missing, due to non-overlap and occlusion

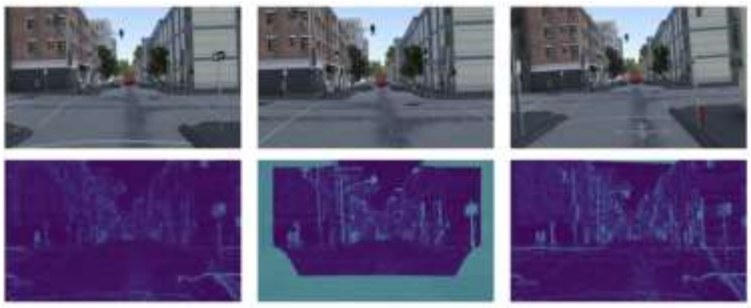
- Digging Into Self-Supervised Monocular Depth Estimation

Challenges

- Constant brightness
    - Objects should maintain appearance across frames
- Low Textured information
    - Ambiguities in photometric loss computation
- Static world
    - Dynamic objects will violate depth + ego-motion warping
- Occlusions
    - Viewpoint changes will create areas without matching

Improvements

- **Feature-metric loss for SSL of Depth estimation and egomotion**
    - Learned feature representation is used for warping (auto encoders)
    - Addition of encoder to generate features and warping the features, instead of the original image, leads to robustness to brightness changes
    - Introduces additional feature network (encoder-decoder)
- **Minimum reprojection loss (digging into Monocular depth estimation)**
    - MonoDepth2
    - Instead of just using 1 image for reprojection, use 2, one earlier and one later in timestamp. Reduces occlusion areas. Mostly in videos, if one area is occluded in previous frame, it will not be in later frame and vice versa.



- **Automasking**
    - Masking objects that have no relative motion
    - Apply photometric loss for unwarped image too

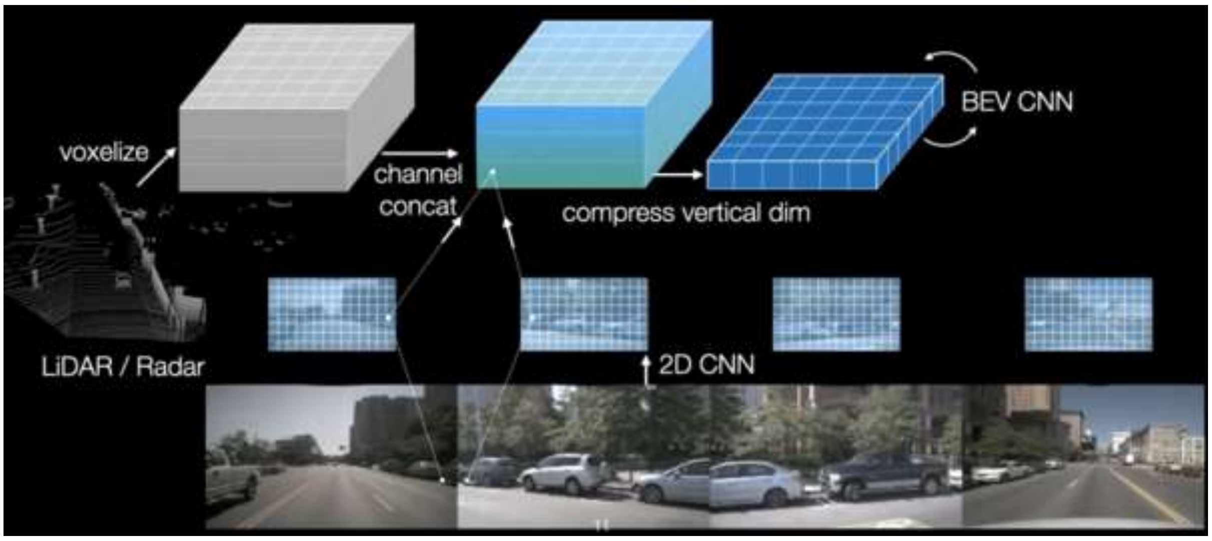
- Infinite depth problem (Semantically guided representation learning for Monocular depth)
    - Objects wit no relative motion can be modelled as infinitely faraway
    - Dataset rebiasing
        - Train a model with all available data
        - Remove frames that exhibit this infinite depth issue
        - Retrain / finetune with the filtered data

Extensions

- Multi camera configurations
    - Can we learn from multiple moving cameras?
    - Full Surround Monodepth
        - DDADs dataset, 6 cameras, Spatio temporal loss function
        - Photometric loss is masked out in areas with invalid information
- Multi-frame inference
    - Can we leverage camera motion at test time?
        - DepthFormer
        - Better feature representation, but more failures
- Modelling dynamic objects (optical, depth and scene flow without real world labels)
    - Can we estimate motion from surrounding objects?
        - Jointly learn depth and scene flow
        - Ill posed problem with 2 variables, 1 equation
        - Mixed batch learning
            - Have a simulation with same geometric prior as real world
            - Synthetic supervision + real world self-supervision
        - Optical flow is estimated from multiple frames using RAFT methodology
            - Incremental / residual optical flow estimation
            - Used as initialization for depth estimation and scene flow
- SSL self-calibration (Self supervised self-calibration from videos)
    - Can we learn from uncalibrated, distorted videos?
    - All above models assume, we have parameters, specific to the camera model – either pinhole / fisheye model with focal length, principal points etc
    - Unified camera model (closed form differentiable projection, unprojection operators), efficient
    - Parameters learned across scenes, instead of each sample wise

SSL for 3D Detection and Tracking

- BEVFormer is SOTA in 2022 in BEV Segmentation
- BEV models use 3D grid (10m x 100m x 100m with specific resolution), centered around vehicle frame origin.
- Fill volume with features. Image -> 2D conv -> lift to 3D -> Fill in 3D feature volume (IPM, project 3D point to 2D coordinate, bilinear sample). Simple, but effective method, can be extended to multiple cameras. The overlapping regions can be averaged / summed etc
- Similar feature volumes can be constructed from Lidar, radar sensors and concatenated with feature volume from camera sensors



- The feature volumes can be compressed vertically, which can be used as base for multiple heads (shallow networks with dedicated tasks)
- Recent trend is to use alternative methods to lift 2d features to BEV space (instead of the bilinear sampling). The accuracy of this “lifting” step, was assumed to determine the accuracy of the downstream task.
  - MLP
  - Depth distribution estimation
  - Transformer based models (BEVFormer)

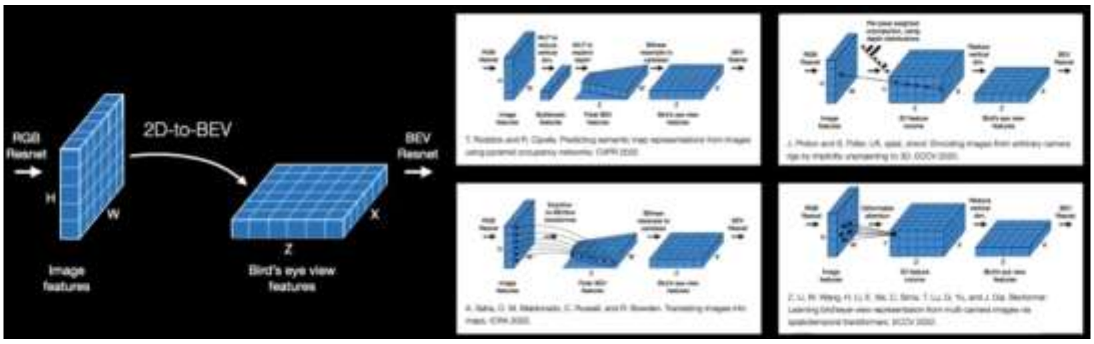


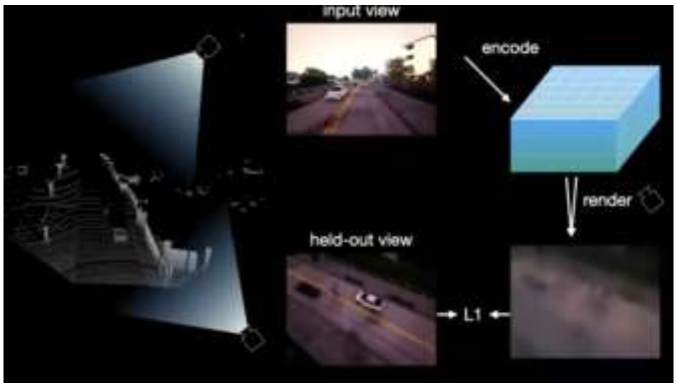
Image reference

- Lift, Splat, Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D
- But Simple-BEV paper illustrates that batch size and resolution have much bigger effect (best and worst models have <5 mIoU difference, when compared apples to apples)
- Camera + radar performs much better, compared to just camera
- nuScenes train metric = 69, val metric = 47. 2 possible reasons
  - Different data distributions (unlikely, as both are from same city, weather conditions)
  - Models are overfit to training dataset (more likely)

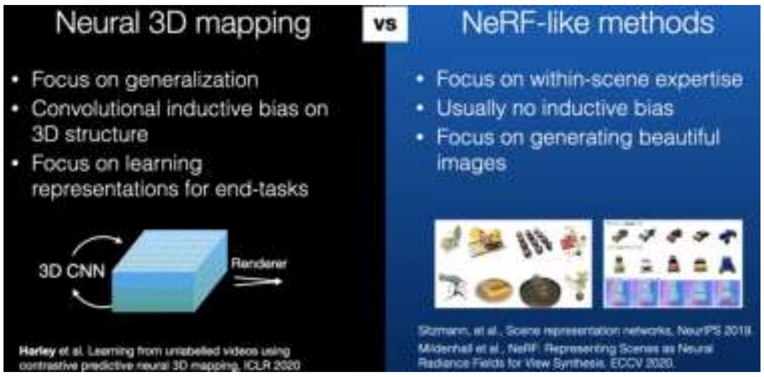
Free supervision from multiple views

- As we generate 3D feature volume, which represents the entire surrounding scene, we can use a renderer (NeRF) to generate views from the 3D volume

- Encode one of views (use 2D CNN + any lifting technique), generate a view using renderer and apply 1 loss b/w predicted view and held-out image (RGB regression method)



- Neural 3D mapping focusses on generating generalized scene representations, rather than synthesizing beautiful images (Learning from unlabelled videos using contrastive predictive neural 3d mapping)



Neural 3D mapping vs NeRF-like methods

- Focus on generalization
- Convolutional inductive bias on 3D structure
- Focus on learning representations for end-tasks

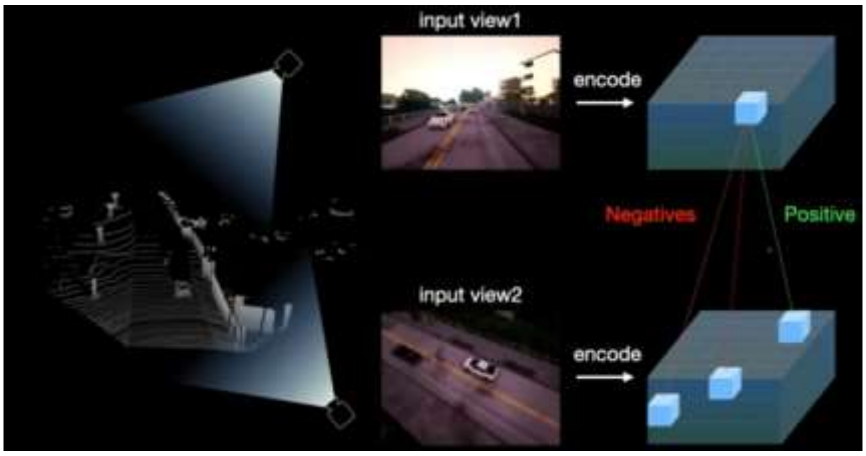
- Focus on within-scene expertise
- Usually no inductive bias
- Focus on generating beautiful images

3D CNN
Renderer

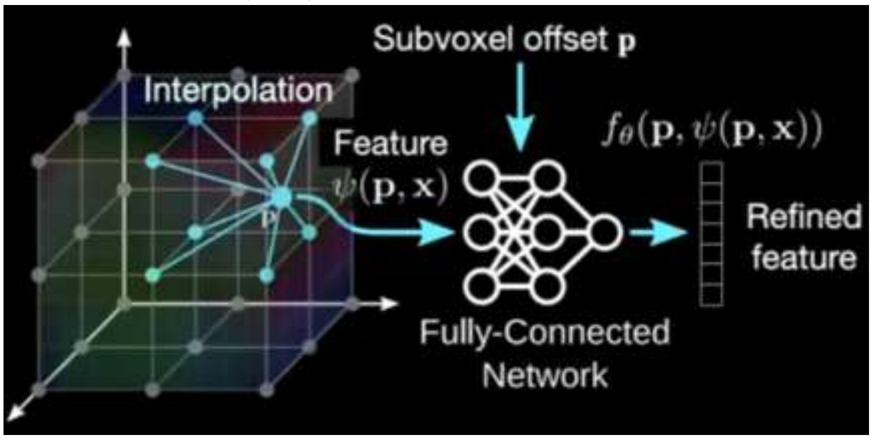
Harley et al. Learning from unlabelled videos using contrastive predictive neural 3D mapping. ICLR 2020

Sitzmann, et al., Scene representation networks. NeurIPS 2019.
Mildenhall et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV 2020.

- LEARNING FROM UNLABELLED VIDEOS USING CONTRASTIVE PREDICTIVE NEURAL 3D MAPPING
- Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations
- NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
- NeRFs also generate good features (few shot learning for semantic segmentation)
- Instead of RGB prediction, Feature prediction and contrastive learning is also useful and is found to be better than RGB prediction method



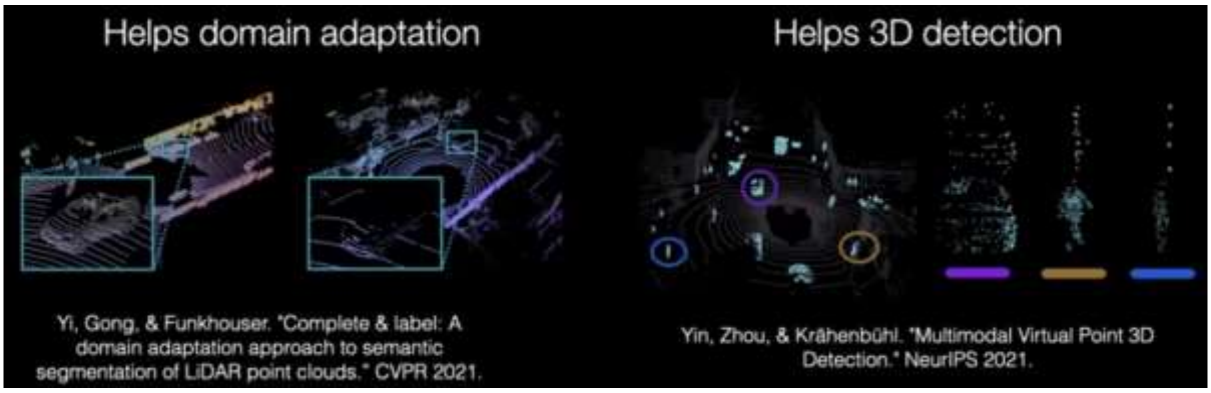
- Continuous 3D feature maps with implicit functions (resolution is very important with such 3D feature maps, contrastive loss is applied at sub-voxel level to ensure feature consistency) (**Convolutional occupancy networks**)


Subvoxel offset $\mathbf{p}$
Interpolation
Feature
$\psi(\mathbf{p}, \mathbf{x})$
$f_\theta(\mathbf{p}, \psi(\mathbf{p}, \mathbf{x}))$
$\mathbf{p}$
Refined feature
Fully-Connected Network

- CoCoNets Continuous Contrastive 3D scene representations
- Aiming for consistent features in invisible locations (either seen in 1 view or not seen in any view) is the reason for outperformance

Free supervision from completeness

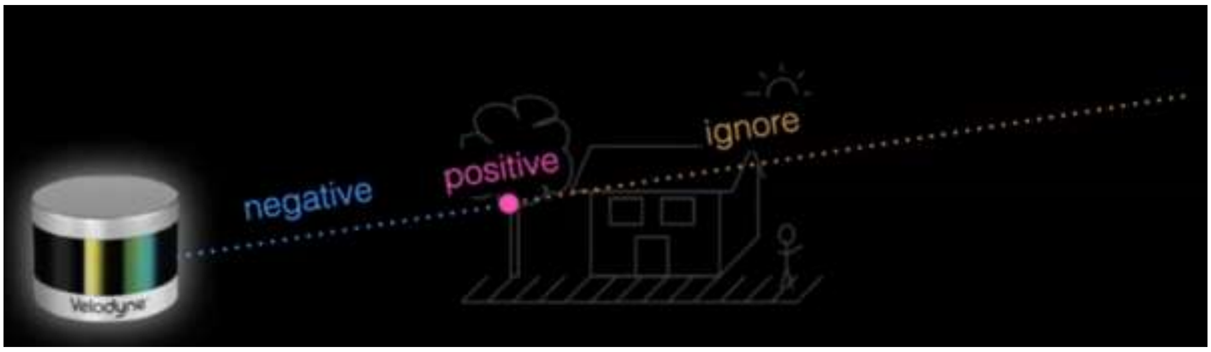
- Lidar data is sparser away from origin points. It makes detection difficult – more difficult to detect object with very few points vs object with >> points
- Completing the points helps improve detection

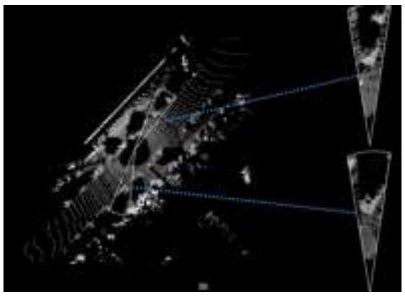
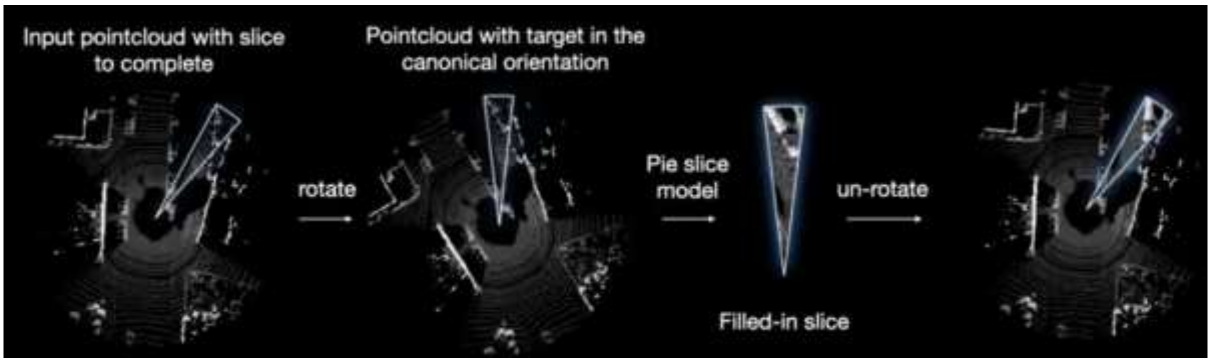

Helps domain adaptation
Helps 3D detection
Yi, Gong, & Funkhouser. "Complete & label: A domain adaptation approach to semantic segmentation of LiDAR point clouds." CVPR 2021.
Yin, Zhou, & Krähenbühl. "Multimodal Virtual Point 3D Detection." NeurIPS 2021.

- Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds

- Multimodal Virtual Point 3D Detection

- Domain adaptation: Create dense point clouds by aggregating multi-views from multiple timestamps. Problem: dynamic objects create artifacts, which have to be remove manually, using heuristics
- No need of multi-views (train sparser -> sparse, then test sparse -> dense)
- SG-NN Sparse Generative Neural Network
- Make use of geometry, all 3d point between ray end point and sensor can be considered negative sample and filled with appropriate label

- 

- Point cloud completion is not translation invariant, but its invariant to rotation about the vertical axis, (shaped like a pie slice)

- Model inference: in paint 1 slice at a time and combine together (single model across all slices)

Input pointcloud with slice to complete
rotate
Pointcloud with target in the canonical orientation
Pie slice model
Filled-in slice
un-rotate
- Model is trained on the dense point cloud. Filling in, pays off, even without multi-view

Free supervision from motion

- Unsupervised learning via reconstruction (auto encoder models)
- Till recently, papers were using simple 3D objects datasets, while 2011 ish era models used motion as cue for segmentation and tracking
- "Track, check and repeat – An EM method to unsupervised tracking", make use of simple heuristics to identify moving objects from sensor data (Lidar, Radar). Even though, they're sparse, use them to train 2D, 3D convs. train on this dataset, repeat, use ensembles etc
- Discovering objects that can move
- SAVi++ Towards Object centric learning from real world videos

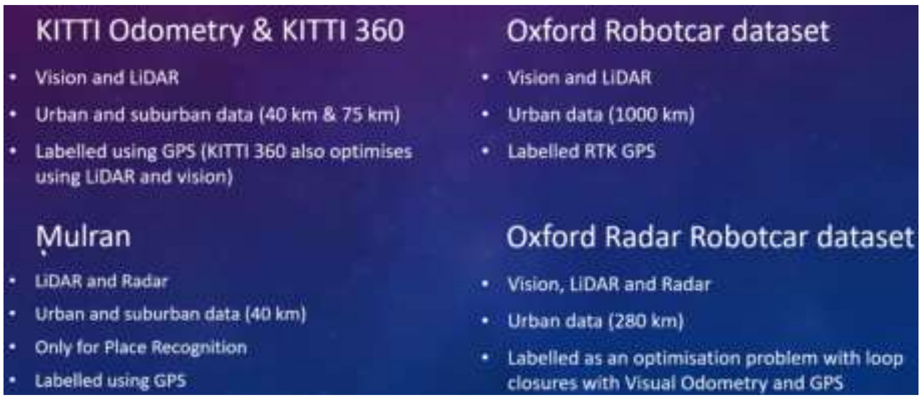
Self-supervised delivering better representations than supervised ones for Imagenet datasets

- Emerging properties in SS vision transformers
- Deep ViT features as Dense Visual descriptors (DINO)

But this is not yet achieved for AD datasets.

SSL for Odometry and Localization

- Odometry is just tf b/w two consecutive frames (small time / space difference)
- Localization (have some kind of database / map to store points or coordinates)
    - Topological – retrieve if current position is in map ~ Sample retrieval
    - Metric – regress the tf b/w current position to some reference point in map ~ registration



KITTI Odometry & KITTI 360

- Vision and LiDAR
- Urban and suburban data (40 km & 75 km)
- Labelled using GPS (KITTI 360 also optimises using LiDAR and vision)

Oxford Robotcar dataset

- Vision and LiDAR
- Urban data (1000 km)
- Labelled RTK GPS

Mulran

- LiDAR and Radar
- Urban and suburban data (40 km)
- Only for Place Recognition
- Labelled using GPS

Oxford Radar Robotcar dataset

- Vision, LiDAR and Radar
- Urban data (280 km)
- Labelled as an optimisation problem with loop closures with Visual Odometry and GPS

Odometry by segmentation

- With Radar data as primary input, visual odometry is used as supervision signal (Fast radar segmentation)
- Masking by moving: learning distraction free radar odometry from pose information
- Under the radar: Learning to predict robust key points for odometry information and metric localization in radar (using point clouds as supervisory signals)

- **Perceiver by DeepMind**

SSL applications

Object Detection

Self-supervised Pretraining: Pretraining a model on self-supervised tasks like predicting object rotations or context-based pretext tasks helps the model learn useful features. These features, when transferred to an object detection task, provide a strong initialization for the model, enhancing its ability to detect objects accurately.

Semantic Segmentation

Spatial Context Understanding: Self-supervised learning can focus on predicting relationships between different parts of an image without explicit segmentation labels. This spatial context understanding, learned through self-supervised tasks, can significantly improve the segmentation model's performance by capturing better contextual information.

Videos

Self-supervised learning has widespread usage across video applications (Unsupervised learning of visual representations using video), including video representation learning and video retrieval. The central concept revolves around utilizing visual tracking as a self-supervised signal. Consequently, two patches connected by a track are expected to possess similar visual representations, as they likely correspond to the same object or belong to the same object part. Several forms of temporal information in videos can be exploited:

Frame Order

- Make use of raw spatiotemporal signals, available in videos
- The frames in video are temporally shuffled, and provided as input to the model with objective of predicting the correct order.
- Alternatively, objective in "odd-one-out learning" [225] is to identify the unrelated or odd element within a set of related elements.

Video Playback direction

- (Learning and using the arrow of time), involves discerning the arrow of time to determine if a video sequence progresses in the forward or backward direction

Video Playback speed

- Speednet (Speednet: Learning the speediness in videos) focused on predicting the speeds of moving objects in videos, determining whether they moved faster or slower than the normal speed.

Temporal Coherence Prediction

- Application to Object Detection: Predicting temporal coherence in video frames encourages the model to understand object motion and dynamics. This can improve object detection in video sequences.

- Application to Instance Segmentation: Temporal coherence learning benefits instance segmentation in dynamic scenes, enabling the model to track and segment objects over time.

Depth Estimation

Relative Depth Prediction: Self-supervised learning tasks such as predicting the relative depth between pixels in an image help the model grasp the 3D structure of the scene. When transferred to a depth estimation task, this knowledge aids in more accurate and robust depth predictions.

Option1

- Assumptions
  - Stereo camera setup
  - Pose transformation between both left and right cameras is known.
- Methodology
  - Left camera frame is provided as input to a CNN model trained from scratch to predict, the corresponding depth map.
  - We reconstruct the left camera frame by combining the predicted depth map with right camera frame and known stereo geometry (referred to as inverse warping)
  - Per-pixel loss function b/w the reconstructed image and the original left camera image is be used to train the model
- Reference: Unsupervised cnn for single view depth estimation: Geometry to the rescue

Option2

- Methodology
  - Same as option1, but we try to predict the right camera image, instead of the left.
- Reference: Self-supervised learning for stereo matching with self-improving ability

Both above methods assume that pose transformation b/w the two camera views is known.

Option3

- Methodology
  - predict, from a temporal sequence of frames, the depth map with a learning model, and the successive camera pose transformations with another learning model.
  - Both models are trained together from end-to-end for making the novel view synthesis of the next frame.
  - However, the pose transformation estimation implies that the predicted depth map is defined up to a scale factor.
- Reference: Unsupervised learning of depth and ego-motion from video

Evaluation

- Self-supervised learning is evaluated by gauging the achieved performance on downstream tasks to ascertain the quality of the extracted features.
- However, this evaluation metric does not provide insights into what the network has specifically learned during self-supervised pre-training.
- To delve into the interpretability of self-supervised features, alternative evaluation metrics, such as network dissection (Network dissection: Quantifying interpretability of deep visual representations), can be employed.

The effectiveness of self-supervised pre-training models, whether on the same dataset or a different dataset, depends on various factors. Here are considerations for both scenarios:

Same Dataset Pre-training:

- Pros: Pre-training on the same dataset allows the model to learn specific features and representations relevant to the target domain (autonomous driving). It can capture the nuances and characteristics present in the data, potentially leading to better performance on the final task.

- Cons: There is a risk of overfitting to the training dataset, especially if it is not diverse enough. The model may not generalize well to new, unseen scenarios, limiting its adaptability.

Different Dataset Pre-training:

- Pros: Pre-training on a diverse dataset, even if not directly related to autonomous driving, can help the model learn generic features and representations. This may improve the model's ability to handle a broader range of scenarios and enhance its generalization to new environments.

- Cons: The features learned on a different dataset may not be optimally aligned with the characteristics of the autonomous driving task. There might be a need for additional fine-tuning on the specific autonomous driving dataset to achieve optimal performance.

In practice, a combination of both strategies, known as transfer learning, is often employed. Initial pre-training on a large, diverse dataset provides a solid foundation, followed by fine-tuning on the target dataset (autonomous driving) to adapt the model to specific requirements.

The choice between same dataset and different dataset pre-training depends on factors like dataset availability, diversity, and the desired balance between task-specific and general features for the autonomous vehicle application. Experimentation and careful evaluation are essential to determine the most effective approach for a given scenario.

References

- Self-supervised learning for autonomous vehicles perception: A conciliation between analytical and learning methods
- A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends
- Unsupervised Representation Learning by Predicting Image Rotations
- Efficient visual pretraining with contrastive detection
- Data-Efficient Image Recognition with Contrastive Predictive Coding
- Self-Supervised Learning - Self prediction and contrastive learning