Deep Learning-based localization of 3D indications from multiple view

with application in X-Ray Imaging

Kartik Viswanathan Ba-Khuong Dang Luca Oriani

SMART DataScience project
ENSAI

February 13, 2025

Overview

Deep Learning-based localization of 3D indications from multiple X-ray imaging

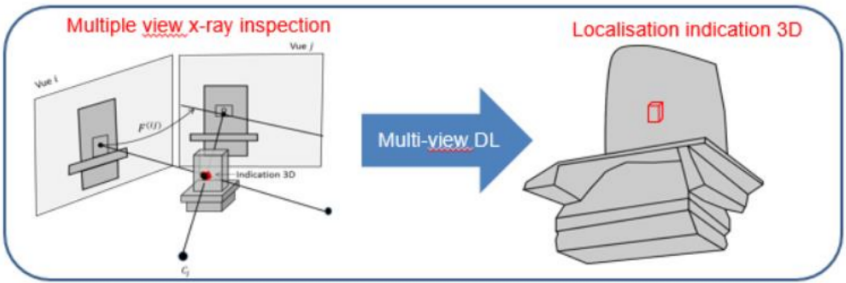


Figure: Visualization of X-ray reconstruction setup

State of the art

There are several approaches that are usually taken into consideration for what it concerns the 2D-to-3D lifting methods.

- DETR3D
- PETR
- Bird's Eye View (BEV)

State of the art

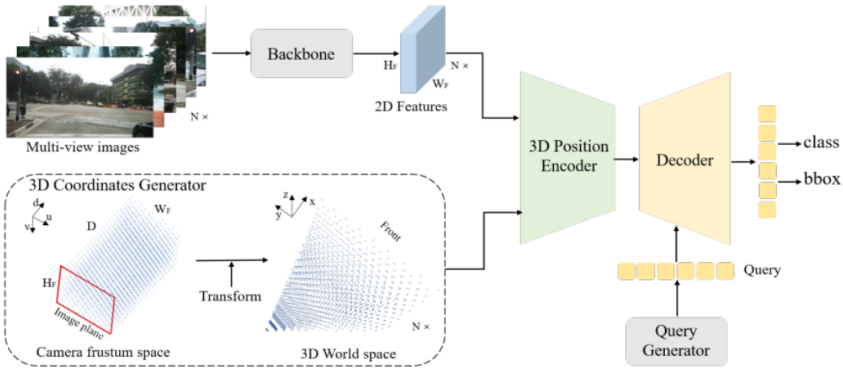


Figure: An illustration of PETR: Encoding 3D positional data directly into 2D image features to enhance efficiency and scalability.

Comparative Analysis of Methods

Method	Backbone	Resolution	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
DETR3D [41]	ResNet-50	1600 × 900	0.373	0.302	0.811	0.282	0.493	0.979	0.212
PETR [26]	ResNet-50	1408 × 512	0.403	0.339	0.748	0.273	0.539	0.907	0.203
PETRv2 [27]	ResNet-50	1600 × 640	0.494	0.398	0.690	0.273	0.467	0.424	0.195
MV2D-S‡	ResNet-50	1408 × 512	0.440	0.398	0.665	0.269	0.507	0.946	0.203
MV2D-T‡	ResNet-50	1600 × 640	**0.546**	**0.459**	**0.613**	**0.265**	**0.388**	**0.385**	**0.179**
FCOS3D† [39]	ResNet-101	1600 × 900	0.415	0.343	0.725	0.263	0.422	1.292	**0.153**
PGD† [40]	ResNet-101	1600 × 900	0.428	0.369	0.683	**0.260**	0.439	1.268	0.185
DETR3D† [41]	ResNet-101	1600 × 900	0.425	0.346	0.773	0.268	0.383	0.842	0.216
BEVFormer-S† [23]	ResNet-101	1600 × 900	0.448	0.375	0.725	0.272	0.391	0.802	0.200
PETR† [26]	ResNet-101	1600 × 900	0.442	0.370	0.711	0.267	0.383	0.865	0.201
BEVFormer [23]†	ResNet-101	1600 × 900	0.517	0.416	0.673	0.274	0.372	0.394	0.198
PolarFormer [17]†	ResNet-101	1600 × 900	0.528	0.432	0.648	0.270	0.348	0.409	0.201
PETRv2† [27]	ResNet-101	1600 × 640	0.524	0.421	0.681	0.267	0.357	0.377	0.186
MV2D-S‡	ResNet-101	1600 × 640	0.470	0.424	0.654	0.267	0.416	0.888	0.200
MV2D-T‡	ResNet-101	1600 × 640	**0.561**	**0.471**	**0.593**	0.262	**0.340**	**0.368**	0.184

Figure: Validation set

Comparative Analysis of Methods

Method	Backbone	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
FCOS3D† [39]	ResNet-101	0.428	0.358	0.690	0.249	0.452	1.434	0.124
PGD† [40]	ResNet-101	0.448	0.386	0.626	**0.245**	0.451	1.509	0.127
BEVFormer† [23]	ResNet-101	0.535	0.445	0.631	0.257	0.405	0.435	0.143
PolarFormer† [17]	ResNet-101	0.543	0.457	0.612	0.257	0.392	0.467	0.129
PETRv2† [27]	ResNet-101	0.553	0.456	0.601	0.249	0.391	**0.382**	0.123
MV2D-T ‡	ResNet-101	**0.573**	**0.483**	**0.567**	0.249	**0.359**	0.395	**0.116**
BEVDet [16]	Swin-Base	0.488	0.424	0.524	0.242	0.373	0.950	0.148
M2BEV‡ [42]	ResNeXt-101	0.474	0.429	0.583	0.254	0.376	1.053	0.190
DETR3D§ [41]	V2-99	0.479	0.412	0.641	0.255	0.394	0.845	0.133
BEVDet4D[15]	Swin-Base	0.569	0.451	**0.511**	**0.241**	0.386	**0.301**	0.121
BEVFormer§ [23]	V2-99	0.569	0.481	0.582	0.256	0.375	0.378	0.126
PolarFormer§ [17]	V2-99	0.572	0.493	0.556	0.256	0.364	0.440	0.127
PETRv2§ [26]	V2-99	0.582	0.490	0.561	0.243	0.361	0.343	**0.120**
MV2D-T§	V2-99	**0.596**	**0.511**	0.525	0.243	**0.357**	0.357	**0.120**

Figure: Test set

The MV2D Architecture

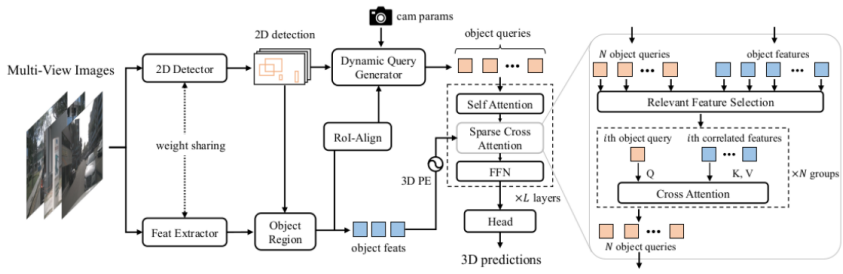


Figure: MV2D Architecture

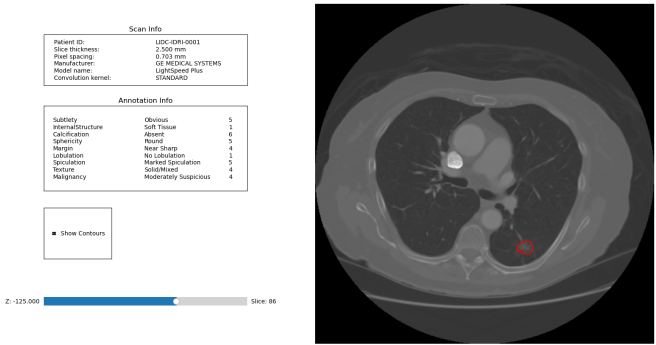
MV2D Introduction

MV2D (Multi-View 2D Objects guided 3D Object Detector) is a two-stage detector.

- 2D Object detection
  - Applied to multi-view images to generate 2D bounding boxes
  - Faster R-CNN, Masked-RCNN, YOLEX, RetinaNet etc.
  - 2D detection results provide priors for object existence and location in image space
- 3D Object detection
  - MV2D derives a 3D reference point conditioned on 2D detection results
  - Dynamic object queries are generated from the 3D reference point
  - Use transformer decoder layers to interact object queries and multi-view features
  - FFN used for 3D object prediction

Data set

- We use LIDC-IDRI database [1] that is available from the NCI Cancer Imaging Archive.
- our input data consists of 791 items, with 632 items used for training (80%) and 159 items used for testing (20%).



[1] LIDC-IDRI Dataset: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI

Data set

3D CT Scan class: DICOM format. Each patient scan is a sequence of 2D slices that together represent a 3D volume with the shape (H, D, W), representing the number of slices along (x, y, z) axes respectively. Convert from voxel representation to physical dimension (in mm) with the `slice_thickness` and `pixel_spacing` attributes.

- Slice thickness: nominal spacing between adjacent slices from the center-to-center of each slice.
- Pixel spacings: distance between the centers of each two-dimensional pixel

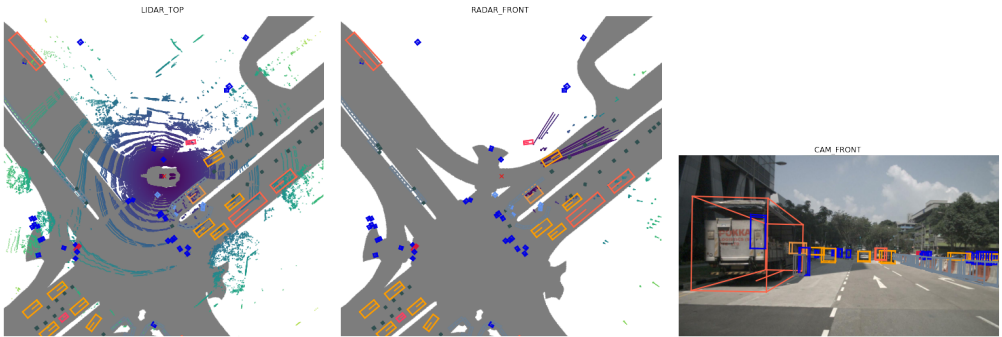
Annotation: segmentation mark up of nodule for each slice.

- Cluster contours in each slice (for the same nodule).
- Ground truth 3d_bbox: is represented as a matrix $A \in \mathbb{R}^{3\times 2}$
- the origin $(0, 0, 0)$ is located at the center of the volume.

Data Formatting

Adaptation from nuScence dataset to LIDC

nuScence dataset use Lidar based system and point cloud.



(a) Lidar top (b) Radar front (c) Cam front

Figure: nuScence pointcloud system

LIDC - Experiment setup

We use `DiffDRR` to digitally reconstructed radiograph (DRR) to generate Xray 2D images from 3D CT scan.

- **Setup:** 10 virtual cameras evenly spaced around a **3D CT volume**.
- Cameras **rotate around the Z-axis** and translate **300 mm** from the CT center.
- Each rotation: $rot_i = 36^\circ i$, where $i \in [0..9]$.
- **Source-to-detector distance: 300 mm** (equivalent to **600 mm** C-Arm in real-world X-ray machines).

LIDC - Experiment setup

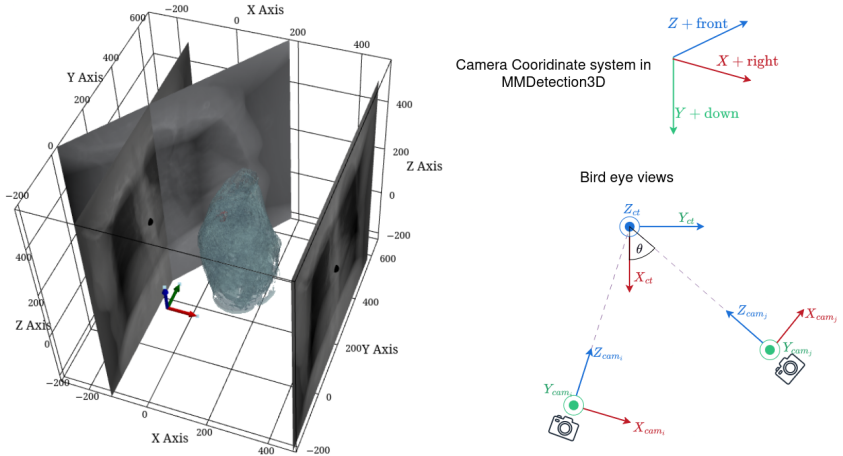


Figure: Visualization of X-ray reconstruction setup

Coordinate Transformation

- **Coordinate transformation and projection plays an important role in how to train the model**
- nuScence uses several systems: LiDAR, ego, sensor, local camera and global world coordinates. $\rightarrow$ align our coordinate system with the nuScenes and MMDetection3D frameworks.
- LIDC systems: CT centered coordinate, Cam coordinate and Image plane.

Coordinate Transformation

- **CT Centered Coordinate (ct):** world coordinate system, origin (0, 0, 0) at the center of the CT volume. The Z-axis points upwards, the Y-axis to the right, and the X-axis points towards the camera (Right-hand coordinate system). The ground truth bounding box is defined in this coordinate system.
- **Camera Local Coordinate ($\mathbf{cam}_i \quad i = 0..9$):** Each camera has its own local coordinate system, defined according to the camera system in MMdetection3D. In this system:
  - The positive Y-axis points downward.
  - The positive X-axis points to the right.
  - The positive Z-axis points towards the object (towards DRR).
- The camera's local coordinate system is obtained by performing rotations and translations from the CT coordinate system. For each camera cam_i, its local coordinate is determined by the transformation from the CT coordinate.

Data processing pipeline

- All images in the dataset have 1024x1024 pixels. Pixel resolution of the images is 0.78125 mm.
- Input data to the model: $I \in R^{1\times10\times3\times1024\times1024}$, which represents 10 views images for 1 patient, each image has 1024×1024 with 3 channels colors.
- Our data is processed by customized MMEngine DataClass and encapsulated in a DataContainer structure, loaded into model using PyTorch's DataLoader.
- We apply photo metric distortion to image sequentially, with a transformation probability of 0.5. Our transformation are: random brightness, random contrast (mode 0), random Gamma Correction, random Gaussian noise, random Gaussian blur, random salt and pepper noise, and random vignetting effect.

Training hyperparameters

- Optimizer: AdamW with based learning rate is 2×10^{-4}.
- Learning rate use scheduling policy CosineAnnealing, decreasing to minimum value is 10^{-3}
- Warm up policy for the first 500 iterations, starting at $\frac{1}{3}$ of base learning rate.
- Our model is trained with 24 epochs.

Feature extraction (ResNet-50)

- Input data: $I \in R^{1\times10\times3\times1024\times1024}$
- Backbone network (ResNet-50) extracts the feature maps, using 50 layers and 4 stages (default setting).

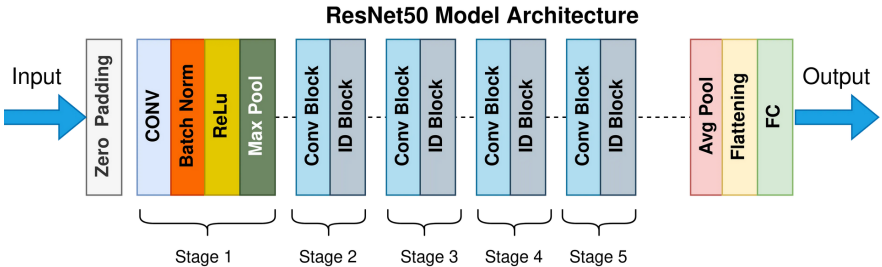


Figure: Architecture of ResNet-50

Feature Pyramid Network (FPN)

Our result is a list of different levels of feature maps $\{\mathcal{F}_i \quad i = 1...5\}$, with $\mathcal{F} = \{\mathbf{F}_v \in R^{C \times H^f \times W^f}\}$.
We use 5 levels of feature map, each has the number of output channel is $C = 256$, and our FPN is built to produce feature maps with downsample stride $\{8, 16, 32, 64, 128\}$

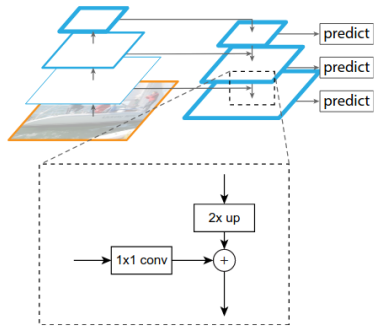


Figure: Architecture of RPN Network

RPN Feature maps

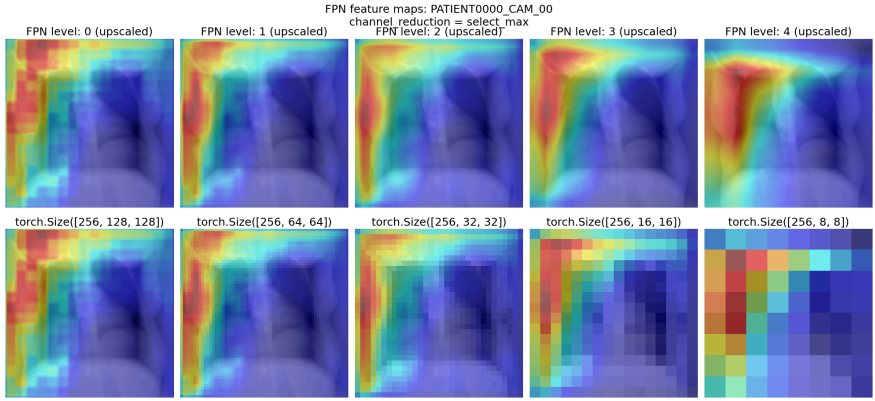


Figure: Feature map result for camera 00, with 5 levels of FPN. Each level has 256 channels. Here we show the channel that has highest activation value. Feature map is upscale to match the image size.

Regional Proposal Network (RPN)

For each image, we receive a result of $Proposal_i \in R^{N \times 5} \quad i : 1...10$, where $N = 1000$ is the number of potential bounding boxes.

- List of anchor boxes is created by sliding window at each feat-map pixel.
- Scale size is adapted for nodule size `scale=[1, 2, 4]` (w.r.t feature map sizes)
- Anchor boxes size w.r.t original picture is scale up corresponding to the RPN strides
  `strides=[8, 16, 32, 64, 128]`
- At each scale, 3 boxes with 3 aspect ratios `ratios=[0.5, 1., 2.]`

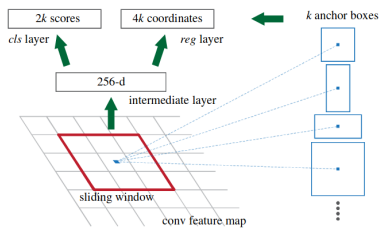


Figure: Region Proposal Network architect. Figure from origin paper [**?**]

Regional Proposal Network (RPN)

- Non-Maximum Suppression (NMS) is used to remove redundant or overlapping bounding boxes.
- Intersection over Union (IoU) threshold for NMS is 0.6.
- RPN loss:
  - Classification loss: binary loss (0-1: contains object or not)
  - Regression loss: L1Loss $L_{\text{L1}} = \sum_{i=1}^{4} |y_i - \hat{y}_i|$

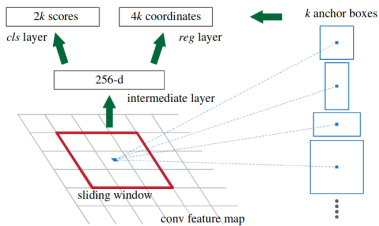


Figure: Region Proposal Network architect. Figure from origin paper [**?**]

Regional Proposal Network (RPN)

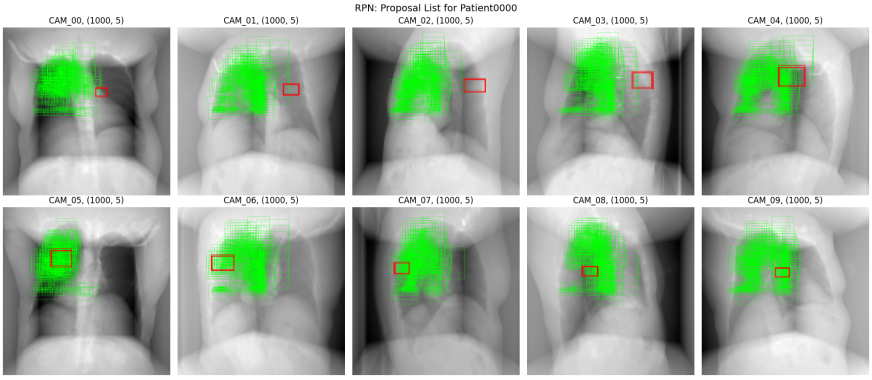


Figure: Original X-Ray image with region proposals (in green).

RoI and 2D Detection

- 2D Detections is calculated based on the upstream features and the proposal list.
- Feature maps are cropped into RPN by using RoI (size 7×7)
- Hyperparameters: the score threshold for a box to be select is 0.05, the maximum number of box per image is 100, and the IoU threshold for NMS is 0.7.

2D detection loss:

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

where:

- p_i and p_i^* are the predicted probability (our score) and ground truth of anchor i being an object. (1 - object, 0 - background)
- t_i and t_i^* represent the 4 coordinates of the predicted bbox and ground-truth bbox.

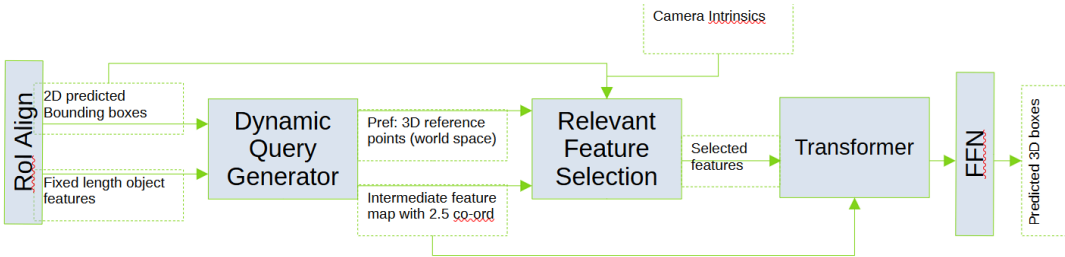
- Classification loss L_{cls} is cross entropy loss over two classes (object vs. not object).
- Regression loss L_{reg} is the L_1 robust loss function.

3D Object detection

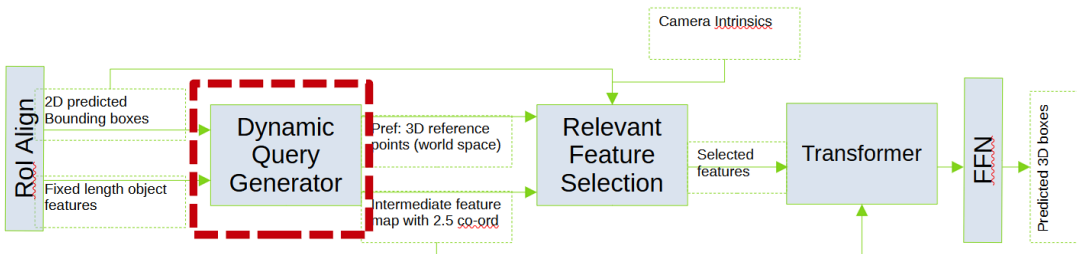
It contains 3 parts:

- Dynamic Query Generator: Generates dynamic queries conditioned on 2D detection results.
- Relevant feature selection: Selects only the features that are pertinent for the 3D object localisation
- Use transformer decoder layers to interact object queries and multi-view features. FFN used for 3D object prediction

3D BBox detection pipeline



Dynamic Query Generator



Dynamic Query Generator

- Input: RoI-Aligned feature maps and 2D bounding boxes
- Output: 2.5D coordinates (p_v^i), 3D reference point (p_{ref}), intermediate feature map (with encoded 2.5D coordinate feature concatenation) and object queries
- Generates dynamic queries conditioned on 2D detection results

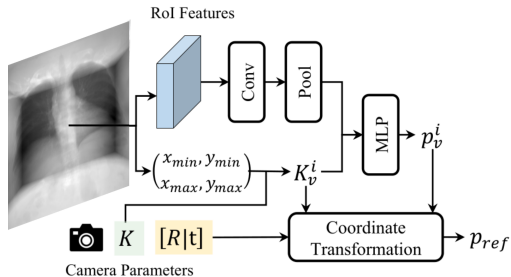


Figure: Query generator Network

Dynamic Query Generator

- 1D conv: Extract spatial features for estimating 3D position
- Average pooling: Reduce spatial dimension of feature map, summarize information
- MLP: Anchor points in 3D space are transformed to object queries by MLP.
- Encode with coordinate information (using camera intrinsic) to get the intermediate feature map. (3D position aware features)

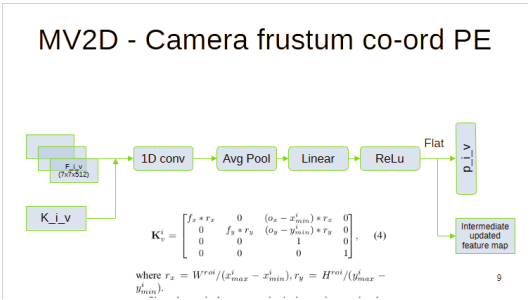


Figure: Camera Frustum (2.5D) representation

Dynamic Query Generator: Object Queries

$$\mathbf{p}_{ref} = [\mathbf{R}|\mathbf{t}]^{-1}(\mathbf{K}_v^i)^{-1}\,\mathbf{p}_v^i$$

Pref is normalized as:

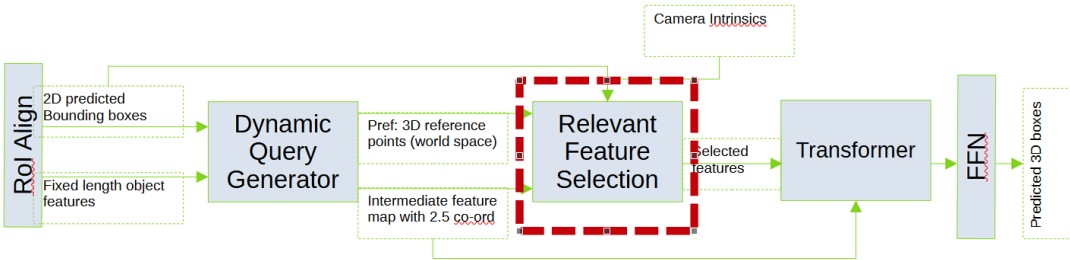
$$\begin{aligned} x_{i,j} &= (x_{i,j} - x_{min})/(x_{max} - x_{min}) \\ y_{i,j} &= (y_{i,j} - y_{min})/(y_{max} - y_{min}) \\ z_{i,j} &= (z_{i,j} - z_{min})/(z_{max} - z_{min}) \end{aligned}$$

Queries are generated as:

$$\begin{aligned} &\mathbf{q} = \text{Linear}(PE), \\ &PE_{[6i:6i+3]} = \sin(\mathbf{p}_{ref}/10000^{2i/C}), 0 \le i < C/2, \\ &PE_{[6i+3:6i+6]} = \cos(\mathbf{p}_{ref}/10000^{2i/C}), 0 \le i < C/2. \end{aligned}$$

p_{ref}: This is the 3D point representing the bounding box in world coordinate space. Positionally encoded p_{ref} is the object query.

Relevant Feature Selection



Filter out irrelevant features before putting them to the transformer (i.e making the input *sparse*).

Relevant feature selection

- Project p_{ref} onto image plane for camera v and bounding box i
- Project the p_{ref} of all other cameras into the image plane of camera v
- Check for overlap
- Top IoU and All overlapped IoU

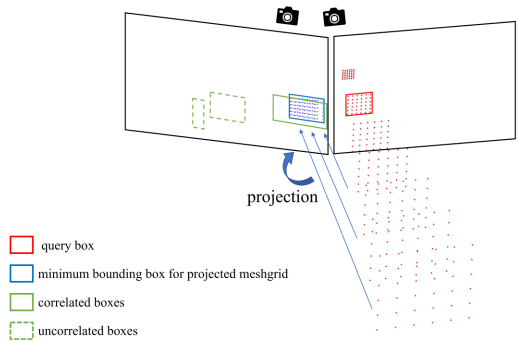
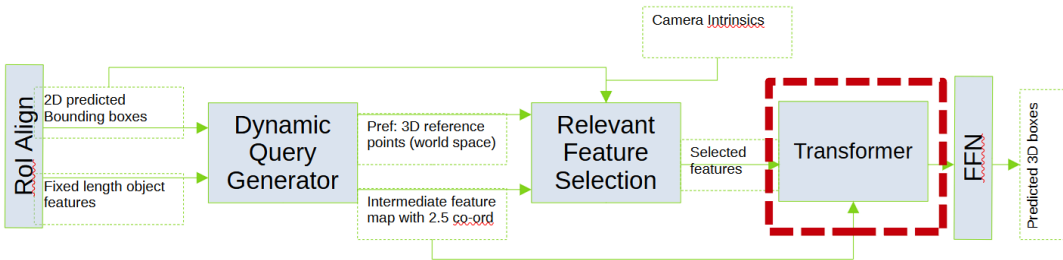


Figure: Camera Frustum (2.5D) representation

Transformer



We want Object queries to further interact with 3D position aware features to predict corresponding 3D objects.

Transformer

- Decoder transformer performs causal self-attention on the position encoded P_{ref} values
- Performs cross-attention to iteratively refine and process object queries by interacting with selected image features
- 6 decoder layers (iterative) to attend the object query to the 3D position aware features
- Multi head attention for parallelism
- A simple feed-forward network (FFN) head in each layer is used to generate 3D object predictions with refined representation of object queries

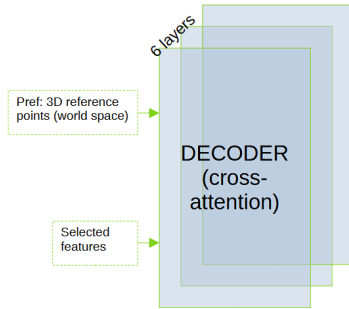


Figure: Decoder transformer

Detection Head

- Hungarian assigner algorithm used for label assignment (matching GT 3D box to predicted 3D box)
- 2 branches: One for classification and one for regression
- Input: Updated object queries from decoder output
- Focal Loss is used for classification
- L1 loss used for regression

The overall loss is given as:

$$\mathcal{L} = \lambda_{cls3d} \times \mathcal{L}_{cls3d} + \mathcal{L}_{reg3d} \tag{1}$$

where $\lambda_{cls3d} = 0.25$ in our setup. The overall loss function of MV2D is a combination of 2D and 3D loss:

$$\mathcal{L} = \mathcal{L}_{2d} + \lambda_{3d} \times \mathcal{L}_{3d} \tag{2}$$

where $\lambda_{3d} = 0.1$ in our setup.

Data Augmentation

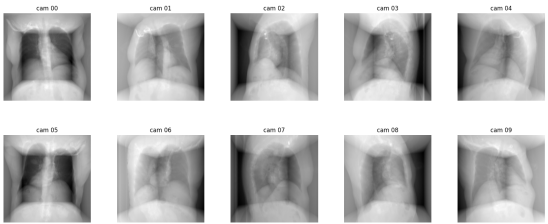


Figure: Regular data

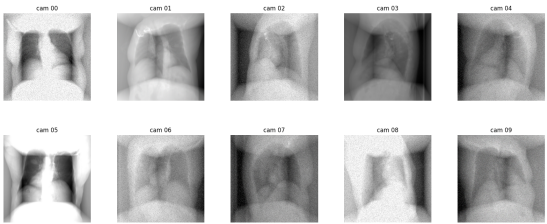


Figure: Augmented data

Brightness, Contrast, Gamma Correction, Gaussian noise, Gaussian blur, Salt and pepper noise, Vignetting effect are applied randomly.

Input verification

- Slight misalignment observed between 2D ground truth and 3D ground truth
- Possible issue in the extrinsic parameter values based on the actual camera positions

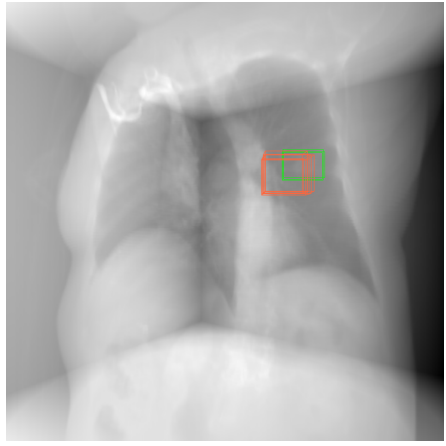
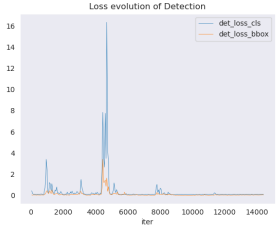
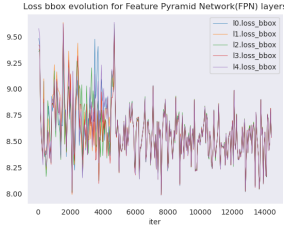


Figure: 3D ground truth matched against 2D ground truth

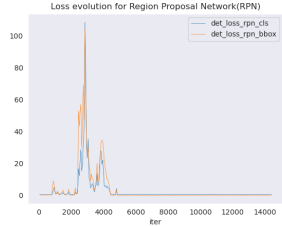
Training



(a) Detection loss



(b) 2D bbox loss



(c) RPN loss

Figure: 24 epochs training run with 632 patient data, AdamW, lr $= 2 \times 10^{-4}$, CosineAnnealing

The overall training time is $8 - 9$ hours with a Tesla-T4 GPU and 16GB of VRAM.

2D detector output

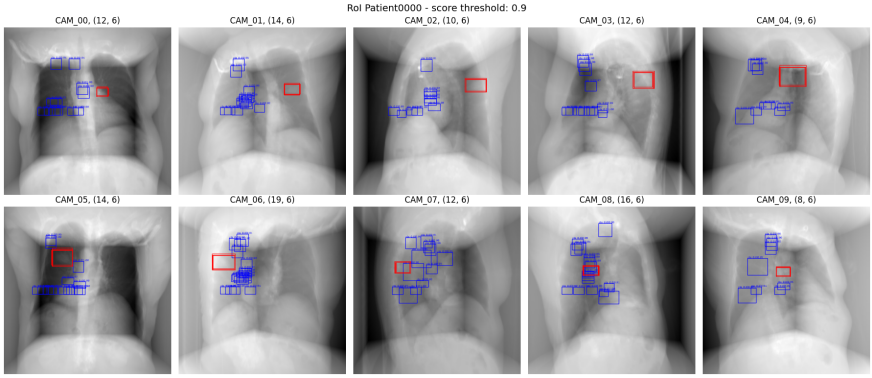
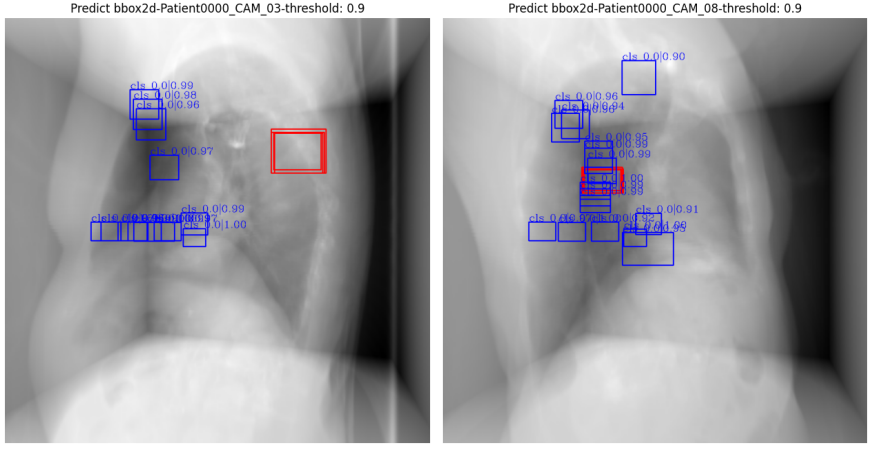


Figure: RoI of Patient0000 with 10 different camera positions. The score threshold is set to 0.9, so the model will only select bounding box that has classification score larger than 0.9. The predictions are tensor of shape `(N, 6)` with `N` is the number of predicted 2D boxes, 6 elements in the last dimension represent `(x, y, dx, dy, class, classification score)`

2D detector output

- Our model fails to recognize the correct position of nodules in most of the images, some predictions are quite closed to the ground truth, while others are really far from ground truth.
- our predictions cluster around the left side of the image, where there is more detail than on the right side.
- suggests that our model mostly focuses on region that has more details in the inputs, and it fails to recognize nodules that are located in blurry region. This comes from the nature of X-Ray image, where sometime the details are not so easy to be captured. Starting from this observation, we trace back the forward pass process to investigate the problem.

2D detector output



(a) CAM03

(b) CAM08

Figure: Detailed bounding boxes prediction for camera 03 and camera 08

Feature maps analysis

Theoretically, FPN captures information at multiple scale and detects objects of various sizes (small, medium, large objects).

- Our nodules is relatively small. We expect that the most distinctive features (to separate nodules from the background) will be captured by the middle levels (levels 0 to 1).
- We analyze feature maps to see "where" our model mostly focus on.

Feature maps analysis

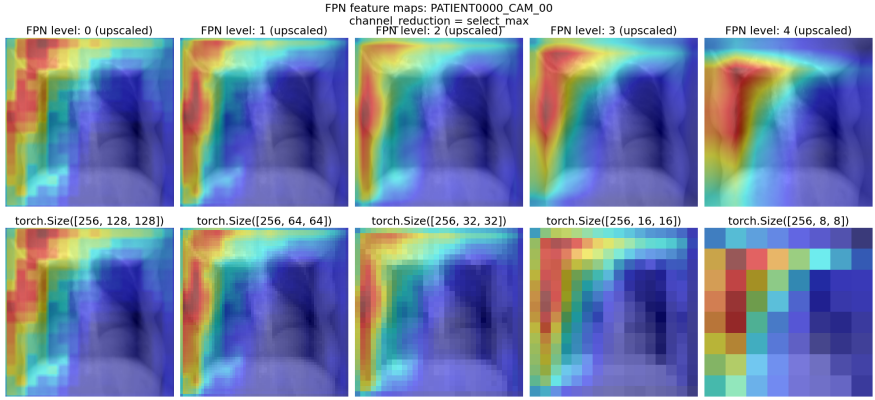


Figure: Example of 5 levels of FPN. We selects the channel that has highest activation value. The feature map is upscale to match the image size.

Feature maps analysis

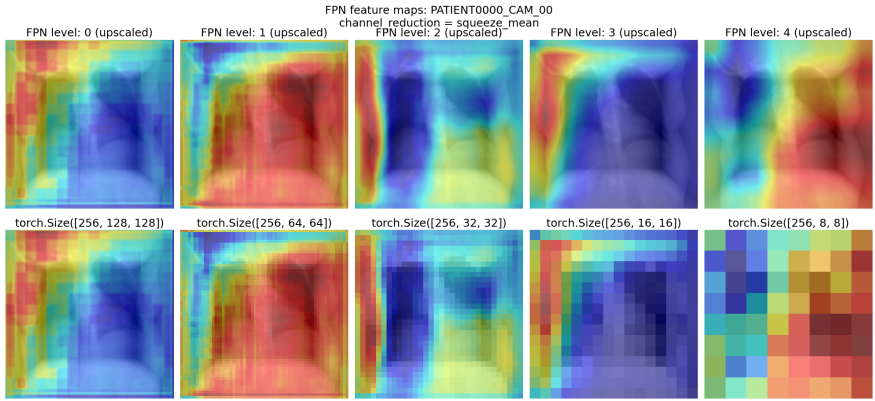


Figure: Example of 5 levels of FPN. We average the activation values across all the channels. The feature map is upscale to match the image size.

Feature maps analysis

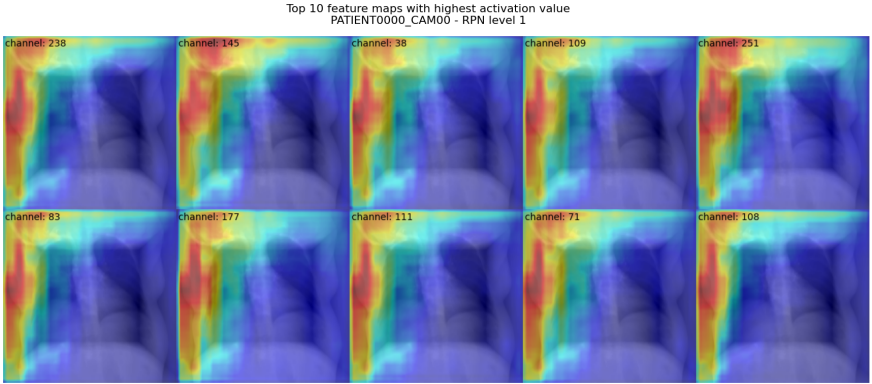


Figure: Top 10 channels with highest activation value from RPN level 1. Feature maps are up-scaled to match original size of image.

Conclusion

- Used MMDetection framework.
- Correlation with the courses:
  - Saw the Hungarian Assigner in action
  - Cosine Annealing
  - Transformer decoder architecture

List of Works:

- Data preparation: A tool was provided to format the input data in order to be functional with the MV2D pipeline
- Modification of the MV2D pipeline to remove Nuscenes specific transformations and added transformations pertaining to the Safran use-case
- Data augmentation added for XRAY images
- Estimation of computing resources for the Safran use-case

Future work

- Retrain the 2D detector with pre-trained weights of XRAY data to enhance 2D detections
- Use deformable convolution networks and deformable attention networks
- Refine feature selection and Attention mechanisms: Since XRay images can be noisy, we need to ensure that object queries focus on genuine anomaly structures rather than noise artifacts
- Improve data augmentation to include rotation and flipping
- Add a transformer encoder. This will allow the features to interact with the other features in the feature map to capture global information.

Thank you