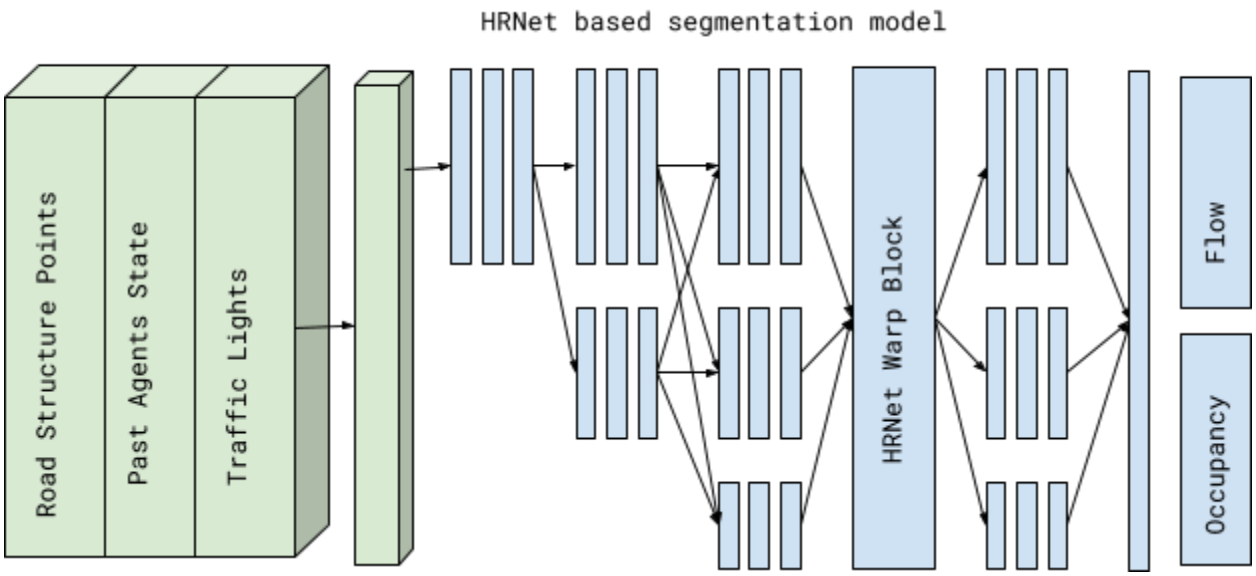
Waymo Open Dataset Occupancy and Flow Prediction Challenge Solution: Look Around

Dmytro Poplavskiy

Dmytro.Poplavskiy@gmail.com

Abstract

The solution is based on the HRNet based segmentation model with a spatial transformer module based residual blocks. The roads network, current, past and projected future agents states are combined as an input to the segmentation network.



Input encoding

The input is encoded as a sparse grid with the same resolution of 3.2 pixels per meter as the output labels, with a separate channel for every feature.

Every input point includes

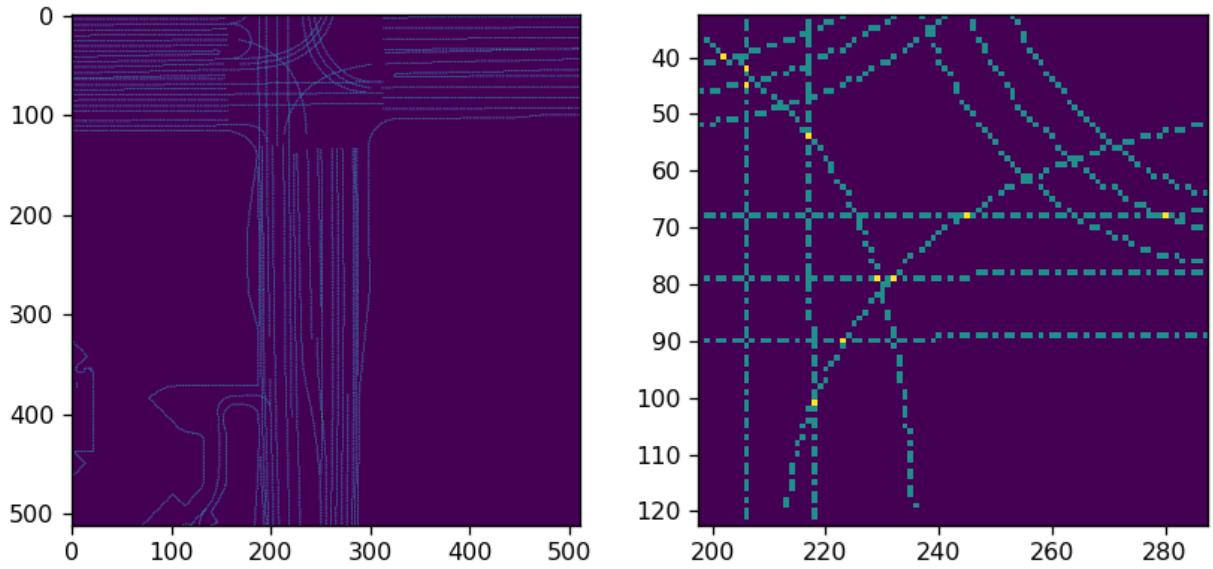
- Road structure points
    - Road feature type, one hot encoded (20 classes)
    - Road feature direction, xyz vector
    - Road feature elevation offset, relative to the SDC
- Past agent states. For the current time step and 0.1, 0.3, 0.6, and 1.0 second time points in the past:
    - Agent type, one hot encoded (5 classes)
    - Agent yaw, encoded as sin/cos
    - Agent velocity vx, vy
    - Agent velocity yaw, encoded as sin/cos
    - Agent length, width, height
    - Agent z, relative to SDC
- Projected future agent states. Using the last estimated velocities, project every agent 1 to 8 seconds ahead, using the same fields as the past agent states.
- Traffic lights. Using the same history time points of the current ts and 0.1, 0.3, 0.6, and 1.0 seconds back. Save either the single point of the end of the TL related lane or the full lane, in both cases one hot encoded (9 traffic light states).

The resulting input tensor is quite deep, with a total depth of 282 channels but very sparse, so to avoid bottlenecks it was beneficial to keep it as sparse data and convert it to the dense presentation, downsampled to 96 channels using 1x1 2D convolution on GPU.

Even though the grid is quite dense, at 3.2 pix/m while the road sample step is 0.5m, so most of the time every pixel would contain only one road feature or agent state (instead of uniformly sampling the agent bounding box, only the single central point is used), it may still be useful to apply PointPillars like encoding.

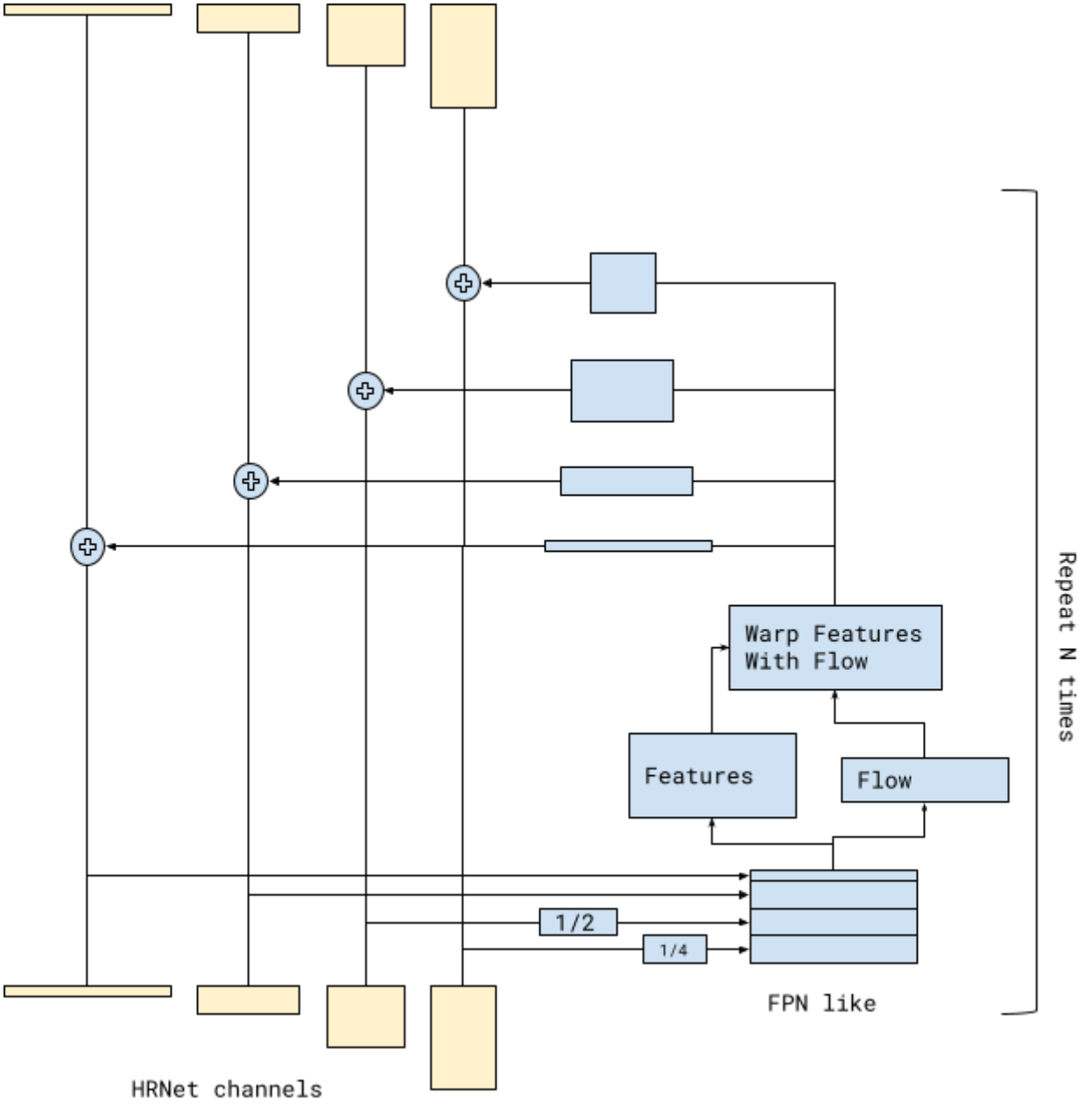
The input is prepared for a 512x512 pixels area, an extra 128 pixels from each side of the target labels.

Here is an example of a scene with the multiple road structures saved for the same input pixe, for such point the information about all the road structure types is preserved due to combination of one hot structure type encoding, but the direction is saved only for the single type:



Decoder Model

Existing segmentation models like HRNet or EfficientDet performed reasonably well, with larger and deeper models performing better. The submission is based on the Imagenet pre-trained HRNet64 backbone, with an extra warp block added between the blocks of the last HRNet stage:



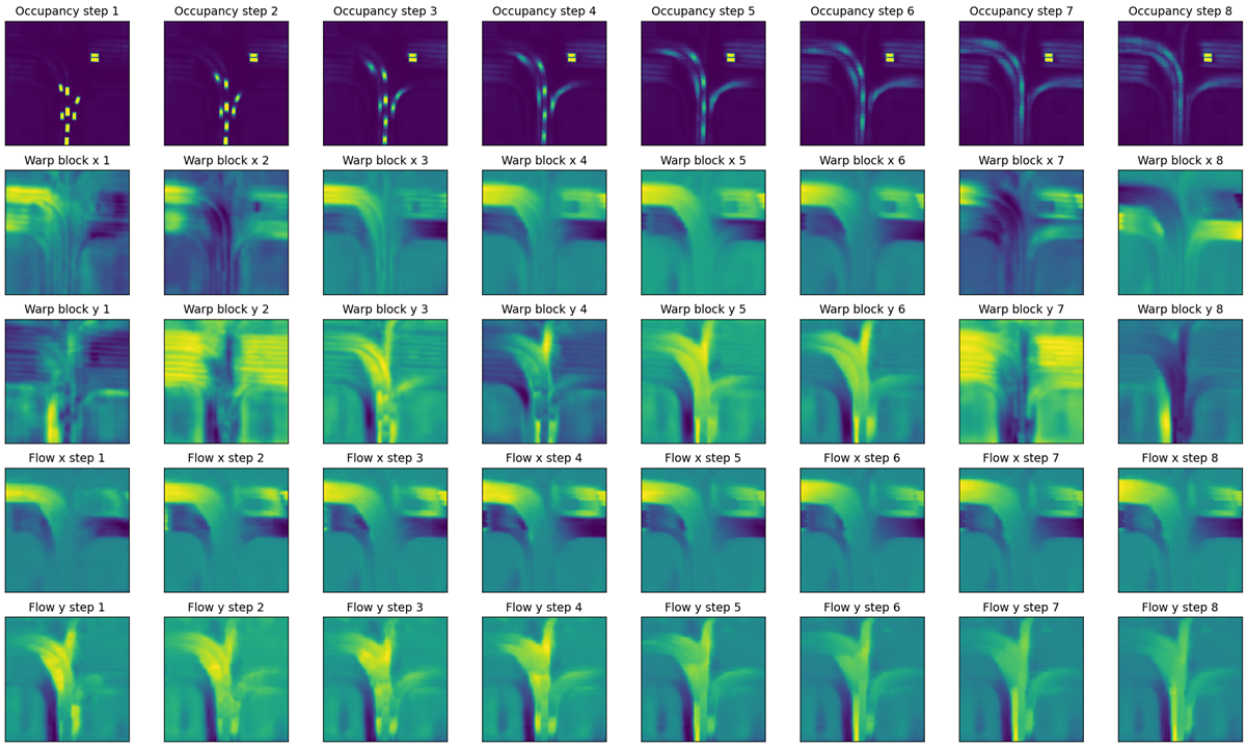
The warp block combines the multiple resolution channels with FPN like block, projected to the learnable flow fields and features tensor, to be warped with the flow. The warped features are scaled back to the resolution of the HRNet channels and added as residual blocks.

In total 8 blocks were added to the HRNet model to be able to make different query types. The warp block can be applied to other decoders like BiFPN from EfficientDet.

The initial intuition behind the warp block was to allow the model to predict and project the agents' state on the grid step by step instead of the single large 8 seconds prediction. However, when added another loss to have the warp flow of every warp block follow the predicted Nth

second flow, the performance improvement from the warp module significantly deteriorated, which may indicate the model has learned to query more useful information.

Example of the x/y flow predicted by model output (the first and 3rd rows) and by the internal warp block:

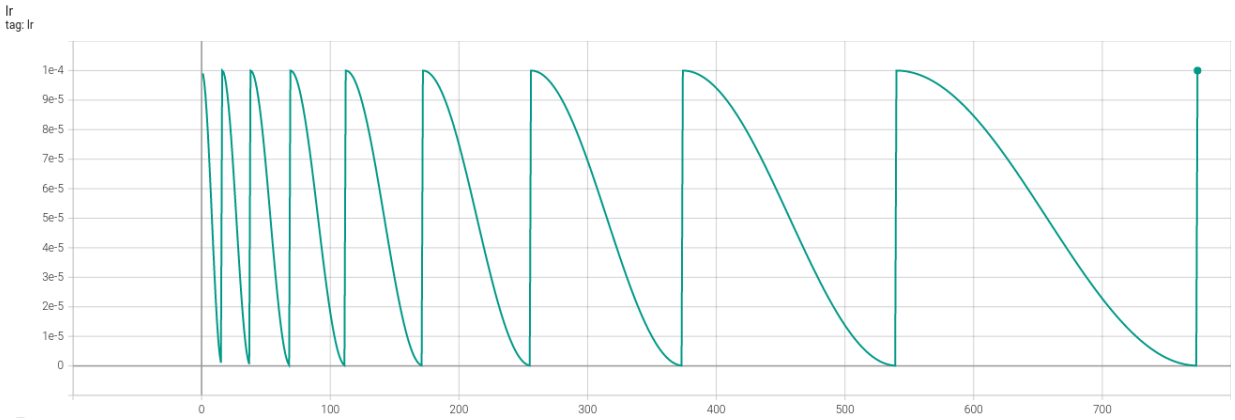


Some warp blocks project features in the same direction as a flow, some in the opposite direction, some check around cars, etc.

Model training and Losses

The model is trained using PyTorch framework, so the target labels are pre-rendered and saved with the scene features. This allowed much faster data preparation, used encoder is very simple and fast. After switching to the sparse representation of inputs, data preparation was not the bottleneck.

The model has been trained using the AdamW optimiser for 770 epochs of 10k samples each, with the initial learning rate of 1e-4 and the CosineAnnealingWarmRestarts scheduler:



The model predicts the flow for 8 steps, the total occupancy and if the predicted occupancy is occluded or not.

The model has been trained using the following losses:

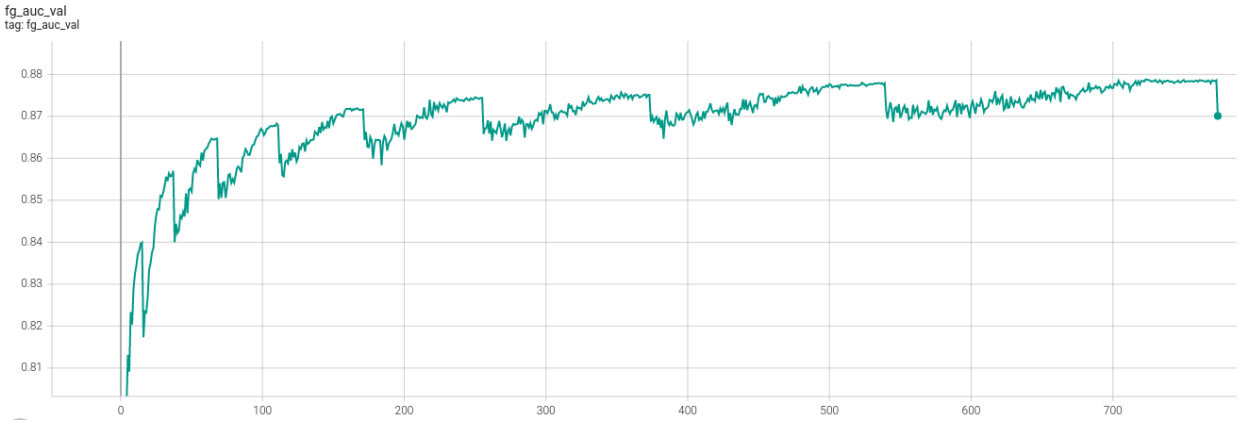
- Flow loss. MAE for all occupied positions at the requested timestamp.
- Occupancy loss: FocalLoss for occupied and occluded masks.
- Flow Warped loss: The predicted occupied mask from the first 7 prediction steps are projected one step ahead using the predicted flow. Applied the Soft IOU and FocalLoss on the predicted occupancy.

The losses are weighted to have approximately similar values, with Soft IOU being lower than FocalLoss:

- occupancy_focal_loss: 10.0 (1.0 was used for the original logloss version, increased the weight 10x since FocalLoss is lower)
- occluded_focal_loss: 2.0
- flow_mae: 10.0
- fg_iou: 0.2 # soft iou
- fg_focal_loss: 2.5

The resulting submission was a simple average ensemble from checkpoints 250, 370, 536 and 750. The improvement from ensembling was only around 0.3% compared to the single 370 or 536 epoch checkpoint.

FG AUC on the validation set (the last shard):



Improvements compared to the baseline solution

What worked, compared to the simple baseline segmentation:

- Replacing LogLoss with FocalLoss, has probably the biggest impact on the AUC score, at the cost of some drop of the SoftIOU score. FocalLoss put much less weight on the simple samples (usually unoccupied areas) and prevents the model from predicting almost exactly zero. Adding separate SoftIOU loss helps to reduce the impact on the SoftIOU metric without much of the AUC degradation. The FocalLoss has probably one of the biggest impact on the final score.
- Increase the input area from 256x256 pixels to 512x512 (adding 128 pixels or 40 meters from each side of the area), so the model has access to agents and road features outside of the target area. The output is cropped back to 256x256 from the centre.
- Add Warp block to HRNet. Not a huge impact but it has improved the AUC score by around 1%.
- Add the model state at future time points, for positions extrapolated using the current velocity. This allows projecting even more agents from outside of the rasterized input. The AUC score improvement was about 0.5%. This would not be necessary with the Scene Transformer like approach.

Overall the Flow-Grounded Occupancy AUC score seems to be quite sensitive to the loss used and can be improved further with more experiments with the loss function.

What did not work so well:

- To include the lane associated with the traffic light instead of the single traffic light point. The score looked the same, with no degradation but no significant improvement.
- To “help” the warp block with the know flow vectors as an aux loss. It worked much better to allow the warp field to be fully trainable.