The whole training procedure of Mask R-CNN can be broken down into 4 main steps described below.

1. **Pretraining DeepLabV3:** We start by loading a pretrained DeepLabV3 implementation to obtain better parameter initialization. We then pretrain DeepLabV3 on the Cityscapes semantic segmentation dataset. At this stage, the FPN is not involved; the ResNet-50 backbone learns features that are more useful for Cityscapes images.
2. **Training Masked RCNN with FPN and heads unfrozen but resnet50 frozen (Stage-1):** Next, we plug the pretrained ResNet-50 backbone into Mask R-CNN. We train the full network on instance segmentation tasks with FPN and all heads of RPN and ROI unfrozen but the resnet50 backbone frozen for only 4 epochs. This allows the FPN to adapt to the newly learned backbone features. Training the network with all of the weights unfrozen in our experience was leading to overfitting after a couple of epochs. This makes sense since resnet50 backbone contains 25 million parameters and the whole Masked R-CNN contains more than 40 million parameters so allowing all parameters update increases risk of overfitting a lot.
3. **Freezing Backbone and FPN to train only RPN and ROI heads(Stage 2) :** Finally, we freeze the backbone, borrow FPN from stage1, and run a hyperparameter tuning procedure on thresholds of RPN and ROI. The ROI and RPN heads are initialized randomly to allow for flexible experimentation with hyperparameters. This approach reduces the number of parameters being trained, speeds up convergence, and makes hyperparameter tuning more efficient. After finding the best hyperparameter configuration we train the model with FPN and backbone freezed for 6 epochs. Hyperparameter tuning revealed that unfreezing FPN in this stage did not yield a noticeable boost so we freeze FPN along with the resnet50 backbone during the training.
4. **Additional 2-epoch training for final boost (Stage 3):**

Lastly, we unfreezed all of the weights again (including backbone) and lowered the learning rate by a factor of 10 for the final boost on the best model we have got after stage 2. We trained on the 2 of the validation set partitions that were used during hyperparameter selection and stage 1 training to monitor the validation [loss. Since the final test set is separate (the last 2 partitions of the whole validation set that weren’t used until final test set evaluation ) we do not worry too much about data leakage. We](http://loss.we) trained only for 2 epochs to avoid overfitting on the training dataset that was observed when we trained Masked RCNN with all of the weights unfrozen for too long.

**Train, Validation and Test Splitting:**

For pretraining the ResNet50 backbone, we used a subset of semantically annotated Cityscapes data. A detailed description of the training and validation sets is available in the folder **“backbone pretraining.”**

The training data for all three stages came from the same source: **2,975 high-quality, instance-annotated street scenes** captured from vehicles driving in Germany. To make data handling more efficient, the dataset was divided into **8 partitions**, each containing **372 samples** (except the last one, which had **371**).

Each image was converted into a PyTorch tensor with **3 channels (RGB)** and stored in **safetensor** format. Consequently, each partition contains:

* An image tensor of size **[372, 3, 1024, 2048]**, representing the stacked RGB images. The tensor is in FP32 so the size of each partition is 9.36 GB.
* An annotation tensor of size **[372, 1024, 2048]**, representing the corresponding semantic labels for each scene. Similarly a FP32 tensor so each partition is 3.12 GB

The validation set was derived from the official Cityscapes validation split, which contains 500 instance-annotated scenes captured in different cities than those used for training. We divided this set into four equal partitions of 125 scenes each, with both the RGB images and their corresponding instance segmentation annotations included.

* **Partition 1** was reserved for experimentation, including trial-and-error with hyperparameter tuning and adjustments to the training loop.
* **Partition 2** was used to monitor validation loss during training across all three stages. This partition and partition 1 was also used for the extra training we mentioned in stage 3.
* **Partitions 3 and 4** served as the final test set, providing an unbiased evaluation of the trained Mask R-CNN on unseen data.

All of the training was performed on Nvidia L4 GPU with 24 GB gddr6 RAM. For full-scale training on 2975 images each epoch took about 30 minutes.