Lanedet-Carla 활용 가이드

목차

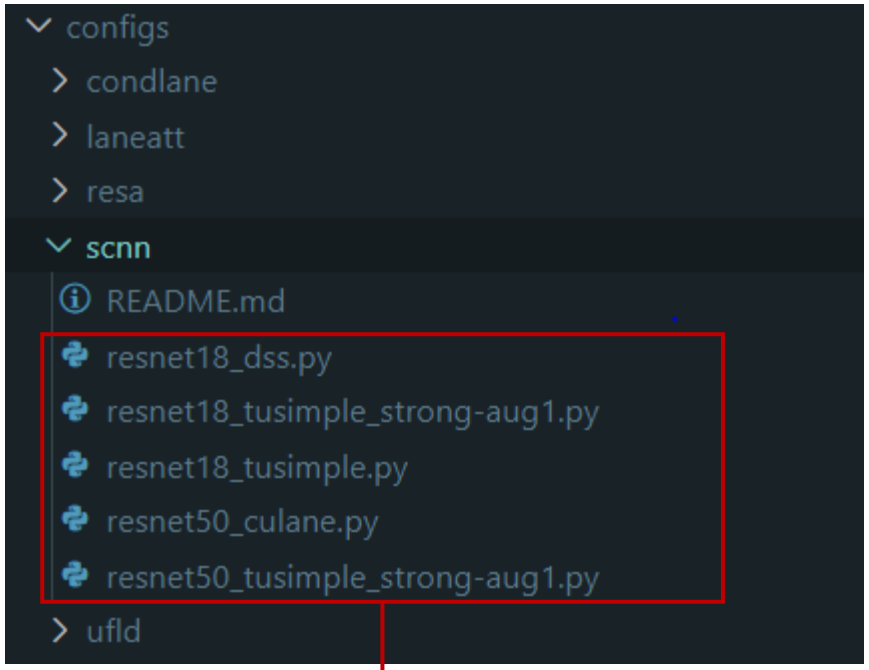
1. Custom 모델 학습 방법 방법

학습에 필요한 설정과 전반적인 학습 파이프라인은
configs/{support_model}/{config_name}.py 파일에서 정의할 수 있다.

Config 파일에서는 아래와 같은 내용을 설정할 수 있다.

- 모델 구조
  - backbone, neck, heads …
- 하이퍼 파라미터
  - num_class, epochs, batch_size, lr, lr_scheduler …
- 학습에 과정에 필요한 요소
  - optimizer, preprocess, augmentation …
- 데이터셋
  - TuSimple, DssDataset, CULane

Config에 설정 된 모든 모듈은 type에 해당하는 이름과
동일한 이름의 클래스로 정의되어 있고 @register로 등록된다.



config file

```
backbone = dict(
    type='ResNetWrapper',
    resnet='resnet18',
    pretrained=True,
    replace_stride_with_dilation=[False, True, True],
    out_conv=True,
)
```

1. Custom 모델 학습 방법 방법

Config 파일이 정의되었다면 main.py 파일을 실행 시켜 학습을 진행할 수 있다. main.py 파일 실행 시 아래와 같은 옵션을 추가할 수 있다.

```python
def parse_args():
    parser = argparse.ArgumentParser(description='Train a detector')
    # parser.add_argument('config', help='train config file path')
    parser.add_argument('--config', default="./configs/scnn/resnet18_tusimple.py")
    parser.add_argument('--exp', type=str, default='test')
    parser.add_argument('--tracking', type=bool, default=False, help='whether to track the experiment')
    parser.add_argument('--work_dirs', type=str, default='work_dirs', help='work dirs')
    parser.add_argument('--load_from', default=None, help='the checkpoint file to resume from')
    parser.add_argument('--finetune_from', default=None, help='whether to finetune from the checkpoint')
    parser.add_argument('--view', action='store_true', help='whether to view')
    parser.add_argument('--validate', action='store_true', help='whether to evaluate the checkpoint during training')
    parser.add_argument('--evaluate', action='store_true', help='whether to evaluate the checkpoint after training')
    parser.add_argument('--gpus', nargs='+', type=str, default='0')
    parser.add_argument('--seed', type=int, default=0, help='random seed')
    args = parser.parse_args()
```

- --exp : 실험의 이름 (실험의 의도를 확인할 수 있는 일관된 형식의 이름을 사용하는 것이 실험 관리에 유리)
- --finetune_from : 사전학습 된 모델의 가중치를 이용 가능 (데이터셋의 양이 부족할 때 유용)
- --tracking : 학습 과정을 모니터링 하고 기록하는 기능 (WandB 지원)
- --gpus : 활용가능한 gpu가 여러 개 일때, multi-gpu로 학습 가능 (gpu id는 공백 없이 ‘,’ 로 구분)

2. 데이터셋 분리 방법 (Dataset Split)

train 데이터셋, valid 데이터셋, test 데이터셋 분리는 lanedet/datasets/dss.py에서
아래와 같이 시나리오 단위로 구분

```
SPLIT_FILES = {
    'trainval': ['scenario_1.txt', 'scenario_2.txt', 'scenario_3.txt', 'scenario_4.txt', 'scenario_5.txt', '
    'train': ['scenario_1.txt', 'scenario_2.txt', 'scenario_3.txt', 'scenario_4.txt', 'scenario_5.txt'],
    'val': ['scenario_6.txt', 'scenario_7.txt', 'scenario_8.txt'],
    'test': ['scenario_9.txt', 'scenario_10.txt'],
}
```

아래와 같이 공백으로 설정할 경우, 자동으로 train : valid : test = 6 : 2 : 2의 비율로 분리 됨

```
SPLIT_FILES = {}
```

- test 데이터셋은 어떠한 경우에서도 학습에 활용되면 안되고 모델의 최종 평가 기준이 되므로 현실 세계를 가장 잘 반영할 수 있는 데이터로 구성하는 것이 중요.
- 또한 test 데이터셋은 레이블의 품질이 높은 것을 이용하는 것이 유리
- 또한 공정하고 일관된 모델의 평가를 위해 test 데이터셋은 변경이 최소화 될 수 있도록 독립적으로 구성하는 것이 유리 (대부분 test 데이터셋은 공개하지 않음)

3. Experiment Tracking

Experiment Tracking Tool 로 WandB 활용

WandB 사용을 위해 계정 생성 필요

```python
    # init wandb
    if args.tracking:
        if cfg.dataset.train.type == "TuSimple":
            wandb.init(project="lanedet_TuSimple", entity="kimjusung2109", reinit=True)
        elif cfg.dataset.train.type == "CULane":
            wandb.init(project="lanedet_CULane", entity="kimjusung2109", reinit=True)
        elif cfg.dataset.train.type == "DssDataset":
            wandb.init(project="lanedet_DSS", entity="kimjusung2109", reinit=True)
        else:
            raise ValueError("Invalid dataset")

        wandb.run.name = args.exp

```

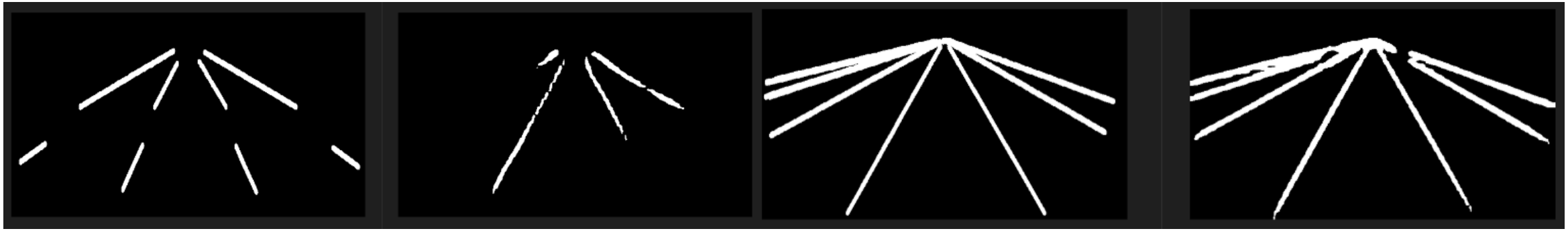
main.py의 entity 부분을 자신의 계정 user name으로 수정

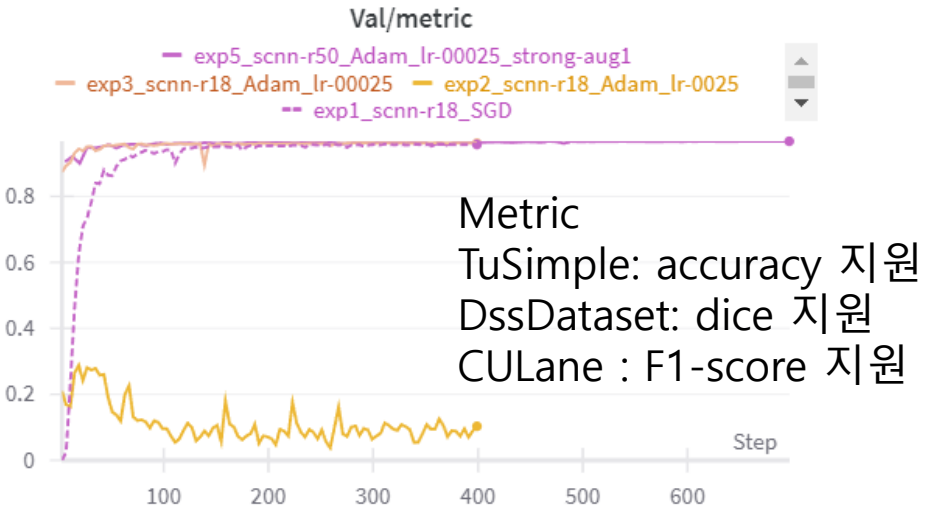
처음 WandB를 사용할 경우 main.py 실행 시, API Key 입력 필요 (API Key는 WandB 사이트에서 확인 가능)

```
wandb: (1) Create a W&B account
wandb: (2) Use an existing W&B account
wandb: (3) Don't visualize my results
wandb: Enter your choice:
```

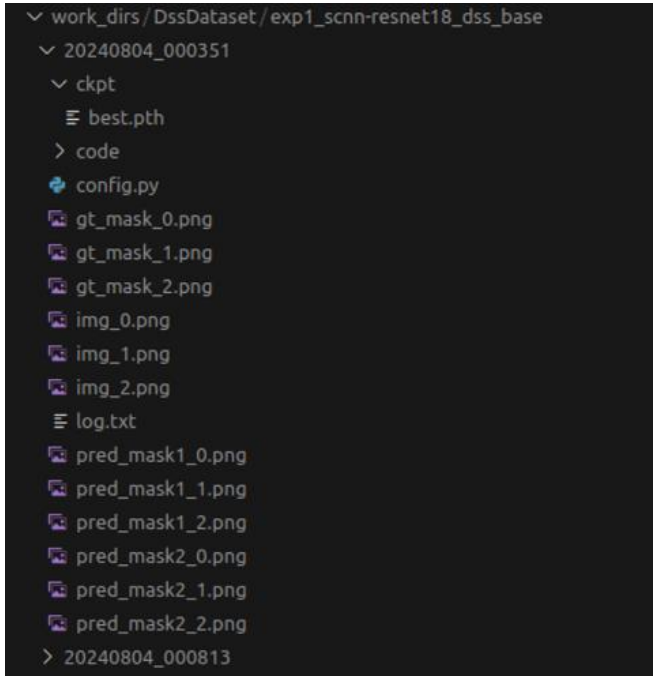
3. Experiment Tracking

학습 과정에서의 inference 결과를 모니터링 가능





Metric
TuSimple: accuracy 지원
DssDataset: dice 지원
CULane : F1-score 지원



exp_name와 동일한 이름의 폴더에 학습 log와 best.pth, config.py 파일이 저장 됨

4. Custom Augmentation 사용법

아래와 같이 Albumentation에서 제공하는 다양한 증강 기법을 학습에 적용할 수 있다

1. Augmentation 정의

```python
# custom transform
@PROCESS.register_module
class JpegCompression(A.JpegCompression):
    def __init__(self, quality_lower=85, quality_upper=95, always_apply=False, p=0.5, cfg=None):
        super(JpegCompression, self).__init__(quality_lower=quality_lower, quality_upper=quality_upper, always_apply=always_apply, p=p)

    def __call__(self, sample):
        sample['img'] = super(JpegCompression, self).__call__(image=sample['img'])['image']
        return sample
```

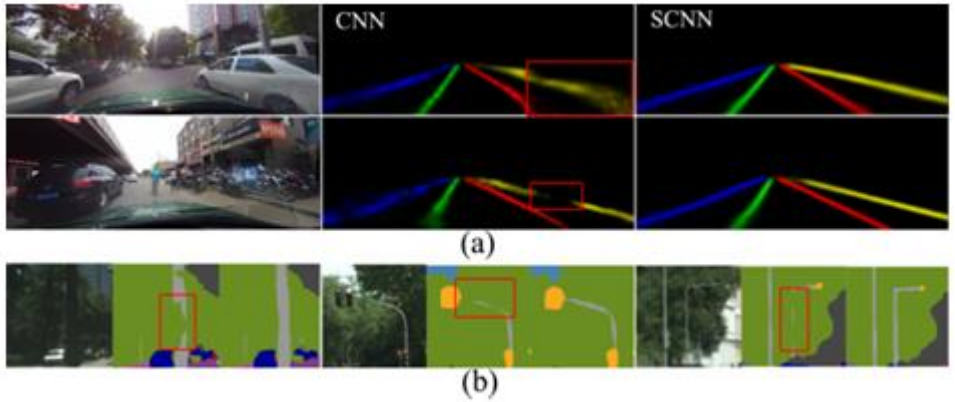
2. Augmentation을 config 파일에 적용

```python
train_process = [
    dict(type='Resize', size=(img_width, img_height)),
    dict(type='RandomRotation'),
    dict(type='RandomHorizontalFlip'),
    dict(type='JpegCompression', quality_lower=85, quality_upper=95, p=0.2),
    dict(type='RandomBrightness', limit=0.2, p=0.6),
    dict(type='GridDropout', p=0.3),
    dict(type='Normalize', img_norm=img_norm),
    dict(type='ToTensor'),
]
```

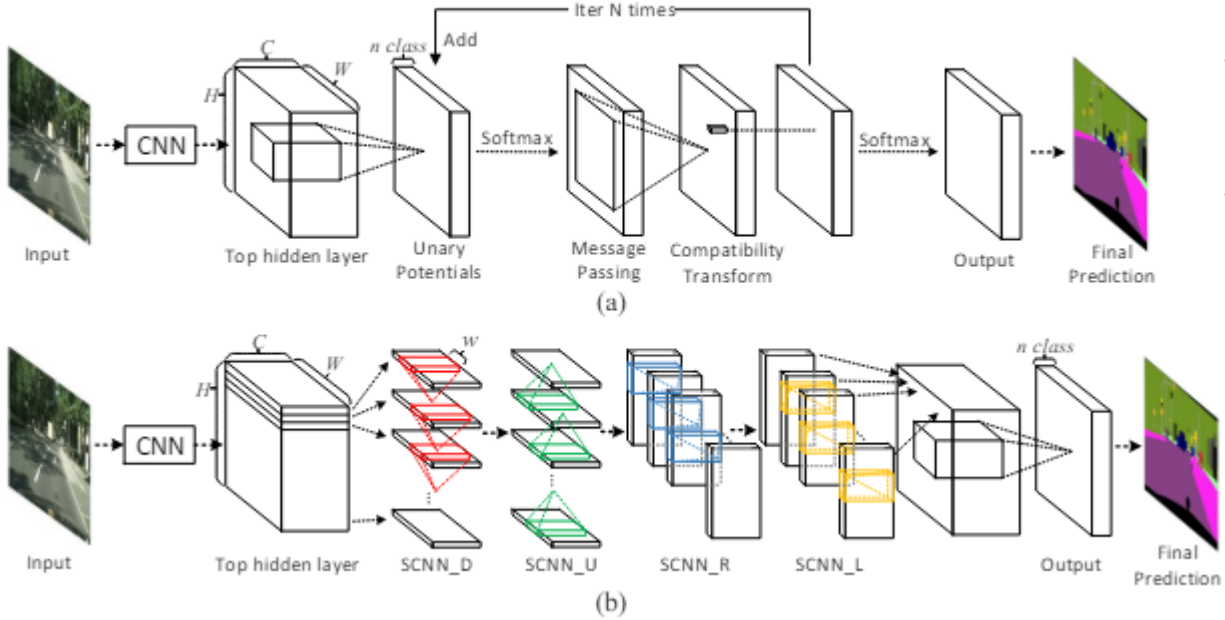
주의. mask label 또한 변형되어야 하는 경우 mask label에도 증강 기법을 적용해야 함

5. Carla Demo

6. SCNN (Spatial CNN) 논문 리뷰



- [Spatial As Deep: Spatial CNN for Traffic Scene Understanding]
- 도로 위 차선과 같은 반드시 검출이 필요하지만
  형태적인 특징(종, 횡 방향으로 얇고 긴) 때문에 특징 추출이 어려운
  객체에 대해 Spatial CNN의 방법을 제안하고 효과적임을 검증
- SCNN (※ spatial convolution과는 다른 개념)
- 또한 오른쪽 사진(a)과 같이 차량에 가려 보이지 않는 영역에 대해서도 효과적으로 차선을 탐지 할 수 있음을 주장



- 기존 CNN은 정보가 layer-wise로 전달 되었다면
  SCNN은 slice-wise로 전달
- 이를 위해 3-D tensor는 높이, 폭, 채널 방향의 layer단위로 나뉘고 각 layer는 RNN의 방식으로 정보를 전달
  (이때 RNN과 마찬가지로 커널 자체는 weight를 공유)
  정보 전달은 D, U, R, L (아래, 위, 오, 왼) 방향으로 수행

6. SCNN (Spatial CNN) 논문 리뷰

```python
def message_passing_once(self, x, conv, vertical=True, reverse=False):
    """
    Argument:
    ----------
    x: input tensor
    vertical: vertical message passing or horizontal
    reverse: False for up-down or left-right, True for down-up or right-left
    """
    nB, C, H, W = x.shape
    if vertical:
        slices = [x[:, :, i:(i + 1), :] for i in range(H)]
        dim = 2
    else:
        slices = [x[:, :, :, i:(i + 1)] for i in range(W)]
        dim = 3
    if reverse:
        slices = slices[::-1]

    out = [slices[0]]
    for i in range(1, len(slices)):
        out.append(slices[i] + F.relu(conv(out[i - 1])))
    if reverse:
        out = out[::-1]
    return torch.cat(out, dim=dim)
```

Feature map을 slice 단위로 분리

(vertical과 reverse 정보로 U, D, R, L 결정)

$$X'_{i,j,k} = \begin{cases} X_{i,j,k}, & j = 1 \\ X_{i,j,k} + f\left(\sum_m \sum_n X'_{m,j-1,k+n-1} \times K_{m,i,n}\right), & j = 2, 3, \ldots, H \end{cases}$$

현재 slice vector와 이전 출력 vector를 합한 값을 2 ~ H 까지 반복하며 누적하고 최종적으로 dim 방향으로 concat 하여 output tensor를 생성

출력 채널이 유지되는 convolution 사용

```python
# ---------------- add message passing ----------------
self.message_passing = nn.ModuleList()
self.message_passing.add_module('up_down', nn.Conv2d(128, 128, (1, ms_ks), padding=(0, ms_ks // 2), bias=False))
self.message_passing.add_module('down_up', nn.Conv2d(128, 128, (1, ms_ks), padding=(0, ms_ks // 2), bias=False))
self.message_passing.add_module('left_right',
                                nn.Conv2d(128, 128, (ms_ks, 1), padding=(ms_ks // 2, 0), bias=False))
self.message_passing.add_module('right_left',
                                nn.Conv2d(128, 128, (ms_ks, 1), padding=(ms_ks // 2, 0), bias=False))
# (nB, 128, 36, 100)
```

6. SCNN (Spatial CNN) 논문 리뷰

- 해당 논문에서는 성능의 평가를 위해 Recall과 Precision을 모두 고려한 **F1 score**를 사용

Evaluation In order to judge whether a lane marking is successfully detected, we view lane markings as lines with widths equal to 30 pixel and calculate the intersection-over-union (IoU) between the ground truth and the prediction. Predictions whose IoUs are larger than certain threshold are viewed as true positives (TP), as shown in Fig. 6. Here we consider 0.3 and 0.5 thresholds corresponding to loose and strict evaluations. Then we employ F-measure $= \frac{(1+\beta^2)\,\text{Precision}\,\text{Recall}}{\beta^2\text{Precision}+\text{Recall}}$ as the final evaluation index, where Precision $= \frac{TP}{TP+FP}$ and Recall $= \frac{TP}{TP+FN}$. Here β is set to 1, corresponding to harmonic mean (F1-measure).

Models	Baseline	ExtraConv	SCNN_D	SCNN_DU	SCNN_DURL
F1 (0.3)	77.7	77.6	79.5	79.9	**80.2**
F1 (0.5)	63.2	64.0	68.6	69.4	**70.4**

Table 2: Experimental results on SCNN with different kernel widths.

Kernel width w	1	3	5	7	9	11
F1 (0.3)	78.5	79.5	80.2	80.5	**80.9**	80.6
F1 (0.5)	66.3	68.9	70.4	71.2	71.6	**71.7**

Table 3: Experimental results on spatial CNN at different positions, with $w = 9$.

Position	Output	Top hidden layer
F1 (0.3)	79.9	80.9
F1 (0.5)	68.8	71.6

SCNN 적용 위치에 따른 성능 비교

더 많은 정보를 갖고 있는 Top hidden layer에 SCNN 모듈을 적용했을 때 더 높은 성능을 보임 (기본적으로 SCNN은 어느 위치든지 적용 가능)

정보 전달 방향의 조합에 따른 성능 비교

DURL 방향을 모두 사용했을 때 가장 높은 성능을 보임

커널 사이즈에 따른 성능 비교

일반적인 CNN 보다 큰 커널 사이즈를 사용했을 때 더 높은 성능을 보임

이는 SCNN에서 종, 횡 방향로 긴 객체의 특징을 추출하기에 더 큰 커널 사이즈를 필요로 하기 때문으로 보임

7. CondLaneNet 논문 리뷰 https://arxiv.org/pdf/2105.05003

CULane Banchmarks에 대한 성능 비교

CondLaneNet overview

- CondLaneNet은 **instance-level discrimination** 문제 해결을 위해 Conditional Convolution for Instance Segmentation (CondInst)의 방식을 사용
- CondInst는 대중적으로 알려진 Instnace Segmentation 방법(=object detection을 선행 후 crop된 ROI에 대해 segmentation 진행)과 비교해 단순(fully convolution network)하면서도 효과적인 구조로 Instance Segmentation task 문제를 해결한 논문
- 단순히 line에 대한 Instance Segmentation하는 것을 넘어 **Lane detection에 특화된 추가적인 방법을 적용** (RIM, row-wise formulation)

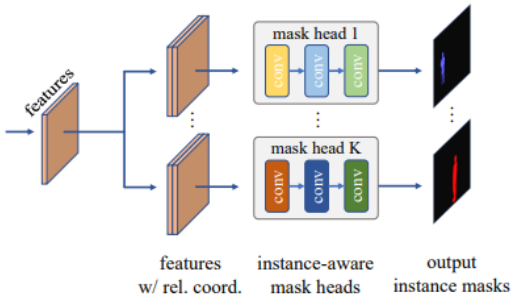
7. CondLaneNet 논문 리뷰

Lane detection에서 왜 Instance-level discrimination 문제를 해결해야 할까?

- 차선과 차선은 구분되어야 한다 (ex. 왼쪽 차선과 오른쪽 차선은 둘 다 차선이지만 서로 다른 차선으로 인식해야 한다)

Instance-level discrimination 문제를 해결하기 위한 다양한 방법

- 해당 문제를 해결하기 위한 가장 쉬운 접근 방법 -> **multi-class classification** -> 인식 가능한 차선의 수가 고정된다
- 이를 해결하기 위해 **post-clustering** 접근이 시도되었지만 밀집된 line의 경우 탐지에 어려움이 있다
- 또다른 방법으로 **anchor 기반의 방법**이 연구되었지만 이 또한 미리 정의된 anchor로 인해 유연성이 떨어진다
  (대부분의 object detection도 anchor 기반의 방법으로 볼 수 있음)
  -> 복잡한 line이나 밀집된 line에서의 탐지 성능이 떨어진다



Dynamic Convolution or Conditional Convolution

> our core idea is that for an image with K instances, **K different mask heads will be dynamically generated**, and each mask head will contain the characteristics of its target instance in their filters. As a result, when the mask is applied to an input, **it will only fire on the pixels of the instance**, thus producing the mask prediction of the instance.

7. CondLaneNet 논문 리뷰

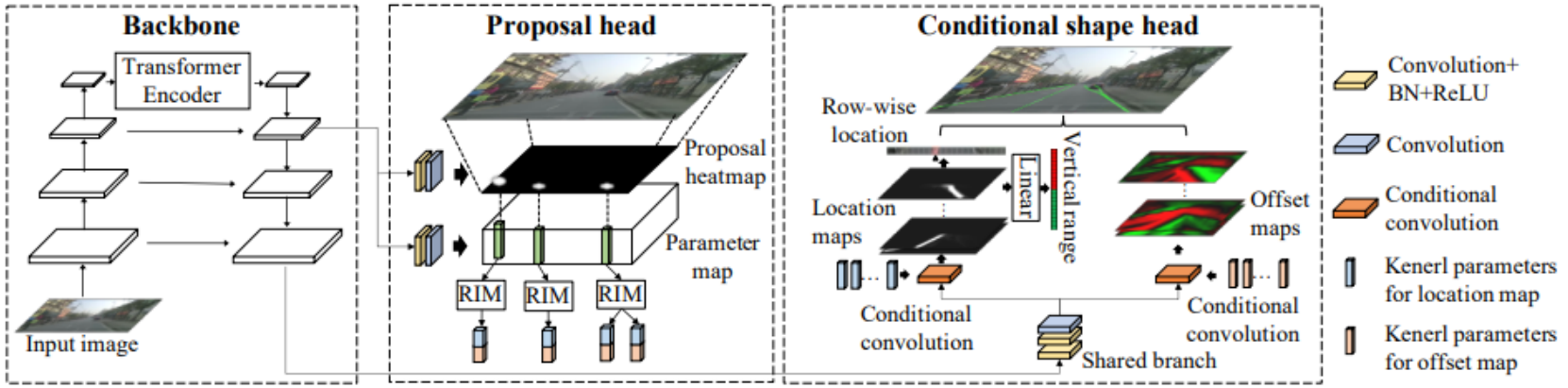
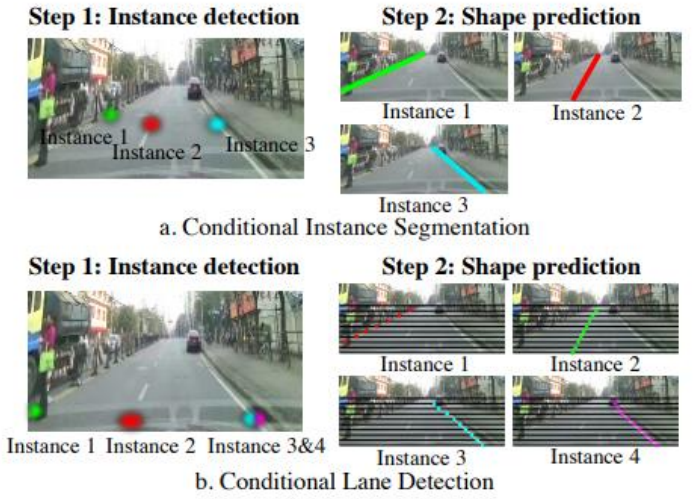


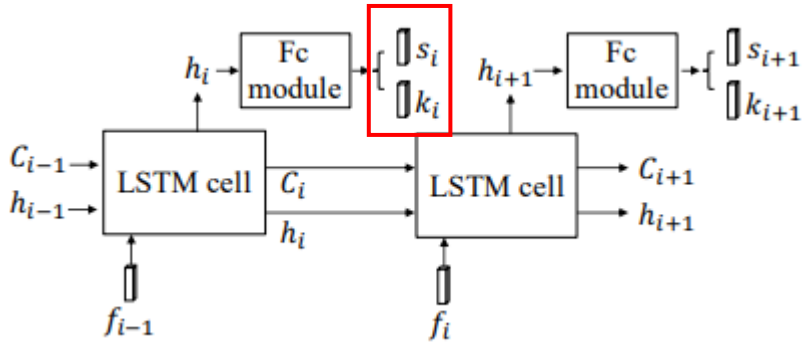
Figure 2. The structure of our CondLaneNet framework. The backbone adopts standard ResNet [8] and FPN [23] for multi-scale feature extraction. The transformer encoder module [37] is added for more efficient context feature extraction. The proposal head is responsible for detecting the proposal points which are located at the start point of the line. Meanwhile, a parameter map that contains the dynamic convolution kernels is predicted. The conditional shape head predicts the row-wise location, the vertical range, and the offset map to describe the shape for each line. To address the cases of dense lines and fork lines, the RIM is designed.

- 모델의 구조가 상당히 복잡한 편...
- 모델을 크게 세가지 모듈로 구성
  - **Backbone** – Resnet을 기본 인코더로 사용하고 FPN 구조를 사용하여 high level feature와 low level feature를 섞어 줌
    추가적으로 top stage feature map에 self-attention 기반의 transformer encoder 사용
    (transformer encoder에 대한 ablation study 결과도 논문에 있음)

7. CondLaneNet 논문 리뷰



- **Proposal head** – line의 시작 point를 instance-level로 proposal
  - RIM (recurrent instance model) – dense line, fork line, 가려진 line 을 잘 탐지하기 위해 이전 정보를 재귀적으로 사용
  - We define s_i as two-dimensional logits that indicate two states: **"continue" or "stop".** The vector k_i contains the **kernel parameters** for **subsequent instance-wise dynamic convolution**
- **Conditional shape head**
  - row-wise location, vertical range, offset map을 예측
  - 세 정보를 종합해 lane의 shape을 결정
    - Row-wise location, vertical range -> 좌표
    - Offset map -> 미세 조정
  - Row-wise formulation을 사용해 최종적으로 line의 shape을 mask 형태가 아니라 coordinate 형태로 묘사



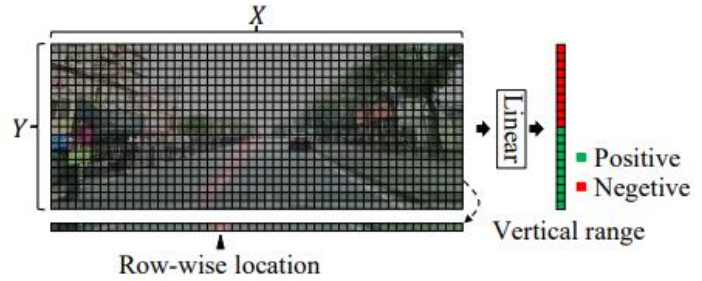


Figure 4. The process of parsing the row-wise location and the vertical range from the location map.