DLThon_RS7(CV)

Pascal VOC challenge

열정의백세시대

강다은 김찬중 조혜원

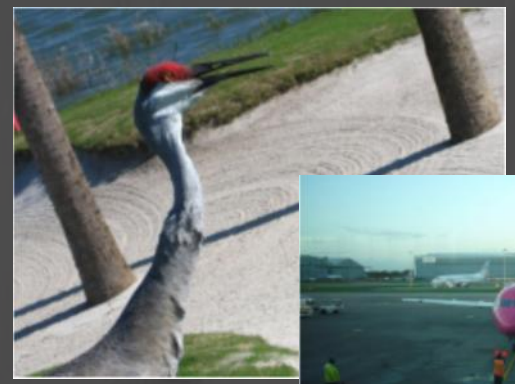
Part 1

문제 정의


Pascal VOC
DL models
심층 학습 기반
IMAGE SEGMENTATION
EDA
W&B
metrics
CAM

PASCAL VOC




일상 사진 2913장으로 이루어진 데이터셋.
일상에서 발견할 수 있는 오브젝트 20가지에 대하여
세그멘테이션 마스크를 정답 데이터로 제공.

목차

Part 1

수행 과정

Part 1

수행 과정 > 실험 계획 수립

학습 데이터

- 원본 데이터
- 증강 데이터
  - 좌우반전
  - 확대
  - 밝기/채도

×

학습 모델

- U-net
- FCN
- DeepLab
- R-CNN
- Panoptic

=

총 10 가지 실험

…을 계획하였으나
FCN, R-CNN
구현 실패

총 6가지 실험

Part 1

수행 과정 > 역할 분담

강다은

- 데이터 전처리 & 증강
- U-net 구현& 실험
- CAM 시각화

조혜원

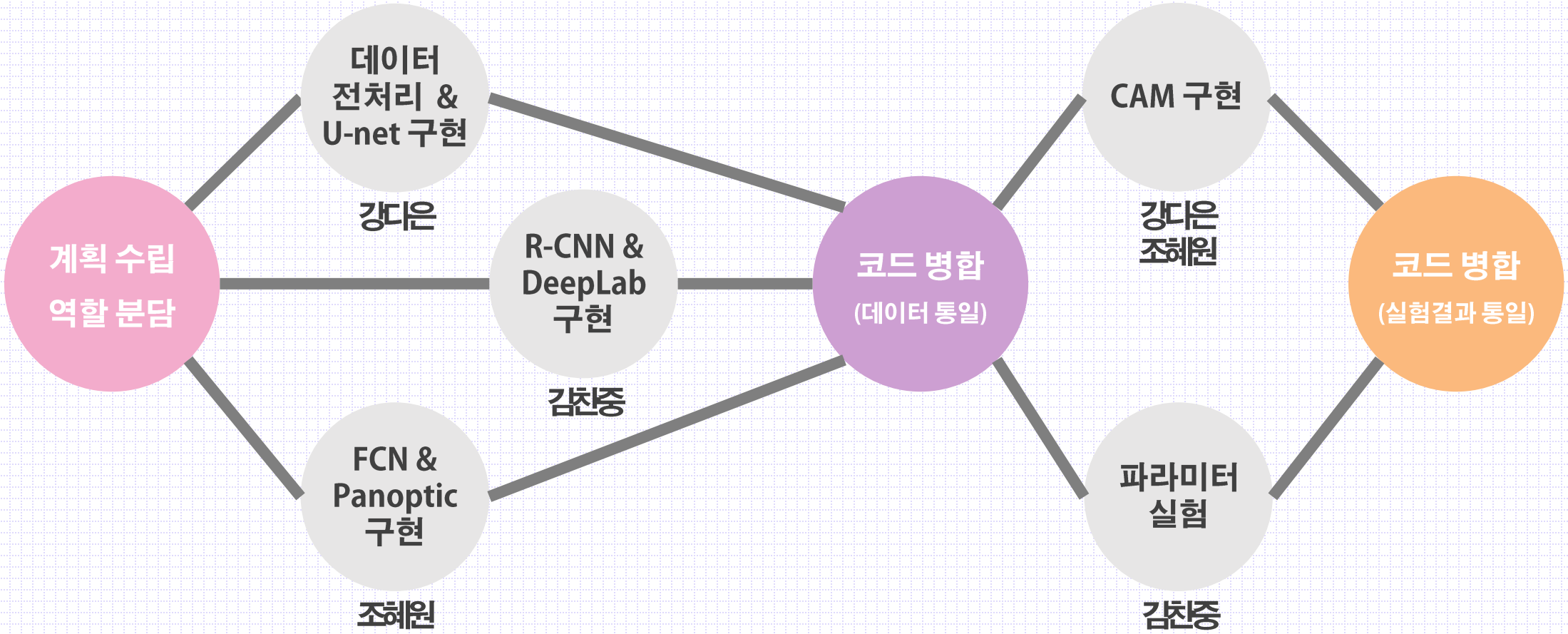
- FCN 구현
- Panoptic 구현&실험
- CAM 시각화

김찬중

- R-CNN 구현
- DeepLab구현 & 실험

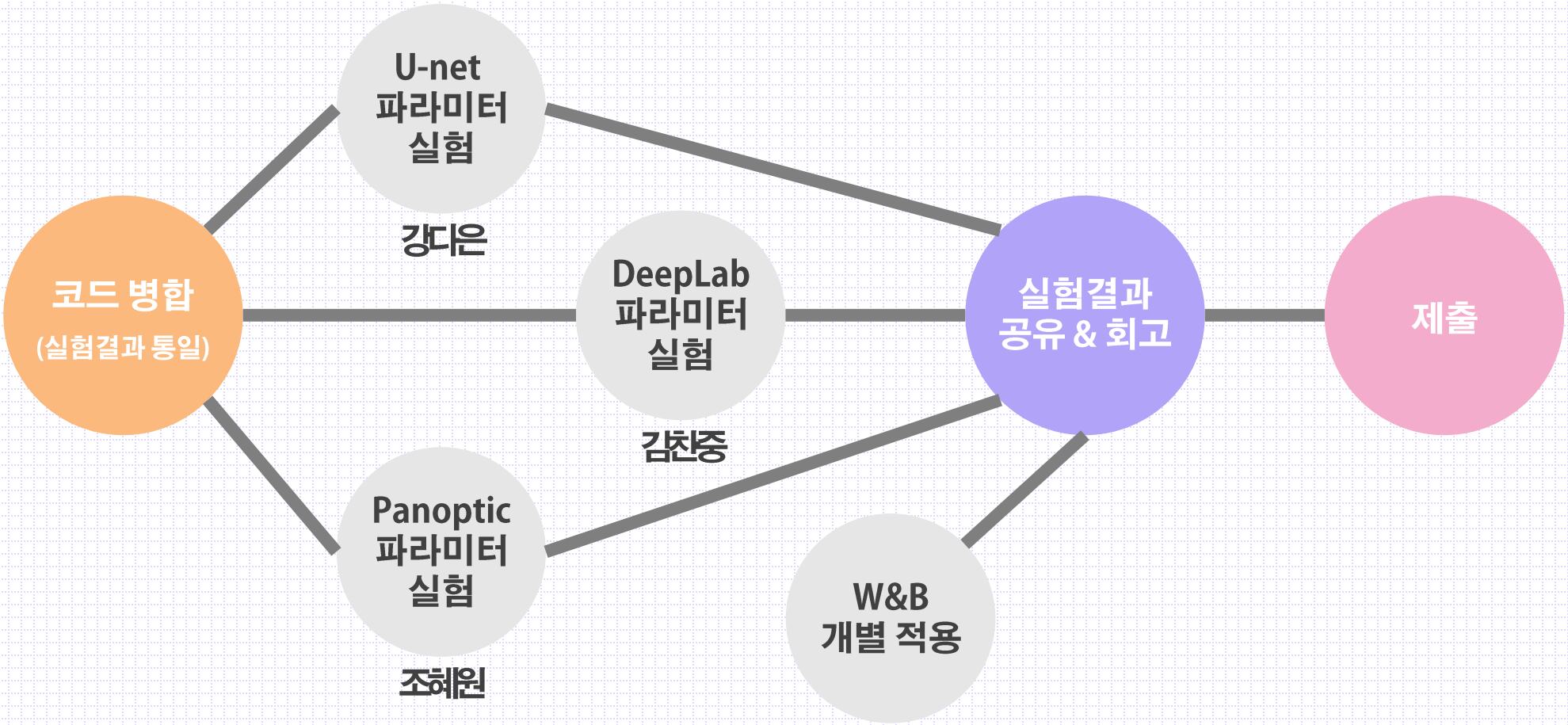
Part 1

수행 과정 > 진행 순서



Part 1

수행 과정 > 진행 순서



Part 2

수행 내용

Part 2

수행 내용 > 데이터 전처리 & 증강 (강다은)

```python
def do_vectorize(ds_target):
    return [np.vectorize(SEGMENT_TO_CLASS.get)(img) for img in ds_target]
```

```python
def get_img_from_bytes(img_bytes, is_input=True):
    pil_img = Image.open(io.BytesIO(img_bytes))
    tf_img = img_to_array(pil_img)

    if is_input == True: # (for input data)
        tf_img = tf_img / 255.
        return tf_img.astype("float32")

    else: # is_input == False: (for target data)
        tf_img = np.array(tf.image.rgb_to_grayscale(tf_img))
        return tf_img.astype("uint8")
```

데이터셋 로드

- 이미지 데이터가 bytes 형태로 저장되어 있으므로 PIL 이미지 및 np.array 형태로 변환
- 마스크 이미지 또한 RGB 형태로 저장되어 있으므로 (0,19) 범위의 정수 값을 갖는 grayscale로 변환

Part 2

수행 내용 > 데이터 전처리 & 증강 (강다은)

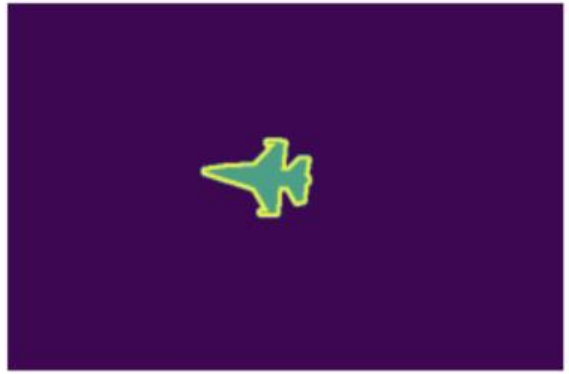
데이터 증강

- ImageDataGenerator 또는 albumentations를 사용하고 싶었으나, 원본 & 정답 데이터 쌍을 동일하게 변형하는 방법을 찾지 못함
- 데이터를 shuffle 하지 않고, 동일한 순서에 해당하는 원본 & 정답 데이터에 동일한 변환을 수행함으로써 수동으로 증강

- 원본 & 정답 데이터에 좌우반전, 확대 적용
- 정답 마스크에는 색상값이 중요한 의미를 가지므로, 밝기 및 채도 변환은 원본 이미지에만 적용

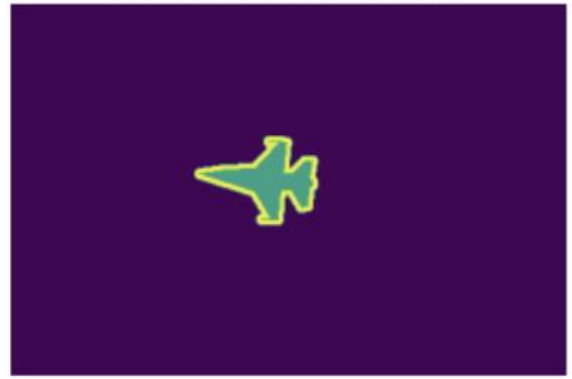
ORIGINAL INPUT (185)

ORIGINAL TARGET (185)



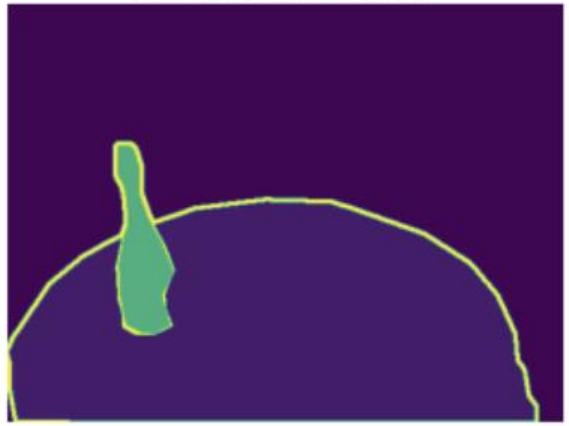
AUGMENTED INPUT (185)

AUGMENTED TARGET (185)



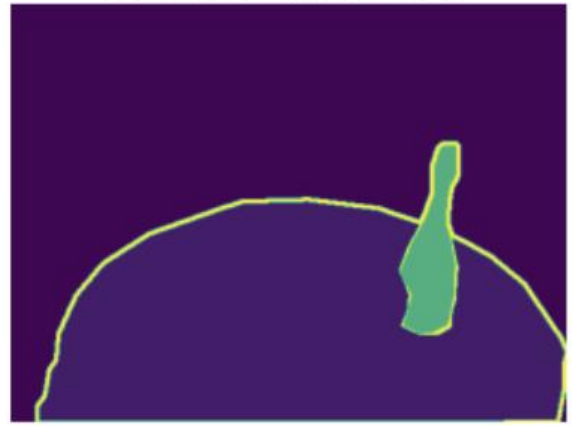
ORIGINAL INPUT (318)

ORIGINAL TARGET (318)



AUGMENTED INPUT (318)

AUGMENTED TARGET (318)



ORIGINAL INPUT (256)

ORIGINAL TARGET (256)

AUGMENTED INPUT (256)

AUGMENTED TARGET (256)

Part 2

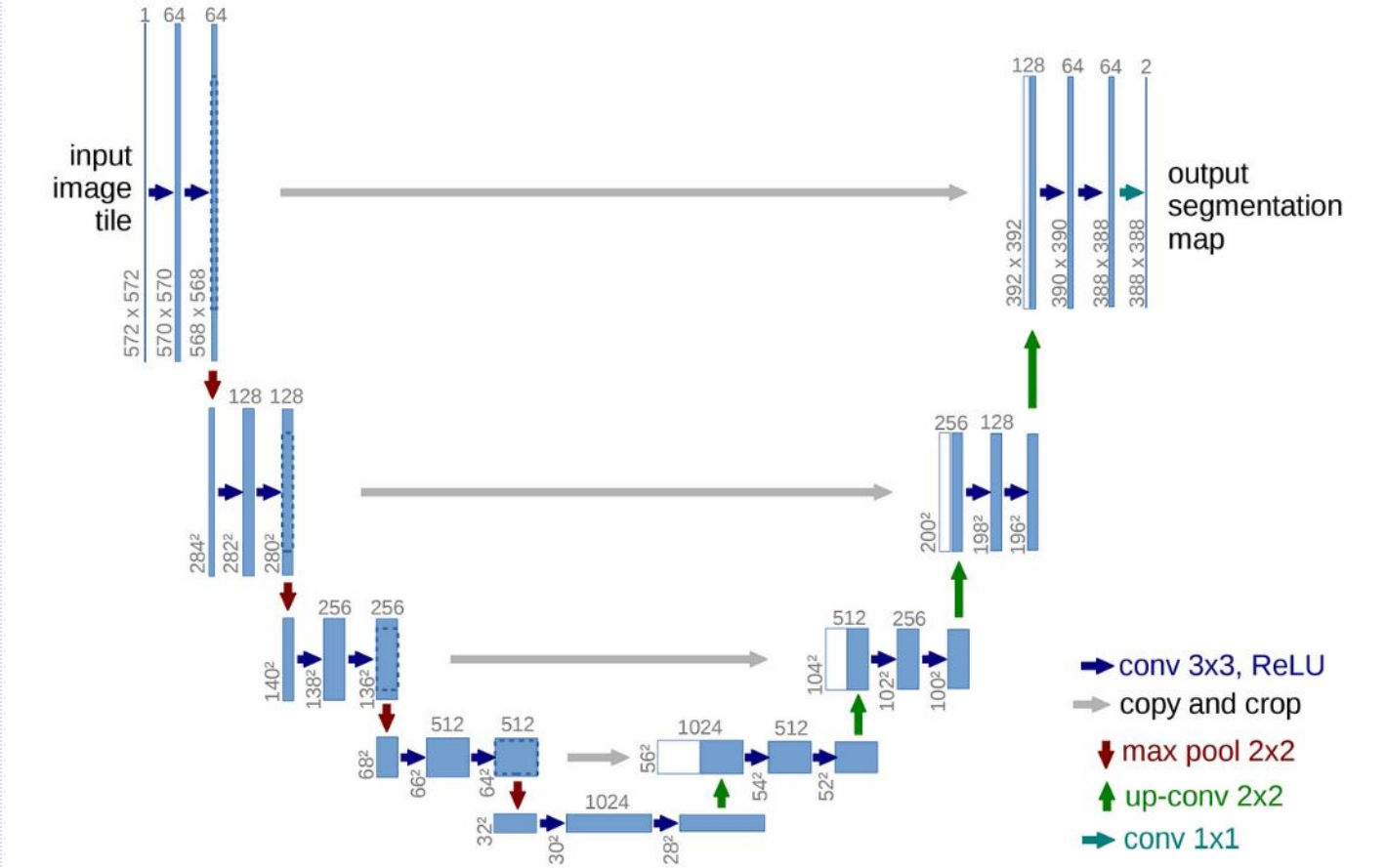
수행 내용 > 데이터 전처리 & 증강 (강다은)

데이터 전처리

- 원본 입력 이미지 (0,1) 범위의 부동소수로 정규화
- 원본 & 정답 모두 256*256 크기로 변환

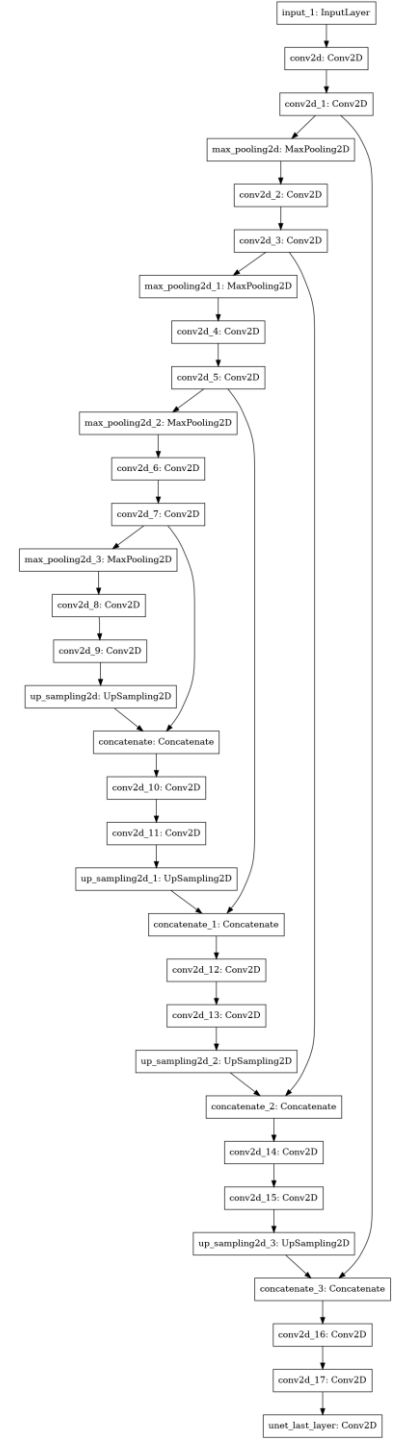
Part 2

수행 내용 > 모델 구성 > U-net (강다은)



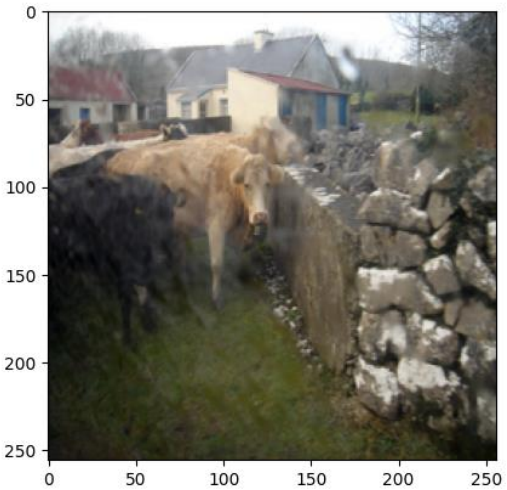
- 기존 퀘스트에서 구현한 코드를 재활용했으므로 특별한 어려움은 없었음
- 학습 시 loss = sparse categorical crossentropy 를 적용하기 위해 마지막 레이어의 activation 제거
- Gradient CAM 이미지 생성을 위해 마지막 레이어에 name = “unet_last_layer” 추가

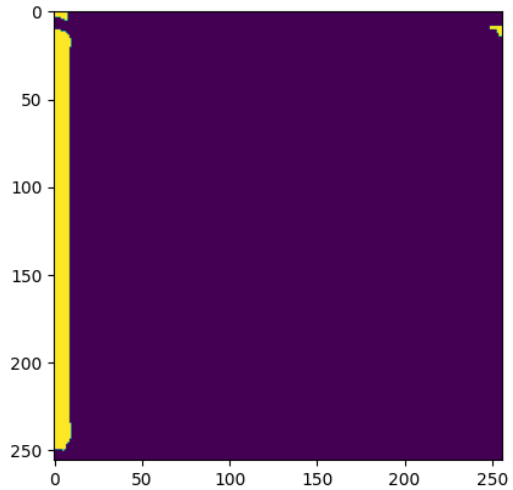
```
up_sampling2d (UpSampling2D)    (None, 32, 32, 1024) 0           conv2d_9[0][0]
_____
concatenate (Concatenate)       (None, 32, 32, 1536) 0           up_sampling2d[0][0]
                                                                 conv2d_7[0][0]
_____
conv2d_10 (Conv2D)              (None, 32, 32, 512)  7078400     concatenate[0][0]
_____
conv2d_11 (Conv2D)              (None, 32, 32, 512)  2359808     conv2d_10[0][0]
_____
up_sampling2d_1 (UpSampling2D)  (None, 64, 64, 512)  0           conv2d_11[0][0]
_____
concatenate_1 (Concatenate)     (None, 64, 64, 768)  0           up_sampling2d_1[0][0]
                                                                 conv2d_5[0][0]
_____
conv2d_12 (Conv2D)              (None, 64, 64, 256)  1769728     concatenate_1[0][0]
_____
conv2d_13 (Conv2D)              (None, 64, 64, 256)  590080      conv2d_12[0][0]
_____
up_sampling2d_2 (UpSampling2D)  (None, 128, 128, 256 0           conv2d_13[0][0]
_____
concatenate_2 (Concatenate)     (None, 128, 128, 384 0           up_sampling2d_2[0][0]
                                                                 conv2d_3[0][0]
_____
conv2d_14 (Conv2D)              (None, 128, 128, 128 442496      concatenate_2[0][0]
_____
conv2d_15 (Conv2D)              (None, 128, 128, 128 147584      conv2d_14[0][0]
_____
up_sampling2d_3 (UpSampling2D)  (None, 256, 256, 128 0           conv2d_15[0][0]
_____
concatenate_3 (Concatenate)     (None, 256, 256, 192 0           up_sampling2d_3[0][0]
                                                                 conv2d_1[0][0]
_____
conv2d_16 (Conv2D)              (None, 256, 256, 64) 110656      concatenate_3[0][0]
_____
conv2d_17 (Conv2D)              (None, 256, 256, 64) 36928       conv2d_16[0][0]
_____
unet_last_layer (Conv2D)        (None, 256, 256, 20) 1300        conv2d_17[0][0]
==================================================================================================
Total params: 31,380,180
Trainable params: 31,380,180
Non-trainable params: 0
__________________________________________________________________________________________________
```



Part 2

수행 내용 > 모델 구성 > FCN (조혜원)





Connection failed

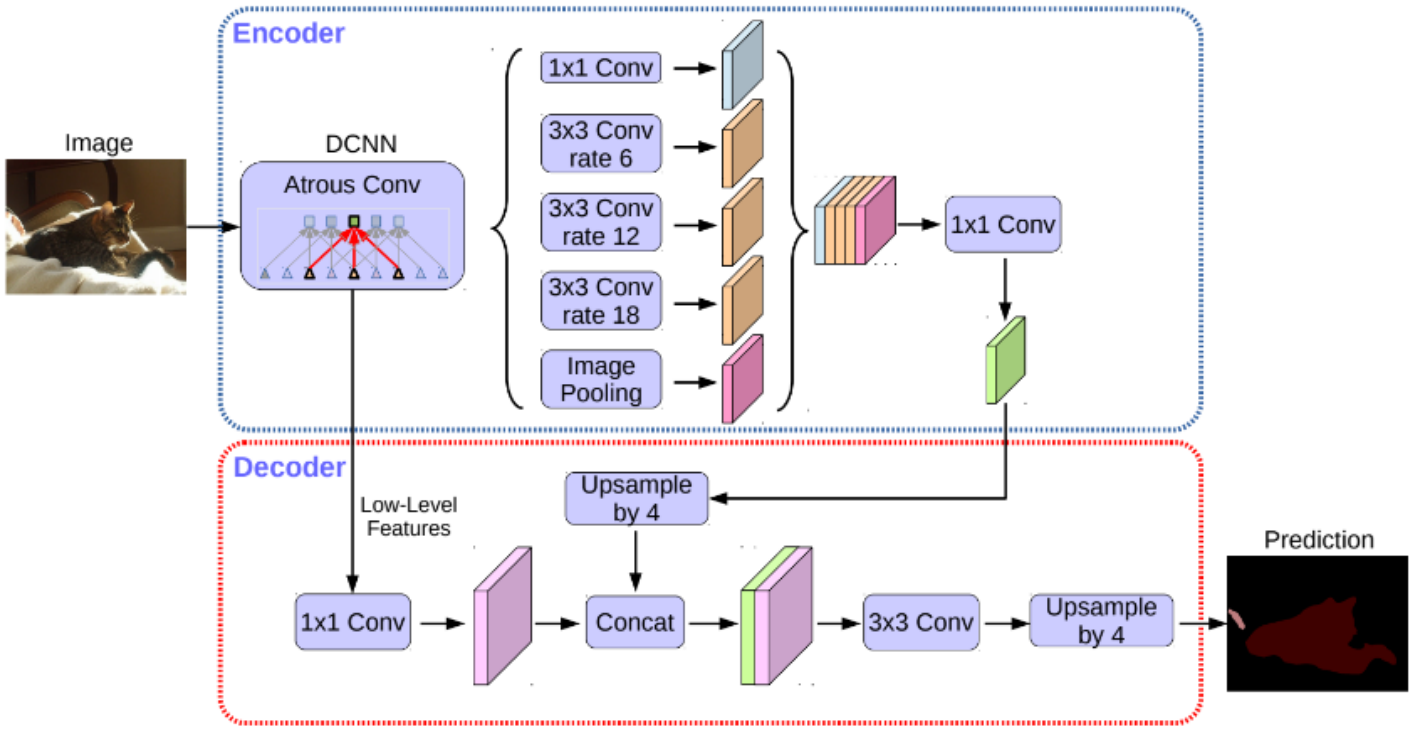
A connection to the notebook server could not be established. The notebook will continue trying to reconnect. Check your network connection or notebook server configuration.

OK

- FCN-32, FCN-16, FCN-8 등을 구현한 후, tf.keras.layers.Conv2D Transpose layer를 추가하고 cropping을 하여 upsampling을 구성
- 모델 학습 시 수시로 kernel dead 현상이 발생하고, 어쩌다 성공하더라도 유의미한 segmentation을 수행하지 못하는 것을 확인 → 시간 관계상 일단 포기

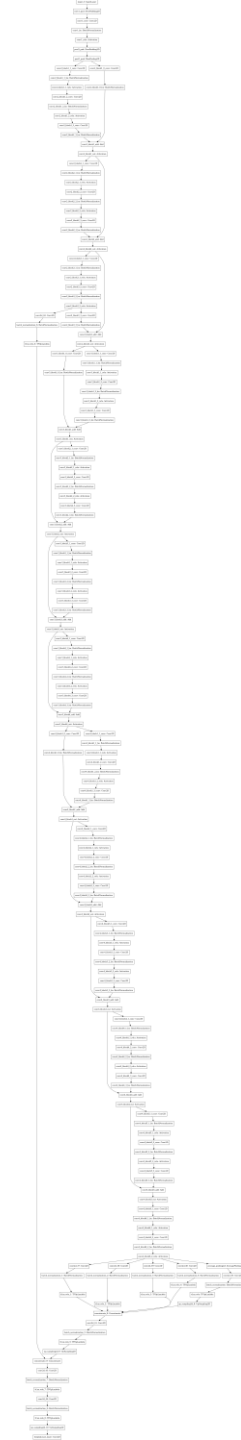
Part 2

수행 내용 > 모델 구성 > DeepLab (김찬중)



- DeepLabV3+ 버전 구현
- 사전 훈련된 ResNet50을 backbone으로 사용
- 학습 시 loss = sparse categorical crossentropy 를 적용하기 위해 마지막 레이어의 activation 제거
- Gradient CAM 이미지 생성을 위해 마지막 레이어에 name = “deeplab_last_layer” 추가

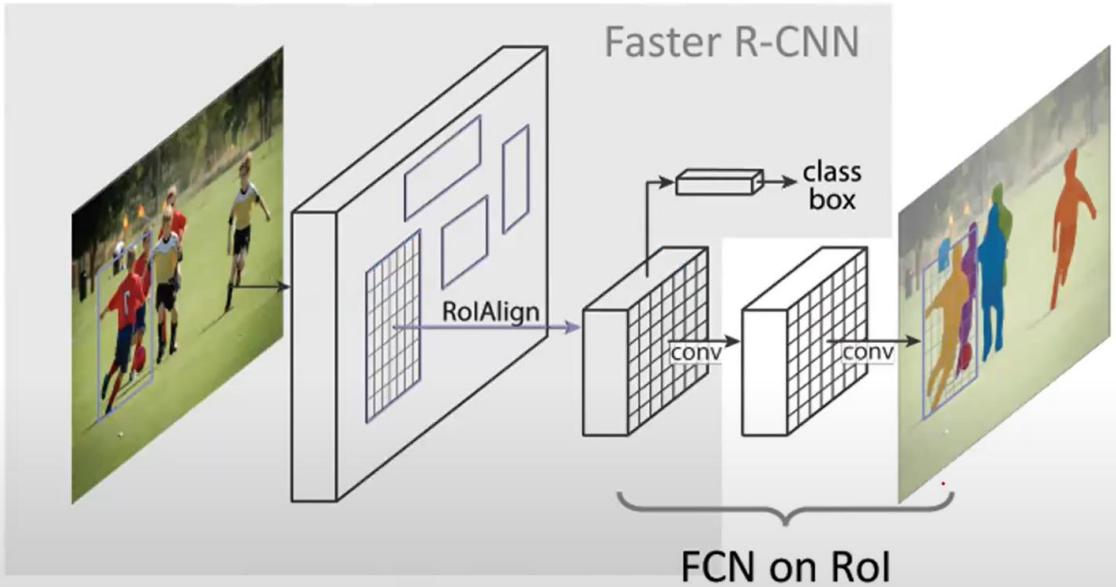
```
conv2d_41 (Conv2D)              (None, 16, 16, 256)  327680      concatenate_8[0][0]
batch_normalization_5 (BatchNor (None, 16, 16, 256)  1024        conv2d_41[0][0]
conv2d_42 (Conv2D)              (None, 64, 64, 48)   3072        conv2_block3_2_relu[0][0]
tf.nn.relu_5 (TFOpLambda)       (None, 16, 16, 256)  0           batch_normalization_5[0]
[0]
batch_normalization_6 (BatchNor (None, 64, 64, 48)   192         conv2d_42[0][0]
up_sampling2d_9 (UpSampling2D)  (None, 64, 64, 256)  0           tf.nn.relu_5[0][0]
tf.nn.relu_6 (TFOpLambda)       (None, 64, 64, 48)   0           batch_normalization_6[0]
[0]
concatenate_9 (Concatenate)     (None, 64, 64, 304)  0           up_sampling2d_9[0][0]
                                                                 tf.nn.relu_6[0][0]
conv2d_43 (Conv2D)              (None, 64, 64, 256)  700416      concatenate_9[0][0]
batch_normalization_7 (BatchNor (None, 64, 64, 256)  1024        conv2d_43[0][0]
tf.nn.relu_7 (TFOpLambda)       (None, 64, 64, 256)  0           batch_normalization_7[0]
[0]
conv2d_44 (Conv2D)              (None, 64, 64, 256)  589824      tf.nn.relu_7[0][0]
batch_normalization_8 (BatchNor (None, 64, 64, 256)  1024        conv2d_44[0][0]
tf.nn.relu_8 (TFOpLambda)       (None, 64, 64, 256)  0           batch_normalization_8[0]
[0]
up_sampling2d_10 (UpSampling2D) (None, 256, 256, 256 0           tf.nn.relu_8[0][0]
deeplab_last_layer (Conv2D)     (None, 256, 256, 20) 5140        up_sampling2d_10[0][0]
==================================================================================================
=======
Total params: 11,857,236
Trainable params: 11,824,500
Non-trainable params: 32,736
__________________________________________________________________________________________________
```



Part 2

수행 내용 > 모델 구성 > R-CNN (김찬중)

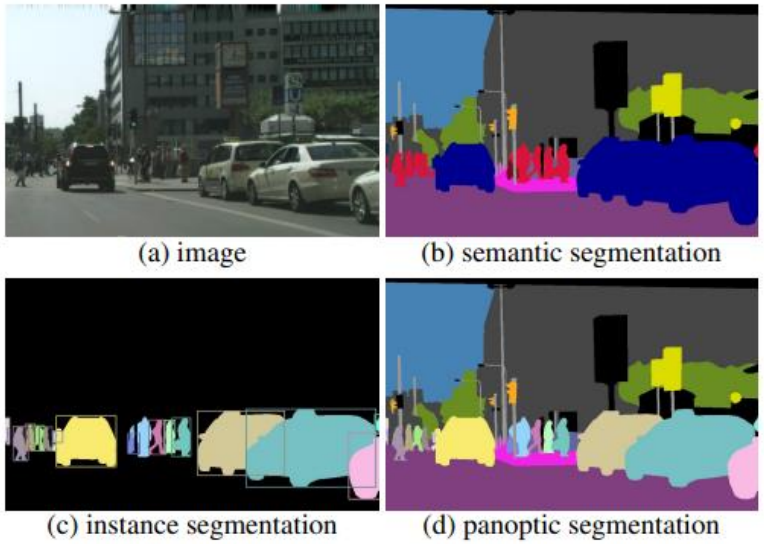
- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs



- segmentation에 효과적이라고 알려진 Faster R-CNN과 FCN을 결합한 MASK R-CNN을 구현 시도
- 모델 구조가 복잡하여 직접 구현에 실패
- FAIR에서 제공하는 Detectron 모델을 설치하는 방법을 시도 → Detectron 설치에 실패하여 일단 포기

Part 2

수행 내용 > 모델 구성 > Panoptic (조혜원)



(a) image (b) semantic segmentation

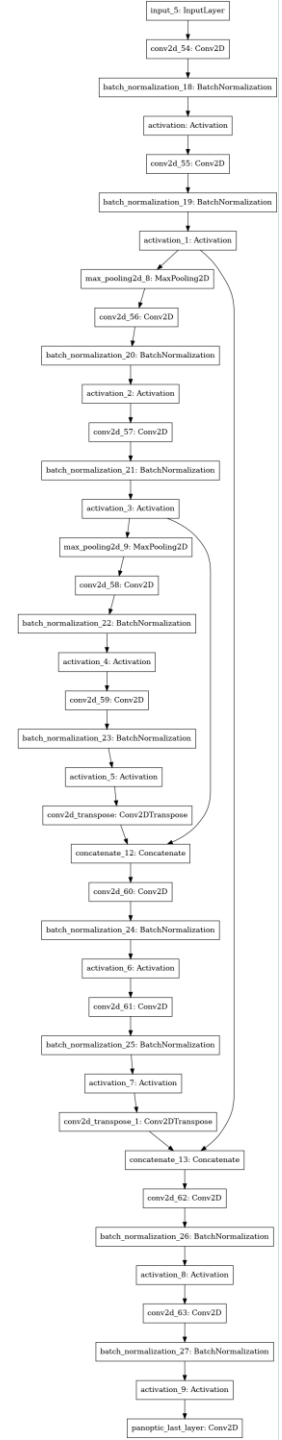
(c) instance segmentation (d) panoptic segmentation

Figure 1: For a given (a) image, we show *ground truth* for: (b) semantic segmentation (per-pixel class labels), (c) instance segmentation (per-object mask and class label), and (d) the proposed *panoptic segmentation* task (per-pixel class+instance labels). The PS task: (1) encompasses both stuff and thing classes, (2) uses a simple but general format, and (3) introduces a uniform evaluation metric for all classes. Panoptic segmentation generalizes both semantic and instance segmentation and we expect the unified task will present novel challenges and enable innovative new methods.

- semantic & instance 모두 수행 가능한 모델
- instance의 경우 instance 개수를 지정해야 하는데 PASCAL VOC에 곧바로 적용하기 어렵다고 판단하여 semantic 선택
- 마지막 레이어의 activation 제거
- 마지막 레이어에 name = “deeplab_last_layer” 추가

concatenate_12 (Concatenate)	(None, 128, 128, 256	0	conv2d_transpose[0][0] activation_3[0][0]
conv2d_60 (Conv2D)	(None, 128, 128, 128	295040	concatenate_12[0][0]
batch_normalization_24 (BatchNo	(None, 128, 128, 128	512	conv2d_60[0][0]
activation_6 (Activation)	(None, 128, 128, 128	0	batch_normalization_24[0][0]
conv2d_61 (Conv2D)	(None, 128, 128, 128	147584	activation_6[0][0]
batch_normalization_25 (BatchNo	(None, 128, 128, 128	512	conv2d_61[0][0]
activation_7 (Activation)	(None, 128, 128, 128	0	batch_normalization_25[0][0]
conv2d_transpose_1 (Conv2DTrans	(None, 256, 256, 64)	32832	activation_7[0][0]
concatenate_13 (Concatenate)	(None, 256, 256, 128	0	conv2d_transpose_1[0][0] activation_1[0][0]
conv2d_62 (Conv2D)	(None, 256, 256, 64)	73792	concatenate_13[0][0]
batch_normalization_26 (BatchNo	(None, 256, 256, 64)	256	conv2d_62[0][0]
activation_8 (Activation)	(None, 256, 256, 64)	0	batch_normalization_26[0][0]
conv2d_63 (Conv2D)	(None, 256, 256, 64)	36928	activation_8[0][0]
batch_normalization_27 (BatchNo	(None, 256, 256, 64)	256	conv2d_63[0][0]
activation_9 (Activation)	(None, 256, 256, 64)	0	batch_normalization_27[0][0]
panoptic_last_layer (Conv2D)	(None, 256, 256, 20)	1300	activation_9[0][0]

Total params: 1,869,204
Trainable params: 1,866,644
Non-trainable params: 2,560



Part 2

수행 내용 > 손실함수 & 평가지표

```python
def dice_coefficient(target, prediction, smooth=1):
    intersection = tf.reduce_sum(target * prediction)
    union = tf.reduce_sum(target) + tf.reduce_sum(prediction)
    dice = (2. * intersection + smooth) / (union + smooth)

    return dice
```

```python
def pixel_accuracy(target, prediction):
    # 정수로 변환하여 비교
    y_true = tf.cast(y_true * 255, tf.int32)
    y_pred = tf.cast(tf.round(y_pred * 255), tf.int32)

    # 정확하게 예측된 픽셀의 개수 계산
    correct_pixels = tf.reduce_sum(tf.cast(tf.equal(y_true, y_pred), tf.float32))

    # 전체 픽셀의 개수 계산
    total_pixels = tf.reduce_sum(tf.ones_like(y_true, dtype=tf.float32))

    # 픽셀 정확도 계산
    accuracy = correct_pixels / total_pixels

    return accuracy.numpy()
```

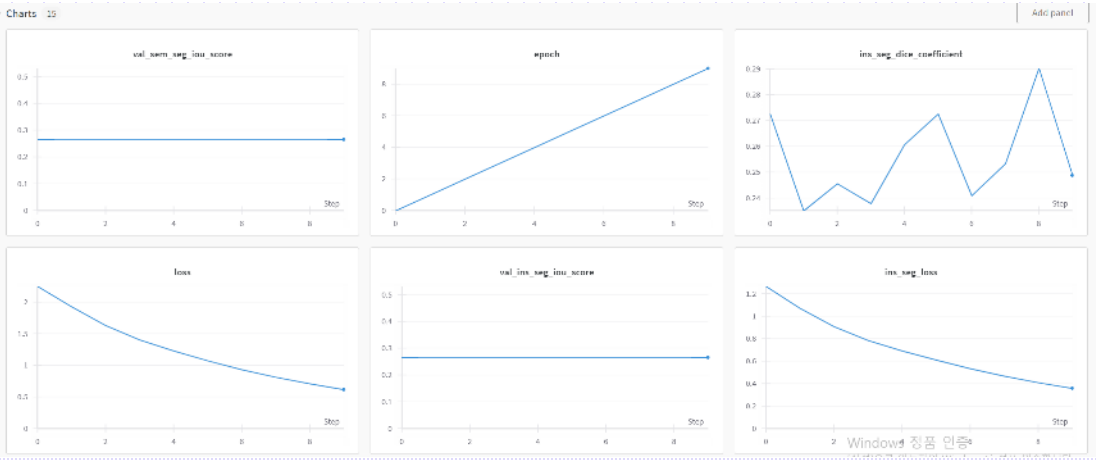
```python
def iou_score(target, prediction):
    target = tf.convert_to_tensor(target, dtype=tf.float32)
    prediction = tf.convert_to_tensor(prediction, dtype=tf.float32)

    intersection = tf.reduce_sum(tf.cast(tf.logical_and(tf.cast(target, tf.bool), tf.cast(pr
    union = tf.reduce_sum(tf.cast(tf.logical_or(tf.cast(target, tf.bool), tf.cast(prediction

    iou_score = tf.math.divide_no_nan(intersection, union)

    return iou_score
```

- 노드에서 학습한대로 dice coefficient, pixel accuracy, iou를 사용하고자 함
- pixel accuracy는 데이터 타입 불일치 문제로 작동 오류 → 시간 관계상 일단 포기
- dice coefficient, iou는 모델 학습 시 epoch에 따른 수치 변화가 전혀 없었기 때문에 무의미하거나 구현에 실패했다고 판단 → 시간 관계상 일단 포기
- 임시 metrics로 accuracy 사용

Part 2

수행 내용 > W&B

lucky-sweep-5	aiffel_dlthon_test	Crashed
vital-sweep-4	aiffel_dlthon_test	Crashed
desert-sweep-3	aiffel_dlthon_test	Crashed
electric-sweep-2	aiffel_dlthon_test	Crashed
silvery-sweep-1	aiffel_dlthon_test	Crashed
azure-jazz-4	quest-RS7_course_09_dlthon_cv_aiffel_ques...	Failed
light-meadow-3	quest-RS7_course_09_dlthon_cv_aiffel_ques...	Finished



- W&B를 사용해보려고 했으나 수시로 failed, crashed 등의 오류 발생
  → 가끔 성공하는 사례도 있었으나 반복적인 실험을 관리하기에는 어렵다고 판단

- 시간 제약으로 인해 사용법을 배웠다는 데에 의의를 두고 과감히 포기
  → LMS 상에서 모델을 학습하고 history를 시각화 하는 방식으로 실험 진행

Part 2

수행 내용 > CAM (강다은, 조혜원)

```python
def get_cam_img_from_model(model, activation_layer, ds_test):

    def get_cam_img(test_img):
        # convert test image into Tensorflow
        img_tensor = tf.convert_to_tensor(test_img, dtype=tf.float32)

        # get cam model
        cam_model = tf.keras.models.Model([model.inputs], [model.get_layer(activation_layer)

        with tf.GradientTape() as tape:
            conv_output, pred = cam_model(tf.expand_dims(img_tensor, 0))
            class_idx = tf.argmax(pred[0], axis=-1)    # get the index of the predicted clas
            one_hot = tf.one_hot(class_idx, depth=20)  # convert class index into one-hot en
            loss = tf.reduce_sum(one_hot * pred[0])
            output = conv_output[0] # 관심있는 layer의 output을 얻습니다.
            grad_val = tape.gradient(loss, conv_output)[0] # 예측값에 따른 Layer의 gradient를 얻습

        # generate CAM image
        weights = np.mean(grad_val, axis=(0, 1))
        cam_img = np.dot(conv_output.numpy(), weights)
        cam_img = (cam_img - np.min(cam_img)) / (np.max(cam_img) - np.min(cam_img)) # normal

        # convert CAM image into 3-RGB-chennel
        cam_img_3ch = np.stack([cam_img]*3, axis=-1).astype(np.float32)
        cam_img_3ch = np.squeeze(cam_img_3ch)

        # blend cam image with original test image
        result_img = cv2.addWeighted(test_img, 0.5, cam_img_3ch, 0.5, 0.0)

        return result_img

    return [get_cam_img(img) for img in ds_test]
```

- 노드에서 배운 gradient CAM 코드를 재활용
- 기존 코드는 이미지 분류를 위한 것으로, segmentation을 수행하기 위하여 일부 코드를 수정함
- Pascal VOC 이미지 데이터와 호환되도록 데이터 형상(shape)을 통일하는 코드를 추가함

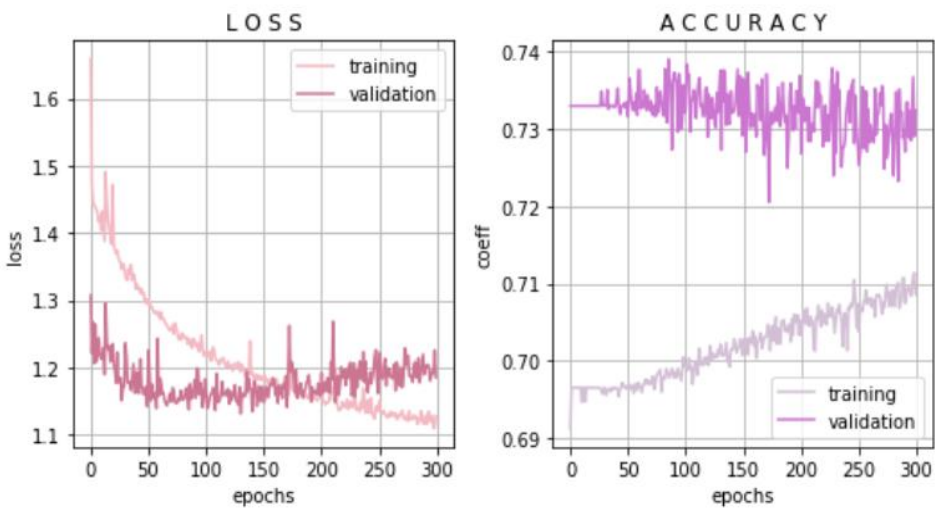
Part 3

수행 결과

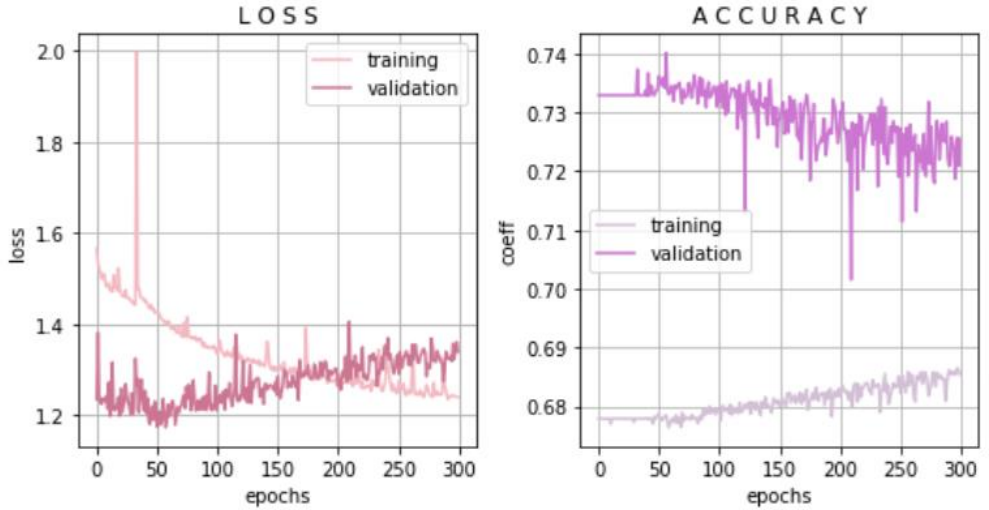
Part 3

수행 결과 > U-net 학습 결과

>>> U-net model training evaluation (original data)



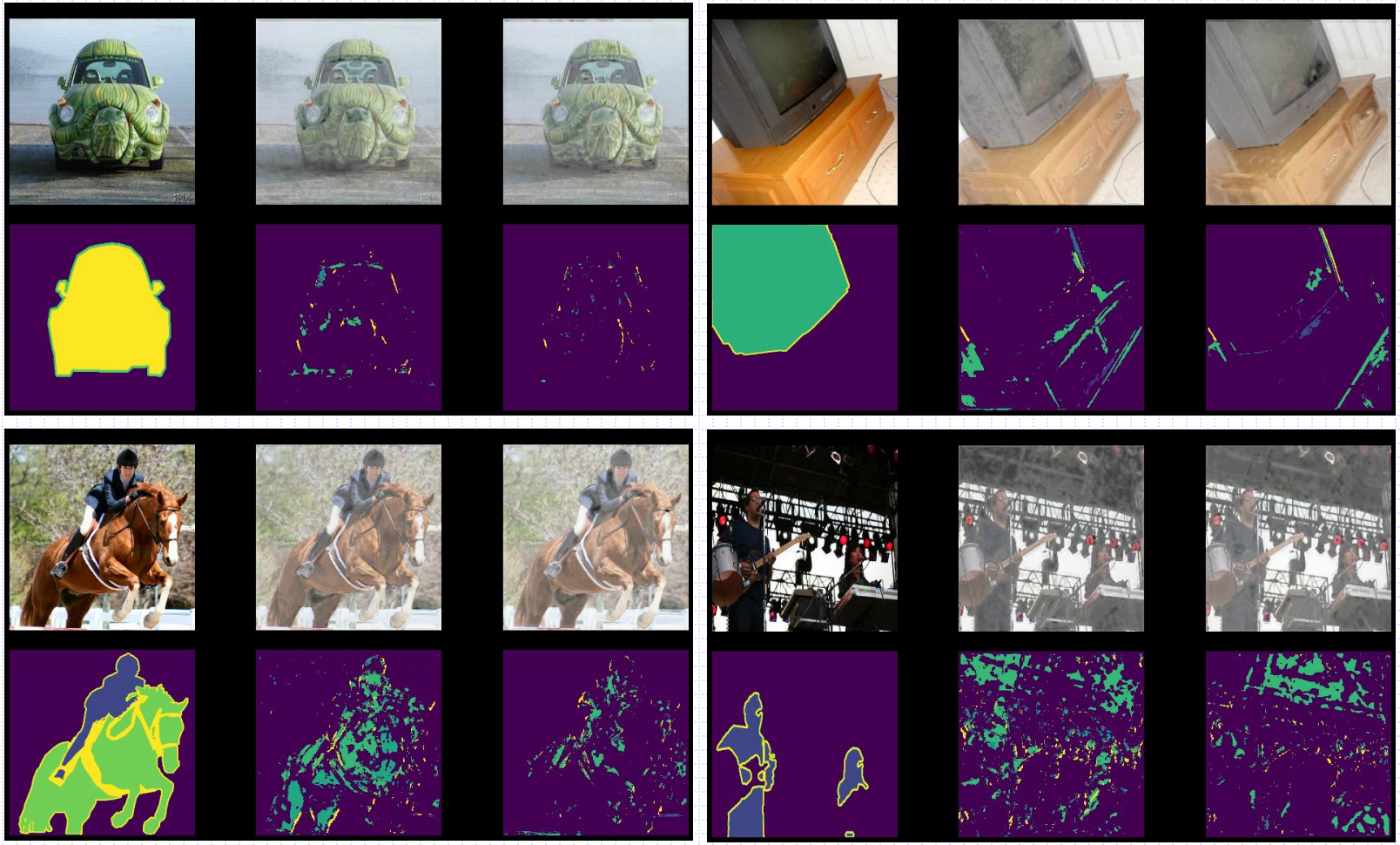
(augmented data)



- Training metrics는 꾸준히 변화하나, validation metrics는 거의 변화가 없음
- 증강 데이터의 경우 원본 데이터에 비해 training metrics의 변화 폭이 안정적임

Part 3

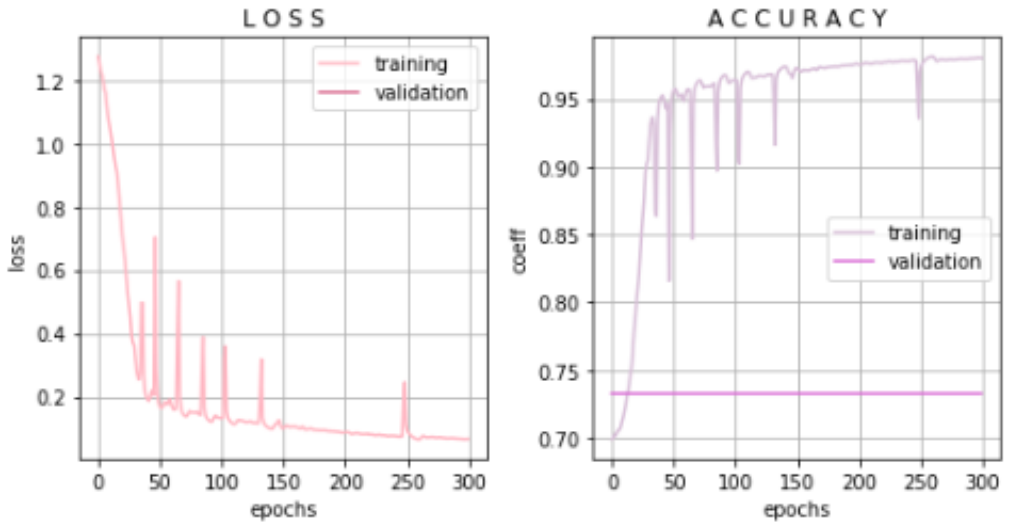
수행 결과 > U-net 실험 결과



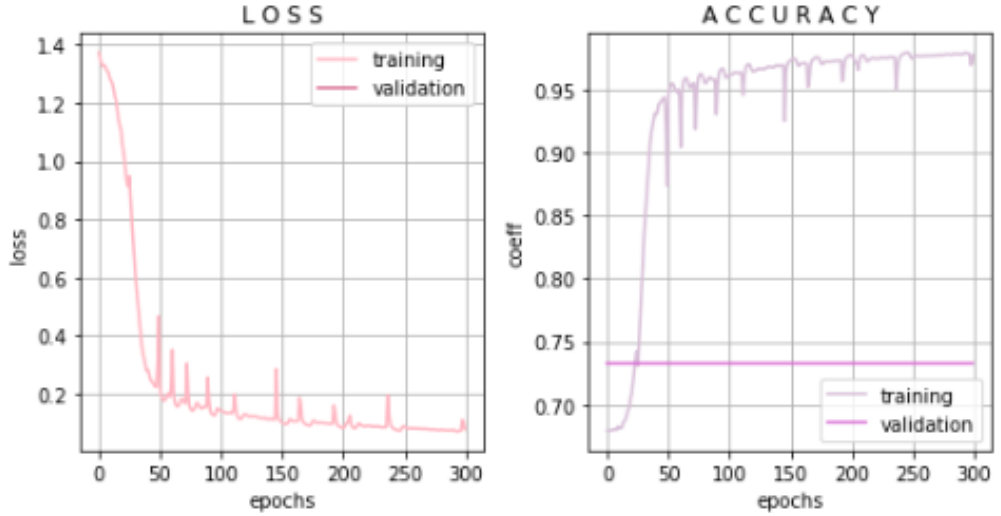
Part 3

수행 결과 > DeepLab 학습 결과

>>> DeepLab model training evaluation (original data) :



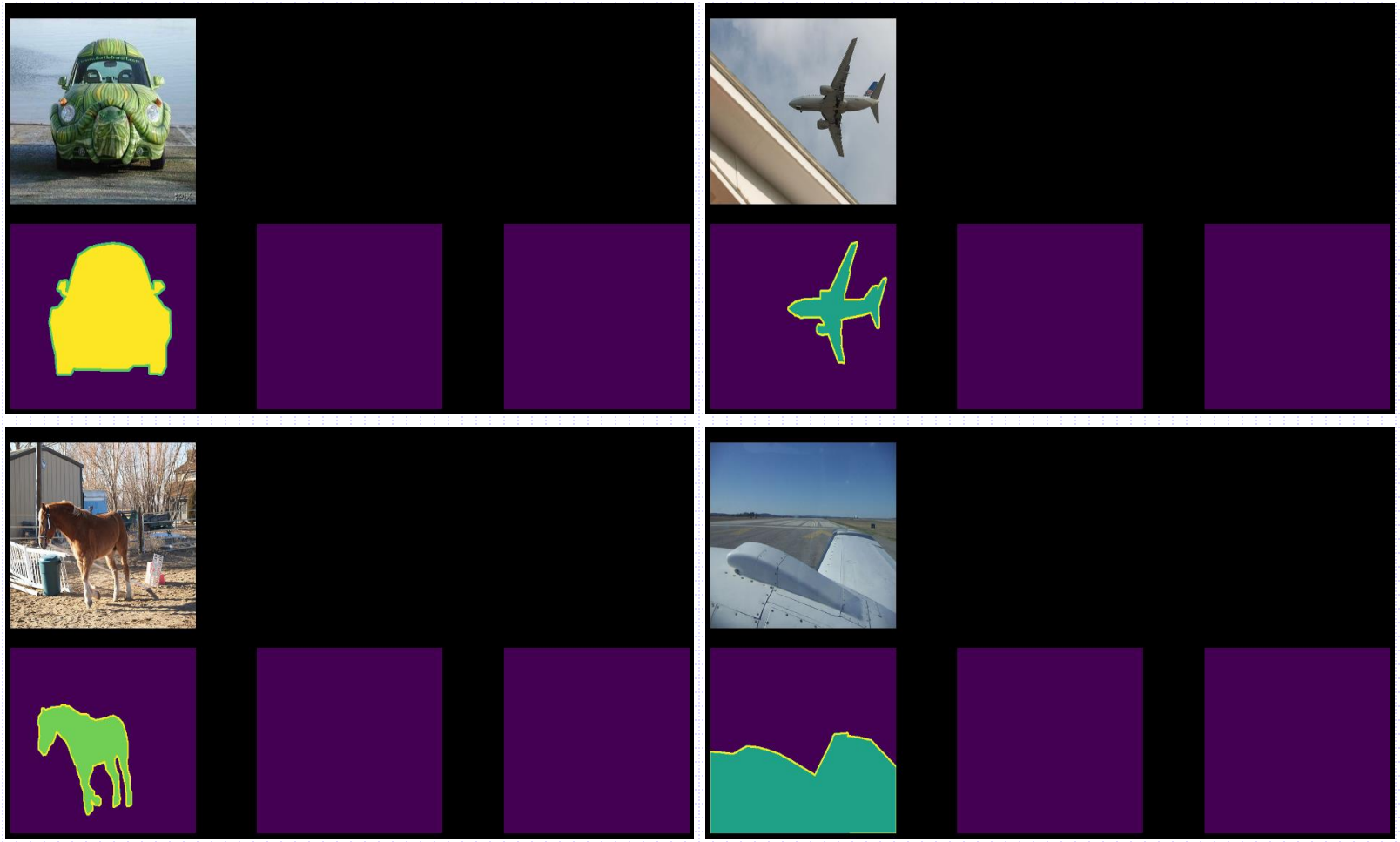
>>> DeepLab model training evaluation (augmented data) :



- Training metrics는 꾸준히 변화하나, validation metrics는 거의 변화가 없음
- 증강 데이터의 경우 원본 데이터에 비해 training metrics의 변화 폭이 안정적임

Part 3

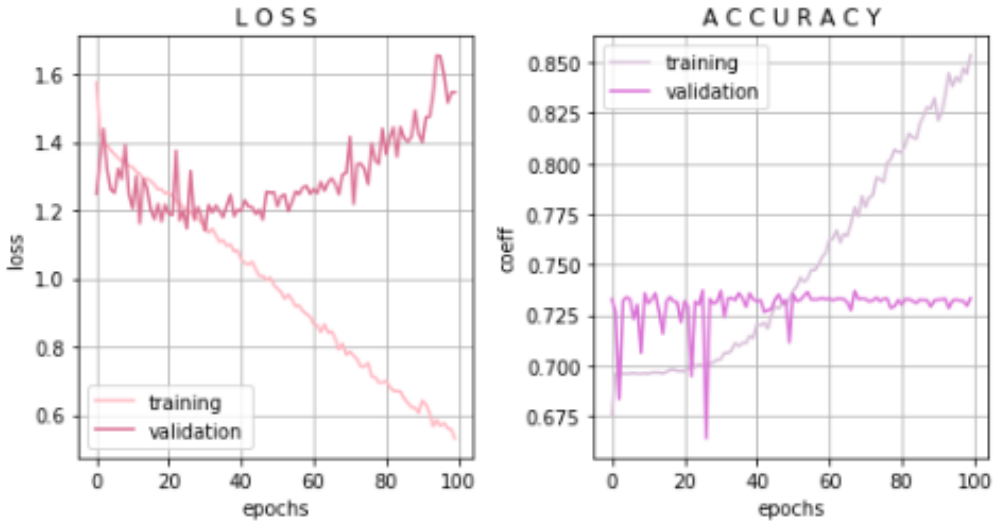
수행 결과 > DeepLab 실험 결과



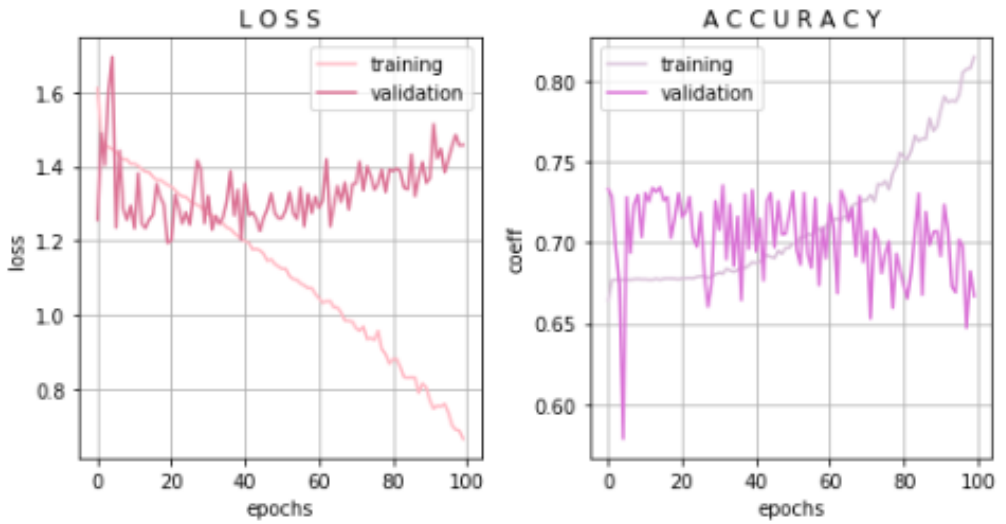
Part 3

수행 결과 > Panoptic 학습 결과

>>> Panoptic model training evaluation (original data) :



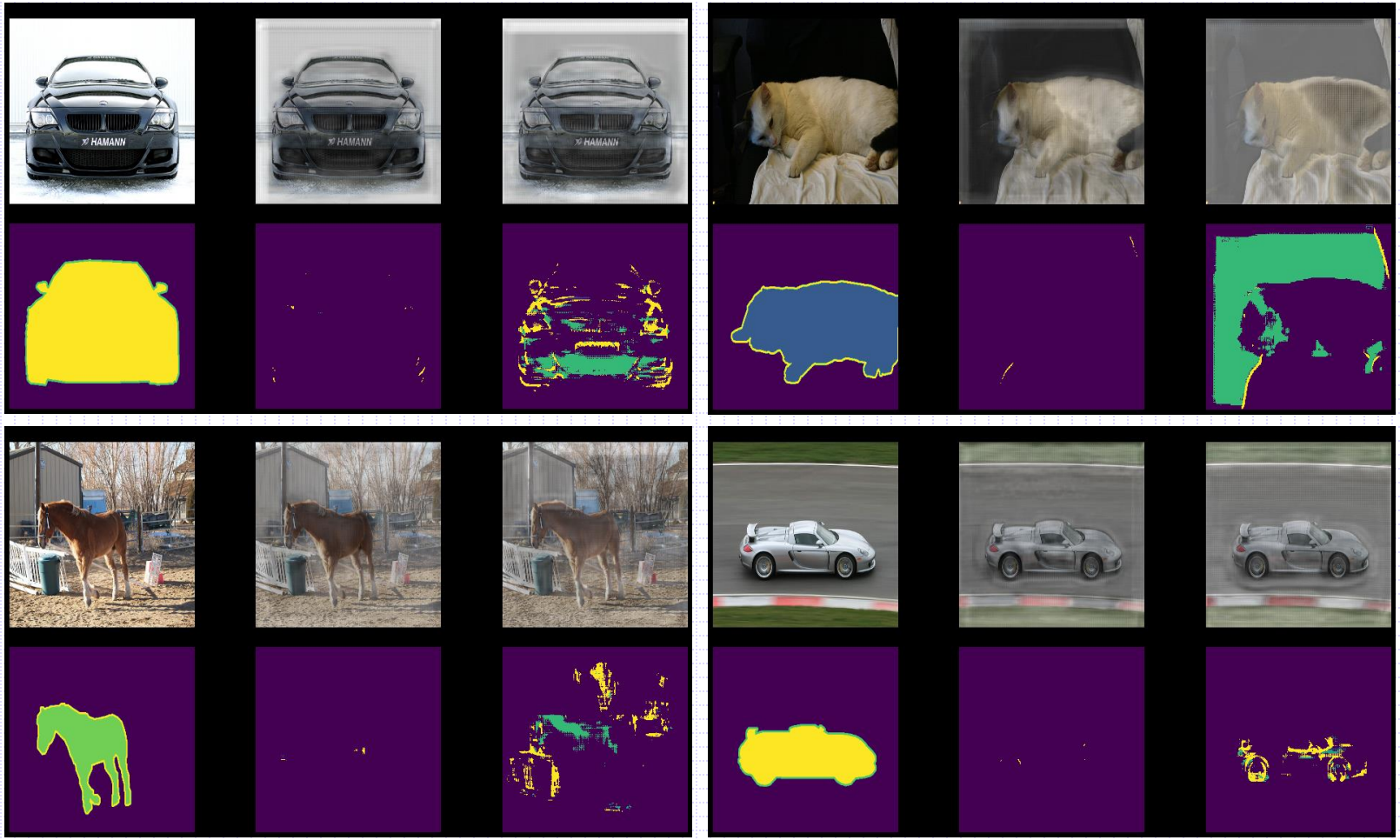
>>> Panoptic model training evaluation (augmented data) :



- Training metrics는 급격하게 변화하는 데 비해, validation metrics는 변화폭이 크지 않음
- 증강 데이터의 경우 원본 데이터에 비해 변화 폭이 큼 / epochs가 거듭될수록 성능이 꾸준히 악화됨

Part 3

수행 결과 > Panoptic 실험 결과



Part 4

최종 회고

Part 4

최종 회고 > 모델 성능 비교

U-net (강다은)

- 사물의 윤곽선에 대한 세그멘테이션을 수행하는 것으로 보인다
- EPOCHS=300으로 수행하였는데, 원만한 세그멘테이션을 위하서는 방대한 학습량이 필요할 것 것으로 추정

DeepLab (김찬중)

- EPOCHS=300으로 학습하였는데도 불구하고, 세그멘테이션을 아예 수행하지 않는 것으로 추정
- 모델 깊이에 비해 learning rate가 낮은 것이 원인이 아닌지 추정

Panoptic (조혜원)

- 사물의 윤곽선에 대한 세그멘테이션을 수행하는 것으로 보인다
- 시간관계상 EPOCHS=100으로 실험하였는데, 다른 모델과 동일하게 300으로 실험했다면 훨씬 나은 결과를 기대할 수 있다

Part 4

최종 회고 > 실험 의의

배운점

- 어떤 모델을 실험할지 계획을 수립하는 단계에서, 다양한 모델 및 방법론에 대해 조사하는 계기가 되었다
- 원핫인코딩 마스크와 정수형 마스크의 차이를 다시금 깨닫고, 마스크 종류에 따른 activation 및 loss 함수의 사용법도 익혔다
- 조원들과 역할을 분담하여 개별로 작업을 수행하는 과정과 함께 협동하여 작업을 수행하는 과정 간의 균형을 잡으며 원만한 팀플을 진행하였다
- 주어진 시간 내에서 페이스 조절을 통해 (비록 실험 결과는 좋지 않았지만) 제출물을 완성할 수 있었다

Part 4

최종 회고 > 실험 한계

아쉬운 점

- 커널/메모리 문제로 자유로운 모델 학습 실험이 불가능하였다
- 시간 관계상 코드를 직접 구현하기 보다는 원리나 구조를 이해하지 못한 채 외부 코드를 적극 참고하였다
- 실험 계획은 원대하게 세웠으나, 시간 관계상 구현 도중 오류가 생기는 것은 과감히 포기해버렸다
- 특히 metrics는 dice coefficient, pixel accuracy, IOU 모두 보기하고 accuracy만 적용하여 모델 학습을 진행하였는데, 과연 적합한 metrics였는지 결과물이 썩 만족스럽지 못한 것은 metrics 때문이 아닌지 의문이 든다

Part 4

최종 회고 > 실험 한계

의문점

- DeepLab 모델의 경우 segmentation 결과를 아예 확인할 수 없었는데, 다른 데이터셋(instace level human parsing)을 이용하여 실험을 수행할 경우 유의미한 결과를 확인할 수 있었다

Q&A