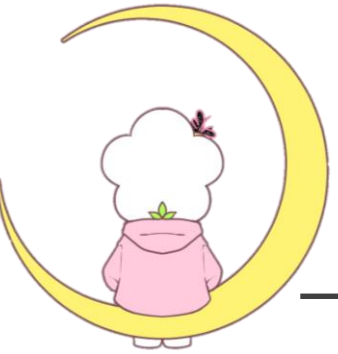
Yolo v1 최종점검

이다희, 박유림



CONTENTS

- VOC 2007 Dataset
- Unified Detection
- Modeling
- Loss Function
- Training
- Loss Evaluation
- Final Output
- Limitations
- Plans

VOC 2007 Dataset

VOC 2007 Dataset 이용

The PASCAL Visual Object Classes Challenge 2007

[click on an image to see the annotation]

-Person : person
-Animal: bird, cat, cow, dog, horse, sheep
-Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
-Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

VOC 2007 Dataset

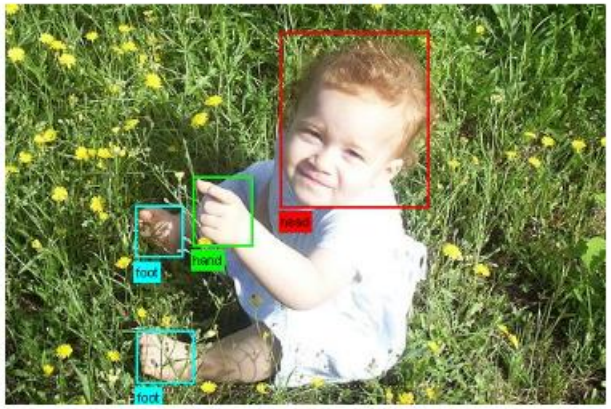
Image

Objects

Class

Image

Person Layout



VOC 2007 Dataset

```
<object>
        <name>chair</name>
        <pose>Rear</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
                <xmin>263</xmin>
                <ymin>211</ymin>
                <xmax>324</xmax>
                <ymax>339</ymax>
        </bndbox>
</object>
<object>
        <name>chair</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
                <xmin>165</xmin>
                <ymin>264</ymin>
                <xmax>253</xmax>
                <ymax>372</ymax>
        </bndbox>
</object>
```

VOC 2007 Dataset

Tensorflow 2.0

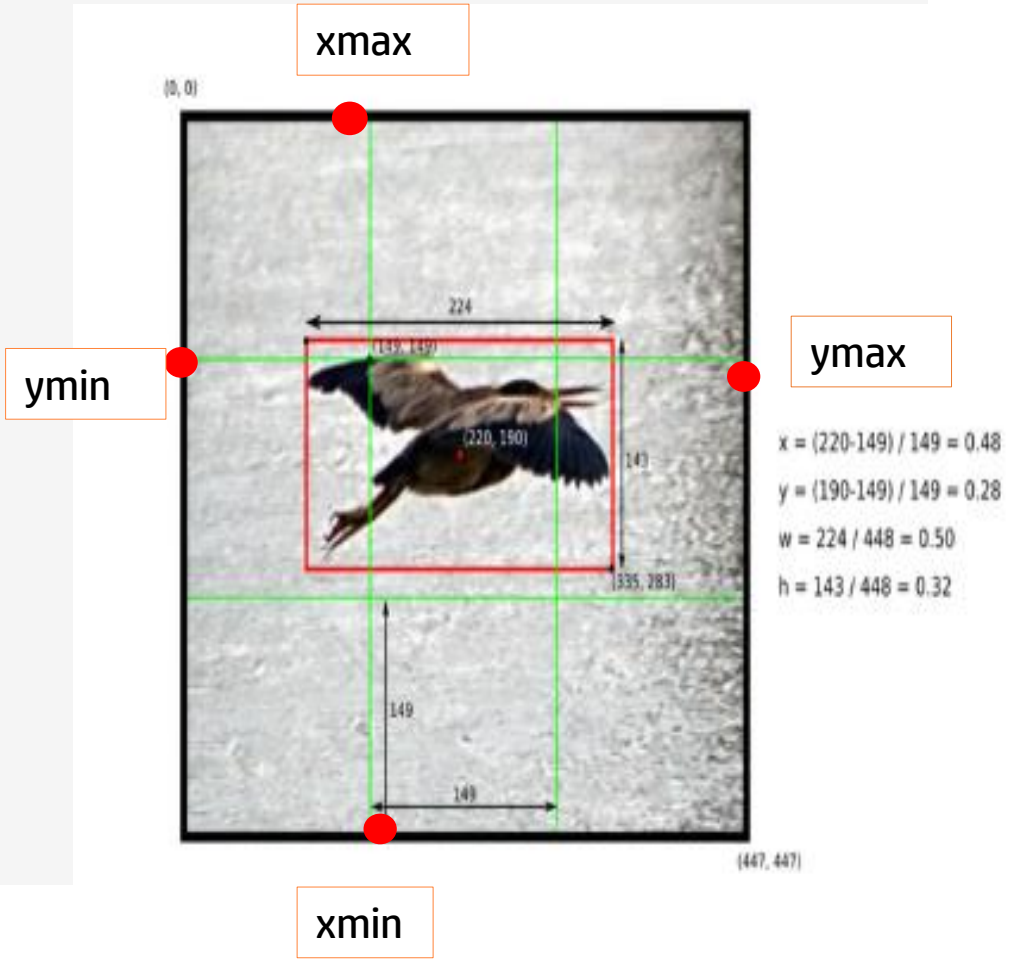
구글의 기계학습 라이브러리
CPU 버전과 GPU 버전이 통합되어 CPU 사용가능

Resnet50

50계층으로 구성된 컨볼루션 신경망

Unified Detection

```
    image_path = image_path.numpy()
    label = label.numpy()
    image = load_image_tf(image_path)
    #image = cv.imread(image_path)
    #image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
    image_h, image_w = image.shape[0:2]
    #image = cv.resize(image, (448, 448))
    image = tf.image.resize(image, (448, 448))
    image = image / 255.
    label_matrix = np.zeros([7, 7, 30]).astype(np.float32)
    for l in label:
        l = bytes.decode(l)
        l = l.split(',')
        l = np.array(l, dtype=np.int32)

        xmin = l[0]
        ymin = l[1]
        xmax = l[2]
        ymax = l[3]
        cls = l[4]
        x = (xmin + xmax) / 2 / image_w
        y = (ymin + ymax) / 2 / image_h
        w = (xmax - xmin) / image_w
        h = (ymax - ymin) / image_h
        loc = [7 * x, 7 * y]
        loc_i = int(loc[1])
        loc_j = int(loc[0])
        y = loc[1] - loc_i
        x = loc[0] - loc_j

        if label_matrix[loc_i, loc_j, 24] == 0:
            label_matrix[loc_i, loc_j, cls] = 1
            label_matrix[loc_i, loc_j, 20:24] = [x, y, w, h]
            label_matrix[loc_i, loc_j, 24] = 1  # response
```

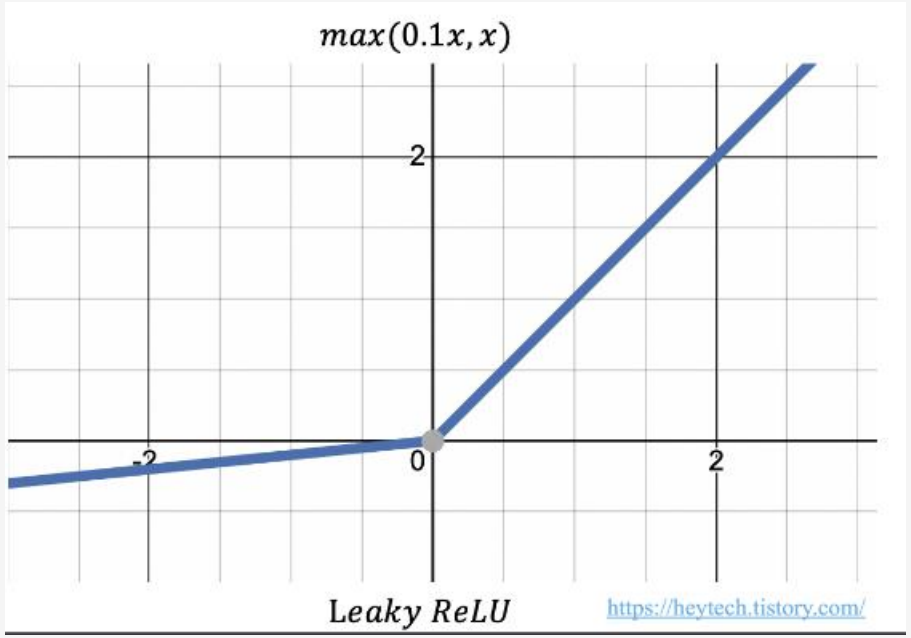


Modeling

```
from tensorflow import keras
import keras.backend as K

class YoloActivation(tf.keras.layers.Layer) :

    def call(self, inputs) :
        classes = tf.nn.softmax(inputs[..., :20], axis = -1)
        coordinates = tf.sigmoid(inputs[..., 20:])
        return tf.concat([classes, coordinates], axis = -1)
```

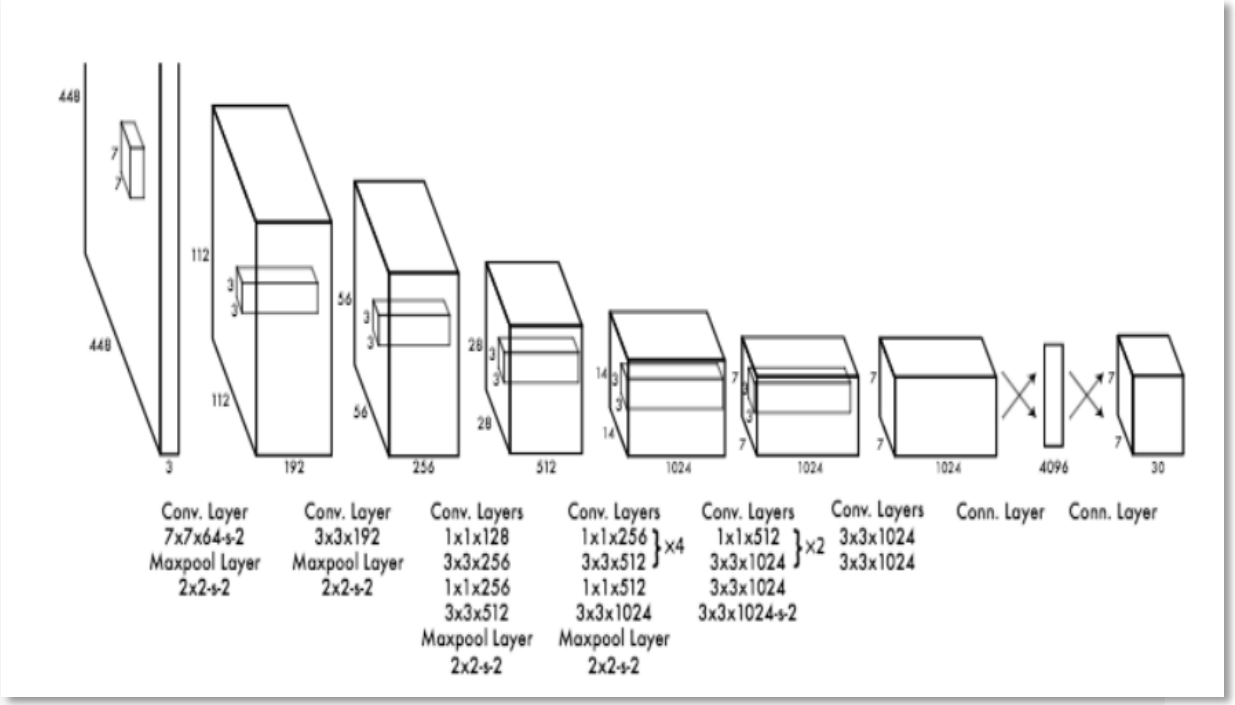
Modeling

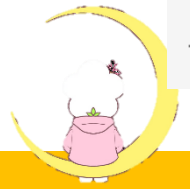
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer, Dropout, Flatten, Reshape
from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalMaxPooling2D, BatchNormalization
from tensorflow.keras.regularizers import l2

lrelu = tf.keras.layers.LeakyReLU(alpha=0.1)

nb_boxes=1
grid_w=7
grid_h=7
cell_w=64
cell_h=64
img_w=grid_w*cell_w
img_h=grid_h*cell_h

featureExtractor = Sequential()
featureExtractor.add(
    tf.keras.applications.ResNet50(
        include_top=False,
        weights = 'imagenet',
        input_shape =(img_h, img_w, 3)
    )
)

featureExtractor.trainable = False
```



Modeling

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer, Dropout, Flatten, Reshape
from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalMaxPooling2D, BatchNormalization
from tensorflow.keras.regularizers import l2

lrelu = tf.keras.layers.LeakyReLU(alpha=0.1)

nb_boxes=1
grid_w=7
grid_h=7
cell_w=64
cell_h=64
img_w=grid_w*cell_w
img_h=grid_h*cell_h

featureExtractor = Sequential()
featureExtractor.add(
    tf.keras.applications.ResNet50(
        include_top=False,
        weights = 'imagenet',
        input_shape =(img_h, img_w, 3)
    )
)

featureExtractor.trainable = False
```





Modeling

```
model = Sequential()
model.add(featureExtractor)
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(1024, activation = lrelu))
model.add(Dropout(0.5))
model.add(Dense(512, activation = lrelu))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(grid_w * grid_h * (20 + 5 * 2)))
model.add(Reshape((grid_w , grid_h , (20 + 5 * 2))))
model.add(YoloActivation())
model.summary()
```

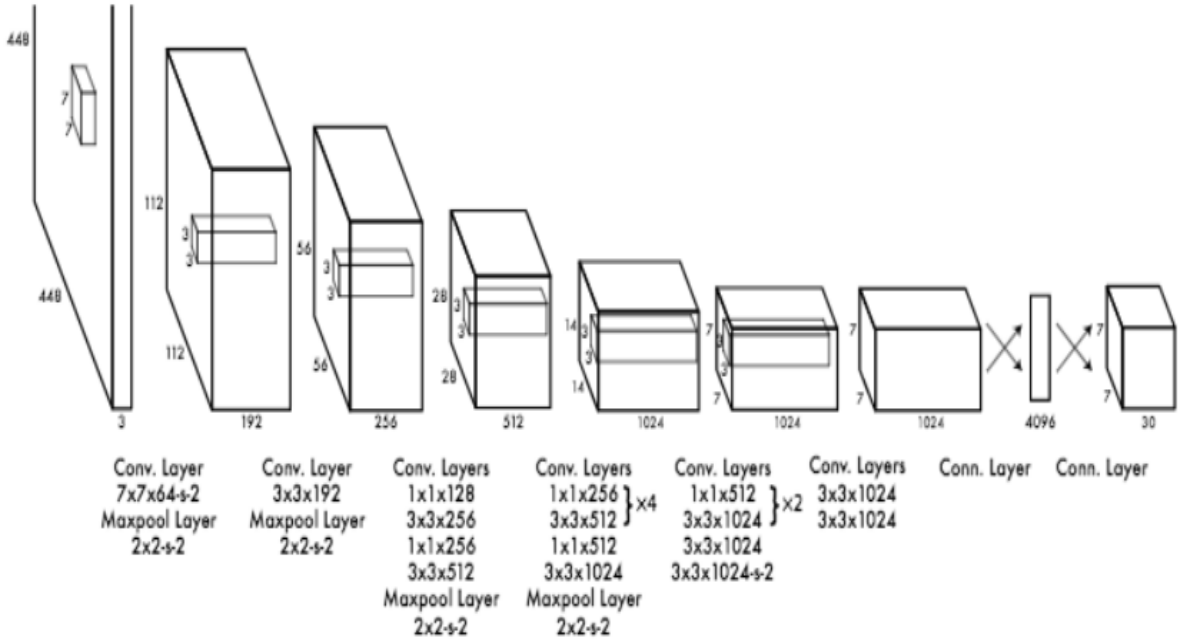
$$S \times S \times (B * 5 + C)$$

S=7, B=2, C=20

Modeling

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 14, 14, 2048)	23587712
max_pooling2d (MaxPooling2D)	(None, 7, 7, 2048)	0
batch_normalization (BatchNormalization)	(None, 7, 7, 2048)	8192
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 1024)	102761472
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_1 (Dropout)	(None, 512)	0
batch_normalization_1 (BatchNormalization)	(None, 512)	2048
dense_2 (Dense)	(None, 1470)	754110
reshape (Reshape)	(None, 7, 7, 30)	0
yolo_activation (YoloActivation)	(None, 7, 7, 30)	0

Total params: 127,638,334
Trainable params: 104,045,502
Non-trainable params: 23,592,832



Modeling

Learning Rate 정의

```
LR_SCHEDULE = [
    # (epoch to start, learning rate) tuples
    (0, 1e-1),
    (2, 1e-2),
    (4, 1e-3),
    (6, 1e-4),
]


def lr_schedule(epoch, lr):
    """Helper function to retrieve the scheduled learning rate based on epoch."""
    if epoch < LR_SCHEDULE[0][0] or epoch > LR_SCHEDULE[-1][0]:
        return lr
    for i in range(len(LR_SCHEDULE)):
        if epoch == LR_SCHEDULE[i][0]:
            return LR_SCHEDULE[i][1]
    return lr
```

Loss Function

```
## YOLO LOSS
def yoloLoss(y_true, y_pred) :
    coordLoss = CoordLoss(y_true, y_pred)
    confidenceLoss = ConfidenceLoss(y_true, y_pred)
    classLoss = ClassLoss(y_true, y_pred)

    return 5 * coordLoss + 2 * confidenceLoss + 0.5 * classLoss

def CoordLoss(y_true, y_pred) :
    #find if the object exists in the grid
    existsObject = tf.expand_dims(y_true[..., 24], -1)

    xy_pred = existsObject * y_pred[..., 20:22]
    xy_true = existsObject * y_true[..., 20:22]

    wh_pred = existsObject * tf.math.sign(y_pred[..., 22:24]) * tf.sqrt(tf.math.abs(y_pred[..., 22:24]))
    wh_true = existsObject * tf.sqrt(y_true[..., 22:24])

    coordLoss = tf.reduce_sum(tf.math.square(wh_pred - wh_true))
    coordLoss += tf.reduce_sum(tf.math.square(xy_pred - xy_true))

    return coordLoss / tf.cast(tf.math.count_nonzero(existsObject) , dtype = tf.float32) #for mean
```

[Yolo 손실 함수 정의 (예측값, 레이블값)]

- Bbox의 좌표에 대한 손실값(coordLoss)을 5배 가중치로 두어 높은 패널티 부여
- 객체를 포함하고 있지 않은 bbox에 대한 손실값(classLoss)에 0.5 가중치를 부여해 패널티를 낮춤

Loss Function

```python
def ConfidenceLoss(y_true, y_pred):
    #find if the object exists in the grid
    existsObject = tf.expand_dims(y_true[..., 24], -1)

    #for object
    confidenceLoss = tf.reduce_sum(tf.math.square(existsObject * (y_true[..., 24:25] - y_pred[..., 24:25])))

    #for no obejct
    confidenceLoss += 0.5 * tf.reduce_sum(tf.math.square((1 - existsObject) * (y_true[..., 24:25] - y_pred[..., 24:25])))

    return confidenceLoss / tf.cast(tf.math.count_nonzero(existsObject) , dtype = tf.float32) #for mean

def ClassLoss(y_true, y_pred) :
    #find if the object exists in the grid
    existsObject = tf.expand_dims(y_true[..., 24], -1)

    classLoss = tf.reduce_sum(tf.math.square(existsObject * (y_true[:20] - y_pred[:20])))

    return classLoss / tf.cast(tf.math.count_nonzero(existsObject) , dtype = tf.float32) #for mean
```

Save weights of the best model

Adding a callback for saving the weights

```
# defining a function to save the weights of best model
from tensorflow.keras.callbacks import ModelCheckpoint

mcp_save = ModelCheckpoint('weight_6.hdf5', save_best_only=True, monitor='val_loss', mode='min')
```

[ModelCheckpoint (mcp_save변수에 저장)]

- 모든 에폭마다 모델을 저장하기에 시간상 비효율적
- 학습하면서 정의한 조건을 만족하였을 경우, 중간에 모델 가중치를 저장함.
- 중간에 메모리 과부하나 오류가 나더라도, 다시 가중치를 호출해 학습을 이어나갈 수 있어 효율적

Training

[Compiling the Model]

- 데이터 훈련 및 모델 생성을 위한 작업

```
model.compile(loss=yoloLoss ,optimizer='adam', metrics = [CoordLoss, ConfidenceLoss, ClassLoss])
```

1. loss: 앞에서 정의한 yoloLoss 를 손실 함수로 설정
2. Optimizer: Adam 최적화 기법을 사용하여 최적값에 수렴하도록 함
3. Metrics: 모델 퍼포먼스 평가를 위하여 CoordLoss, ConfidenceLoss, ClassLoss 지정

Training

```
hist = model.fit(x=train,
          batch_size=batch_size,
          epochs = 6,
          verbose = 1,
          workers= 8,
          validation_data = val,
           callbacks=[
              CustomLearningRateScheduler(lr_schedule),
              mcp_save
          ])
```

1. model.fit() 으로 학습 시작 : VOC2007 Train 데이터로 학습
2. Epoch : 4로 지정
3. Workers : 스레딩 기반 처리 과정을 사용할 경우, 가동할 수 있는 처리 과정의 최대 개수 (다중 처리 방식)
4. Callback : 에폭에 따라 학습률을 조정. 인자로 mcp_save를 사용하여 콜백 함수 사용

Training

```
Epoch 00000: Learning rate is 0.1000.
Epoch 1/4
1251/1251 [==============================] - 7576s 6s/step - loss: 10.2906 - CoordLoss: 1.2294 - ConfidenceLoss: 1.0347 - ClassLoss: 4.
1490 - val_loss: 10.2673 - val_CoordLoss: 1.2371 - val_ConfidenceLoss: 1.0000 - val_ClassLoss: 4.1641

Epoch 00001: Learning rate is 0.1000.
Epoch 2/4
1251/1251 [==============================] - 8406s 7s/step - loss: 9.5356 - CoordLoss: 1.1254 - ConfidenceLoss: 0.9719 - ClassLoss: 3.9
314 - val_loss: 9.4275 - val_CoordLoss: 1.1085 - val_ConfidenceLoss: 0.9718 - val_ClassLoss: 3.8827

Epoch 00002: Learning rate is 0.0100.
Epoch 3/4
1251/1251 [==============================] - 7820s 6s/step - loss: 9.3337 - CoordLoss: 1.0951 - ConfidenceLoss: 0.9624 - ClassLoss: 3.8
674 - val_loss: 9.4349 - val_CoordLoss: 1.1078 - val_ConfidenceLoss: 0.9753 - val_ClassLoss: 3.8902

Epoch 00003: Learning rate is 0.0100.
Epoch 4/4
1251/1251 [==============================] - 7006s 6s/step - loss: 9.3600 - CoordLoss: 1.0974 - ConfidenceLoss: 0.9688 - ClassLoss: 3.8
703 - val_loss: 9.4355 - val_CoordLoss: 1.1068 - val_ConfidenceLoss: 0.9779 - val_ClassLoss: 3.8916
```

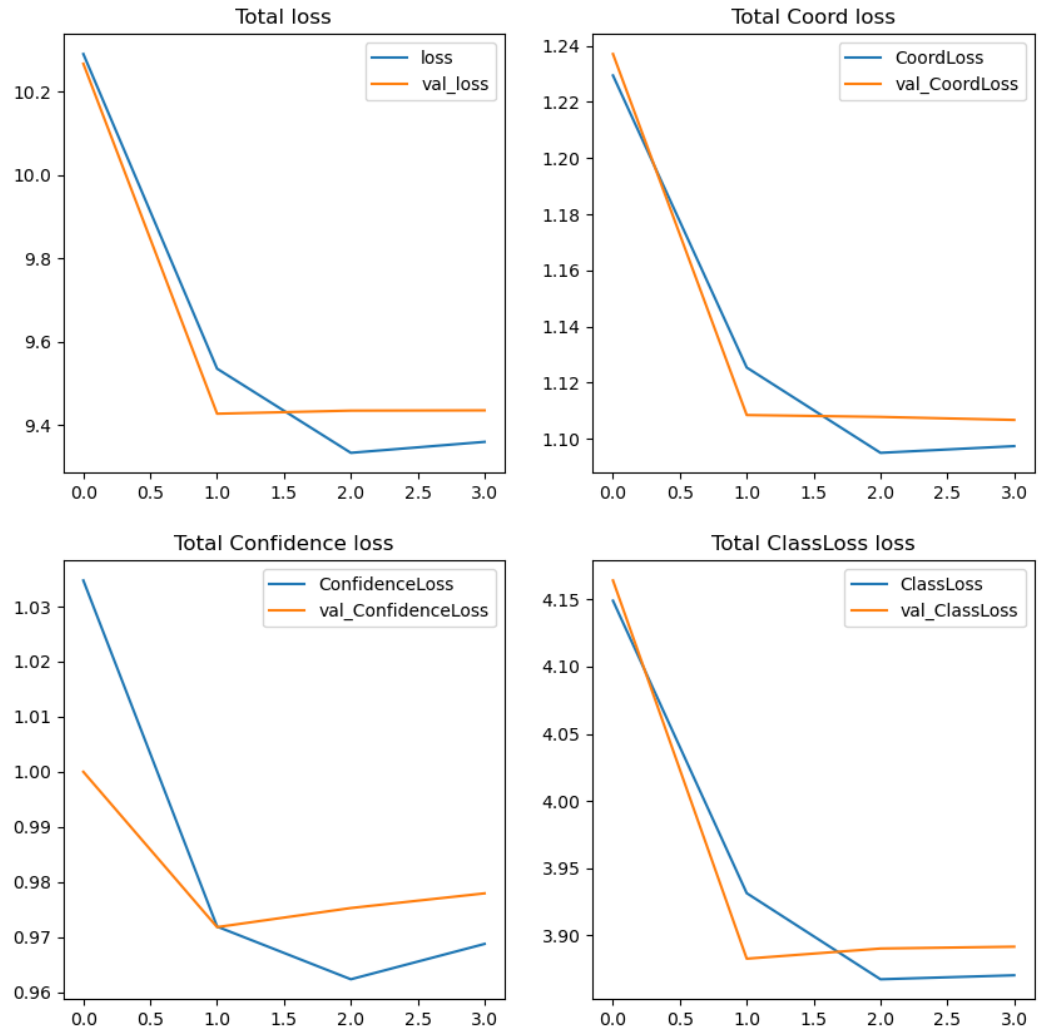
Loss Evaluation – loss output

```
fig, ax = plt.subplots(ncols =2, nrows =2,  figsize= (10, 10))

ax[0][0].plot(hist.history['loss'], label = 'loss')
ax[0][0].plot(hist.history['val_loss'], label = 'val_loss')
ax[0][0].title.set_text('Total loss')
ax[0][0].legend()

ax[0][1].plot(hist.history['CoordLoss'], label = 'CoordLoss')
ax[0][1].plot(hist.history['val_CoordLoss'], label = 'val_CoordLoss')
ax[0][1].title.set_text('Total Coord loss')
ax[0][1].legend()

ax[1][0].plot(hist.history['ConfidenceLoss'], label = 'ConfidenceLoss')
ax[1][0].plot(hist.history['val_ConfidenceLoss'], label = 'val_ConfidenceLoss')
ax[1][0].title.set_text('Total Confidence loss')
ax[1][0].legend()

ax[1][1].plot(hist.history['ClassLoss'], label = 'ClassLoss')
ax[1][1].plot(hist.history['val_ClassLoss'], label = 'val_ClassLoss')
ax[1][1].title.set_text('Total ClassLoss loss')
ax[1][1].legend()
```

총 4개의 그래프로
손실 함수 시각화

Loss Evaluation – loss output

```
<matplotlib.legend.Legend at 0x2808fb84520>
```



Final Output

```
confidence_thresh = 0.1
grid_width = 1 / 7
grid_dimen = np.array([448 * grid_width, 448 * grid_width]).astype(np.int32)
list_classes = list(classes_num)
plots = 2
fig, ax = plt.subplots(ncols=plots, figsize = (20, 20))
for idx in range(plots) :
    sample_image = (res[0][idx] * 255).astype(np.uint8)
```

- 이미지를 나누어 7*7 그리드 셀 설정 → grid_width 변수에 1/7 값 저장
- 448*448 (가로*세로) 각각 1/7을 곱해주면, 그리드 셀의 차원을 알 수 있음

데이터를 np.uint8 (부호 없는 8비트 정수형 배열) 로 변환 -> 정규화 필요

Final Output

```
for i in range(7) :
    for j in range(7) :
        if res[1][idx, i, j, 24] < confidence_thresh: continue
        sample_coords = res[1][idx,i, j, 20:24] #[x, y, w, h]
        grid_width = 1/7

        #width = sample_co
        #height = sample_c
        width = sample_coo
        height = sample_co
        x1 = (j * grid_dim
        y1 = (i * grid_dim
        x2 = x1 + width
        y2 = y1 + height

         cv.rectangle(sampl
                      tuple
                      tuple
                      (255,
                      2)

         cv.putText(img=sa
                text = list
                org = tuple
                fontFace =
                fontScale =
                color = (25
                thickness =2,
                 lineType = cv.LINE_AA
                )
    ax[idx].imshow(sample_image)
plt.show()
```

Limitations

Geforce GPU 부재로 인한 NVIDIA 드라이브 사용 불가

```
~₩anaconda3₩lib₩site-packages₩torch₩cuda₩__init__.py in _lazy_init()
    227         if 'CUDA_MODULE_LOADING' not in os.environ:
    228             os.environ['CUDA_MODULE_LOADING'] = 'LAZY'
--> 229         torch._C._cuda_init()
    230         # Some of the queued calls may reentrantly call _lazy_init();
    231         # we need to just return without initializing in that case.

RuntimeError: Found no NVIDIA driver on your system. Please check that you have an NVIDIA GPU and installed a driver from http://www.
vidia.com/Download/index.aspx
```

GPU 사용 불가로 인하여 CPU 사용 -> Train 시간 소요 多

```
Epoch 00004: Learning rate is 0.0010.
Epoch 5/6
1251/1251 [==============================] - 7300s 6s/step - loss: 10.3248 - CoordLoss: 1.2057 - ConfidenceLoss: 1.1519 - ClassLoss: 3.
9873 - val_loss: 10.4318 - val_CoordLoss: 1.2237 - val_ConfidenceLoss: 1.1601 - val_ClassLoss: 3.9861

Epoch 00005: Learning rate is 0.0010.
Epoch 6/6
1251/1251 [==============================] - 12291s 10s/step - loss: 10.3171 - CoordLoss: 1.2039 - ConfidenceLoss: 1.1535 - ClassLoss:
3.9823 - val_loss: 10.4251 - val_CoordLoss: 1.2251 - val_ConfidenceLoss: 1.1546 - val_ClassLoss: 3.9814
```

Limitations

Trial 1 - Pytorch version

```
from loss import YoloLoss

seed = 123
torch.manual_seed(seed) #get same data set loading

# Hyperparameters etc.
LEARNING_RATE = 2e-5
DEVICE = "cuda" if torch.cuda.is_available else "cpu"
BATCH_SIZE = 16 # 64 in original paper but I don't have that much vram, grad accum?
#batch size 64라고 언급된 논문 본적 없음, gpu 업그레이드 필요함
WEIGHT_DECAY = 0 #not train entire model- take a long time to train , 계산상의 이유로 무거운 데이터 증강, computational reasons
#정규화 없음
EPOCHS = 1000
NUM_WORKERS = 2
PIN_MEMORY = True
LOAD_MODEL = False
LOAD_MODEL_FILE = "overfit.pth.tar" #오버피팅하고 나중에 실행할 예정
IMG_DIR = "C:/Users/Owner/Downloads/archive/images"
LABEL_DIR = "C:/Users/Owner/Downloads/archive/labels"
```

```
230         # Some of the queued calls may reentrantly call _lazy_init();
231         # we need to just return without initializing in that case.

RuntimeError: Found no NVIDIA driver on your system. Please check that you have an NVIDIA GPU and installed a driver from http://www.nvidia.com/Download/index.aspx
```

Limitations

Trial 2 - Tensorflow version

```
In [26]: print(trainingDataGenerator)

<__main__.DataGenerator object at 0x000001C06AA12DF0>
```

```
In [14]: x_train, y_train = trainingDataGenerator.__getitem__(0)

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[14], line 1
----> 1 x_train, y_train = trainingDataGenerator.__getitem__(0)
      2 #AttributeError: 'NoneType' object has no attribute 'shape' -
      3 #이미지가 보일수 없어서 생긴 에러.?
      4 #The AttributeError: 'nonetype' object has no attribute 'shape' error occurs when you try to access the shape attribute of
an object that is None.

Cell In[12], line 17, in DataGenerator.__getitem__(self, idx)
     15 for i in range(0, len(batch_x)):
     16     img_path, label = batch_x[i], batch_y[i]
---> 17     image, label_matrix = self.read(img_path, label) #actual image array (IMAGE_SIZE, IMAGE_SIZE, 3) (GRID_SIZE, GRID_SIZE, 5 *
BOX_SIZE + CLASS)
     18     train_image.append(image)
     19     train_label.append(label_matrix)

Cell In[12], line 31, in DataGenerator.read(self, img_path, label)
     29 def read(self, img_path, label):
     30     image = cv.imread(img_path)
---> 31     h, w = image.shape[0:2]
     32     #h, w = image.shape[0]
     33     image = cv.resize(image, (IMAGE_SIZE, IMAGE_SIZE))

AttributeError: 'NoneType' object has no attribute 'shape'
```

Limitations

Trial 3 – YOLO on CoLab

```
keras.backend.image_data_format()

def get_model():
  model = Sequential()

  #Layer 1
  model.add(Convolution2D(16, (3, 3), padding='same', input_shape=(3, 448, 448),strides=(1,1))) #은 학습에 사용되는 데이터 샘플링 비율
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D(pool_size=(2, 2)))


  # Layer 2
  model.add(Convolution2D(32, (3, 3), padding='same'))
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D(pool_size=(2, 2),padding='valid'))


  # Layer 3
  model.add(Convolution2D(64, (3, 3), padding='same'))
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D(pool_size=(2, 2),padding='valid'))

  # Layer 4
  model.add(Convolution2D(128, (3, 3), padding='same'))
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D(pool_size=(2, 2),padding='valid'))

  # Layer 5
  model.add(Convolution2D(256, (3, 3), padding='same'))
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D(pool_size=(2, 2),padding='valid'))
```

Limitations

Trial 3 – YOLO on CoLab

```
test_image = mpimg.imread('/content/test_images/test1.jpg')
aa = preprocess(test_image)
batchch = np.expand_dims(aa, axis=0)
#bb = list(bb)
batch_output= model.predict(batchch)
```

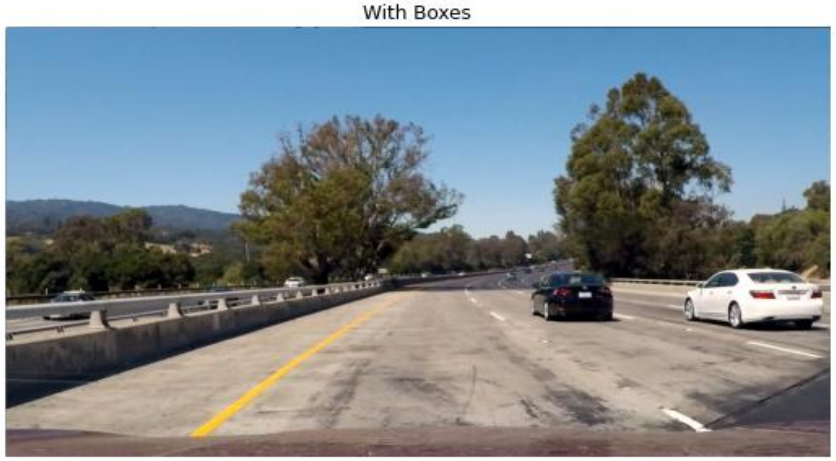
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-d0509f283017> in <module>
      3 batchch = np.expand_dims(aa, axis=0)
      4 #bb = list(bb)
----> 5 batch_output= model.predict(batchch)
      6

                                 2 frames
/usr/local/lib/python3.8/dist-packages/tensorflow/python/eager/polymorphic_function/polymorphic_function.py in _call(self, *args, **kwds)
    910       # In this case we have created variables on the first call, so we run the
    911       # defunned version which is guaranteed to never create variables.
--> 912       return self._no_variable_creation_fn(*args, **kwds)  # pylint: disable=not-callable
    913     elif self._variable_creation_fn is not None:
    914       # Release the lock early so that multiple threads can perform the call

TypeError: 'NoneType' object is not callable
```

Limitations

Trial 3 – YOLO on CoLab

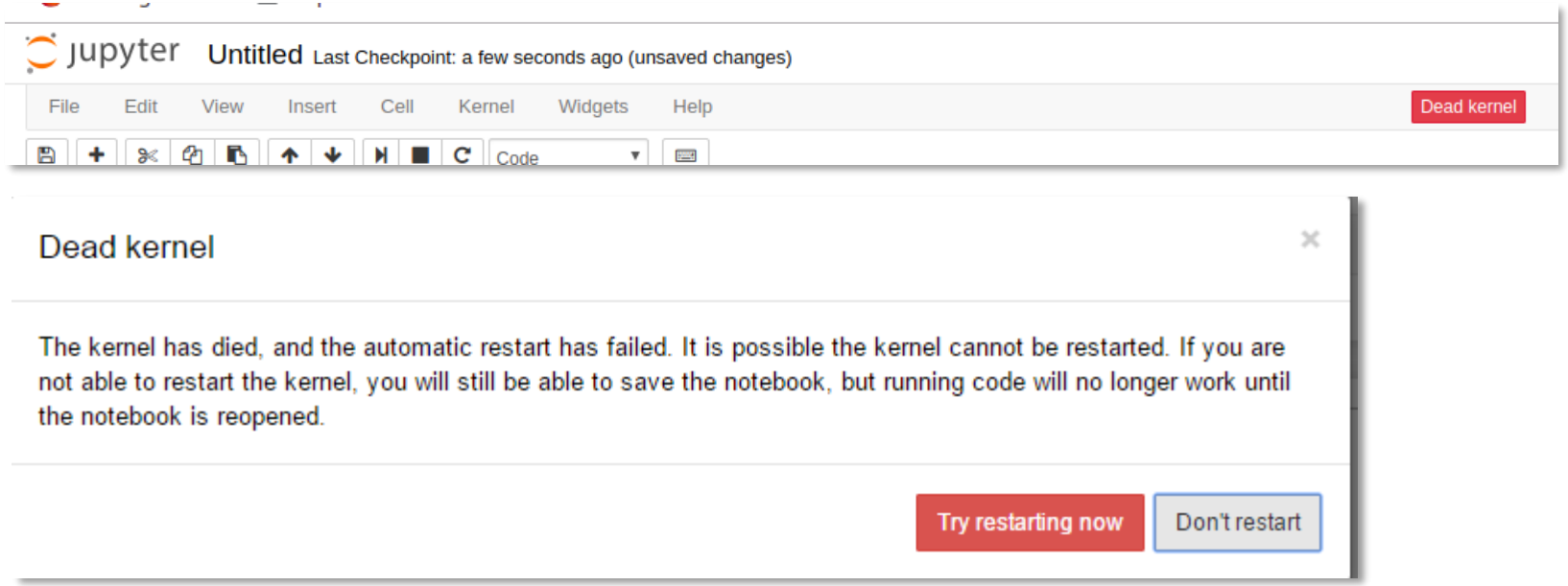
```
]  boxes = yolo_output_to_car_boxes(batch_output[0], threshold=0.25)
   final = draw_boxes(boxes, test_image, ((0,test_image.shape[1]),(0,test_image.shape[0])))

   plt.rcParams['figure.figsize'] = (20, 11.2)
   plt.subplot(1,2,1)
   plt.imshow(test_image)
   plt.axis('off')
   plt.title("Original Image")
   plt.subplot(1,2,2)
   plt.imshow(final)
   plt.axis('off')
   plt.title("With Boxes")
```

```
<ipython-input-18-9041899fc097>:96: RuntimeWarning: invalid value encountered in power
  bx.w = cords[grid, b, 2] ** sqrt
<ipython-input-18-9041899fc097>:97: RuntimeWarning: invalid value encountered in power
  bx.h = cords[grid, b, 3] ** sqrt
Text(0.5, 1.0, 'With Boxes')
```



Limitations

Trial 4 – Dead Kernel

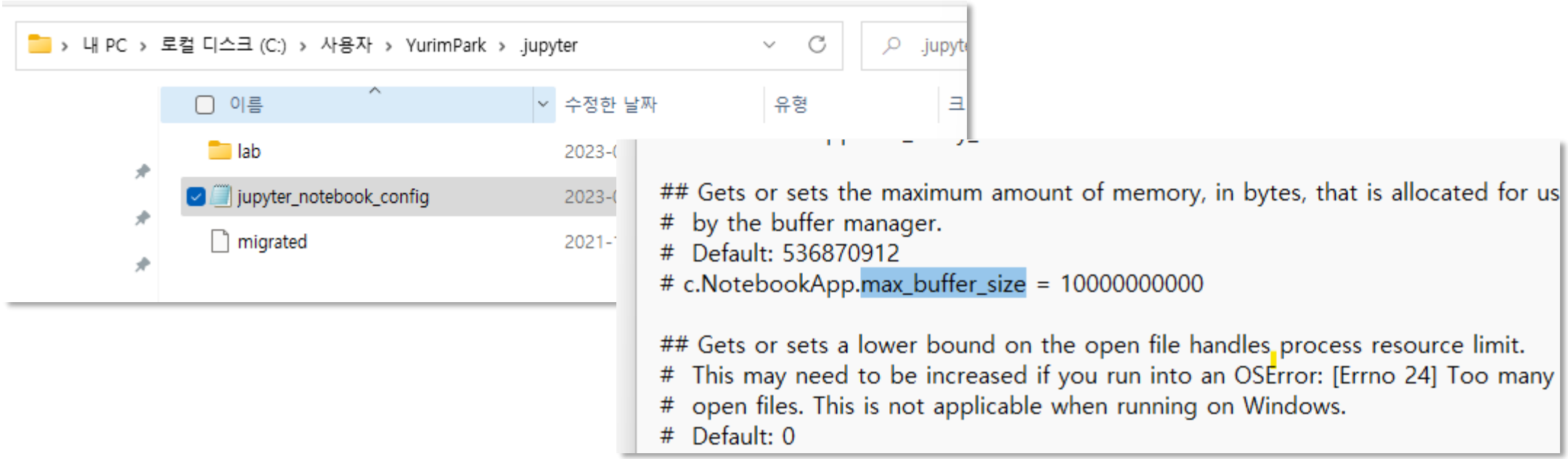


[jupyter notebook – 커널이 계속 죽는 문제]

- stackoverflow, 구글링 → 메모리 할당량 초과로 인한 문제

Limitations

Trial 4 – Dead Kernel



[jupyter notebook – 커널이 계속 죽는 문제]

- .jupyter 파일에 들어가 해당 .py파일에 들어감

- max_buffer_size = 536870912 > max_buffer_size =10000000000 로 변환해주었으나, 미해결

Plans

- 넉넉한 용량과 Geforce GPU 지원이 되는 데스크탑 or 랩탑 필요 최우선
- batch size, epoch, learning rate 변경 및 train
- mAP 측정 및 비교
- 에러 해결을 위한 추가적인 딥러닝 공부

Plan

논문 리딩 + 구현 '완성'

" 하드웨어 문제로 인한 에러로 시간과 에너지가 많이 소요되었지만, 오히려 지금 경험한 시행착오들이 도움이 될 것이라 생각 "

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

9 May 2016

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by