# 소모둠 스터디

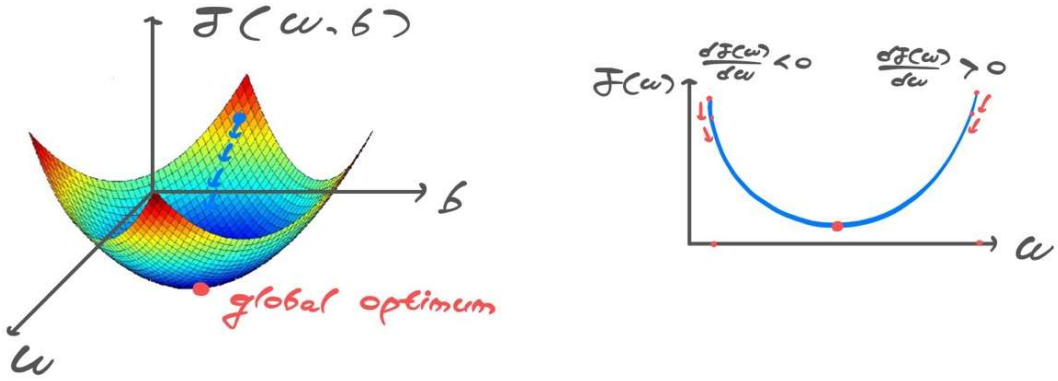
## AI 논문리뷰

오서영, 최연석, 허지혜

# 경사 하강법 (Gradient Descent)

오서영

**경사 하강법 (Gradient Descent)**
비용함수 J(W, b)를 최소화하는 W, b 찾기




$$w := w - \alpha \frac{\partial J(w,b)}{\partial w} \qquad \alpha : \text{learning rate}$$

$$b := b - \alpha \frac{\partial J(w,b)}{\partial b}$$

> for $t = 1, \cdots, 5000$

Forward Prop on $X^{\{t\}}$

$$Z^{[1]} = W^{[1]} X^{\{t\}} + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$\vdots$$
$$A^{[L]} = g^{[L]}(Z^{[L]})$$

Compute cost $J = \frac{1}{1000} \sum_{i=1}^{l} L(\hat{y}^{(i)}, y^{(i)})$ from $X^{\{t\}}, Y^{\{t\}}$

Backprop to compute gradient wrt $J^{\{t\}}$

$$W^{[l]} := W^{[l]} - \alpha dW^{[l]}$$
$$b^{[l]} := b^{[l]} - \alpha db^{[l]}$$

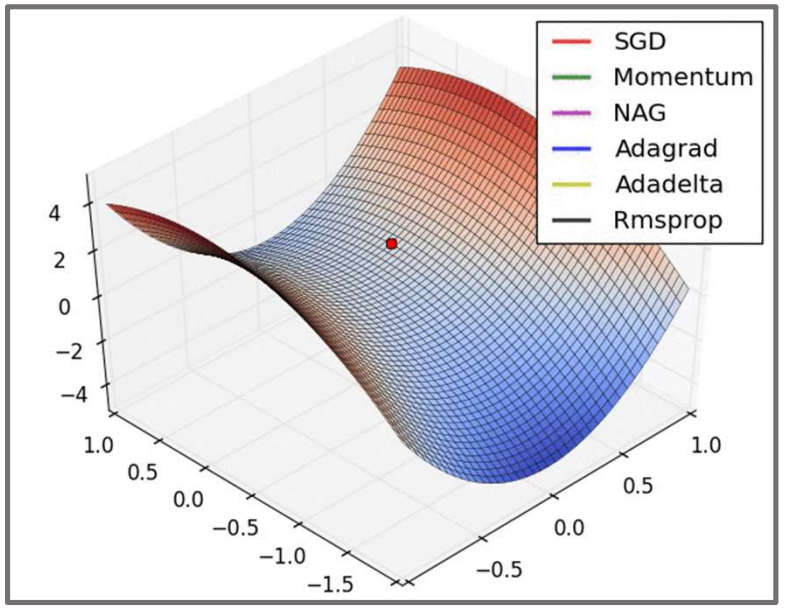
**Mini-batch gradient descent**

# 그 외 최적화 알고리즘

## ADAM

**Algorithm 1:** *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. $g_t^2$ indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With $\beta_1^t$ and $\beta_2^t$ we denote $\beta_1$ and $\beta_2$ to the power $t$.

**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Require:** $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  **while** $\theta_t$ not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  **end while**
  **return** $\theta_t$ (Resulting parameters)



# 파이썬 구현 1 (without DL framework)

**Gradient descent**

```python
def update_params_gd(parameters, grads, learning_rate):

    L = len(parameters) // 2 # layers

    for l in range(L):
        parameters["W" + str(l+1)] = parameters["W" + str(l+1)] - learning_rate*grads['dW' + str(l+1)]
        parameters["b" + str(l+1)] = parameters["b" + str(l+1)] - learning_rate*grads['db' + str(l+1)]

    return parameters
```

```python
def forward_prop(X, parameters):


    # retrieve parameters
    W1 = parameters["W1"]
    b1 = parameters["b1"]
    W2 = parameters["W2"]
    b2 = parameters["b2"]
    W3 = parameters["W3"]
    b3 = parameters["b3"]

    z1 = np.dot(W1, X) + b1
    a1 = relu(z1)
    z2 = np.dot(W2, a1) + b2
    a2 = relu(z2)
    z3 = np.dot(W3, a2) + b3
    a3 = sigmoid(z3)

    cache = (z1, a1, W1, b1, z2, a2, W2, b2, z3, a3, W3, b3)

    return a3, cache
```

```python
def backward_prop(X, Y, cache):

    m = X.shape[1]
    (z1, a1, W1, b1, z2, a2, W2, b2, z3, a3, W3, b3) = cache

    dz3 = 1./m * (a3 - Y)
    dW3 = np.dot(dz3, a2.T)
    db3 = np.sum(dz3, axis=1, keepdims = True)

    da2 = np.dot(W3.T, dz3)
    dz2 = np.multiply(da2, np.int64(a2 > 0))
    dW2 = np.dot(dz2, a1.T)
    db2 = np.sum(dz2, axis=1, keepdims = True)

    da1 = np.dot(W2.T, dz2)
    dz1 = np.multiply(da1, np.int64(a1 > 0))
    dW1 = np.dot(dz1, X.T)
    db1 = np.sum(dz1, axis=1, keepdims = True)

    gradients = {"dz3": dz3, "dW3": dW3, "db3": db3,
                 "da2": da2, "dz2": dz2, "dW2": dW2, "db2": db2,
                 "da1": da1, "dz1": dz1, "dW1": dW1, "db1": db1}

    return gradients
```

# 파이썬 구현 2 (custom optimizer in keras)

```
from keras.optimizers import Optimizer
from keras.legacy import interfaces
from keras import backend as K

class custom_Adam(Optimizer):

    def __init__(self, lr = 0.001, beta1 = 0.9, beta2 = 0.999,
                 epsilon = 1e-10, decay = 0, **kwargs):

        # store hyperparameters
        super(Adam, self).__init__(**kwargs)

                                    ⋮

    def get_updates(self, loss, params):
        grads = self.get_gradients(loss, params)
        self.updates = [K.update_add(self.iterations, 1)]

        lr = self.lr
        if self.initial_decay > 0:
            lr = lr * (1 / (1 + self.decay * K.cast(self.iterations,
                                                    K.dtype(self.decay))))

        t = K.cast(self.iterations, K.floatx()) + 1
        lr_t = lr * (K.sqrt(1. - K.pow(self.beta2, t)) /
                     (1 - K.pow(self.beta1, t)))

                                    ⋮
```

**ADAM**

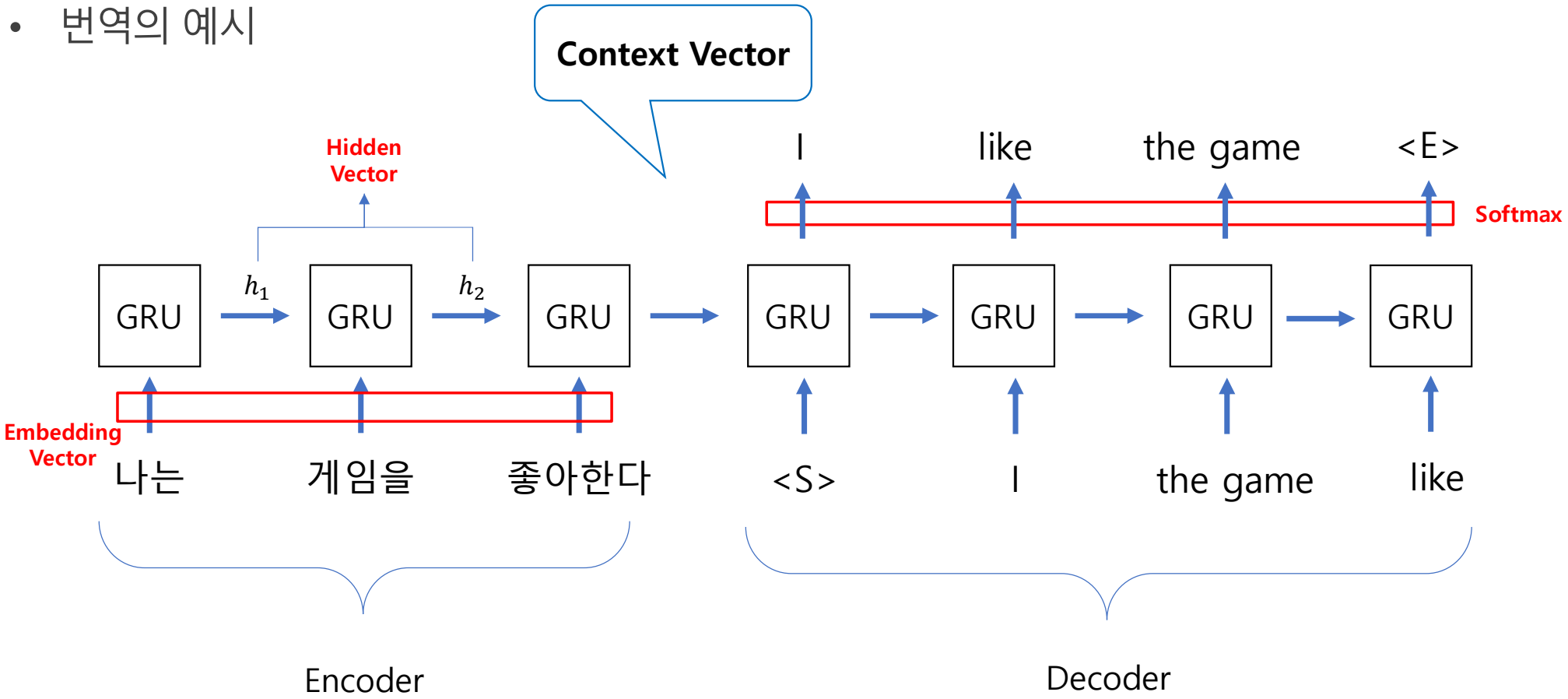
```
ms = [K.zeros(K.int_shape(p), dtype=K.dtype(p)) for p in params]
vs = [K.zeros(K.int_shape(p), dtype=K.dtype(p)) for p in params]
if self.amsgrad:
    vhats = [K.zeros(K.int_shape(p), dtype=K.dtype(p)) for p in params]
else:
    vhats = [K.zeros((1,1)) for _ in params]
self.weights = [self.iterations] + ms + vs + vhats

for p, g, m, v, vhat in zip(params, grads, ms, vs, vhats):
    m_t = (self.beta1 * m) + (1. - self.beta1) * g
    v_t = (self.beta2 * v) + (1. - self.beta2) * K.square(g)
    if self.amsgrad:
        vhat_t = K.maximum(vhat, v_t)
        p_t = p - lr_t * m_t / (K.sqrt(vhat_t) + self.epsilon)
        self.updates.append(K.update(vhat, vhat_t))
    else:
        p_t = p - lr_t * m_t / (K.sqrt(v_t) + self.epsilon)

    self.updates.append(K.update(m, m_t))
    self.updates.append(K.update(v, v_t))
    new_p = p_t

                                    ⋮
```

```
model.compile(loss='sparse_categorical_crossentropy', optimizer = custom_Adam, metrics=['accuracy'])
hist = model.fit(x_train, y_train,
```

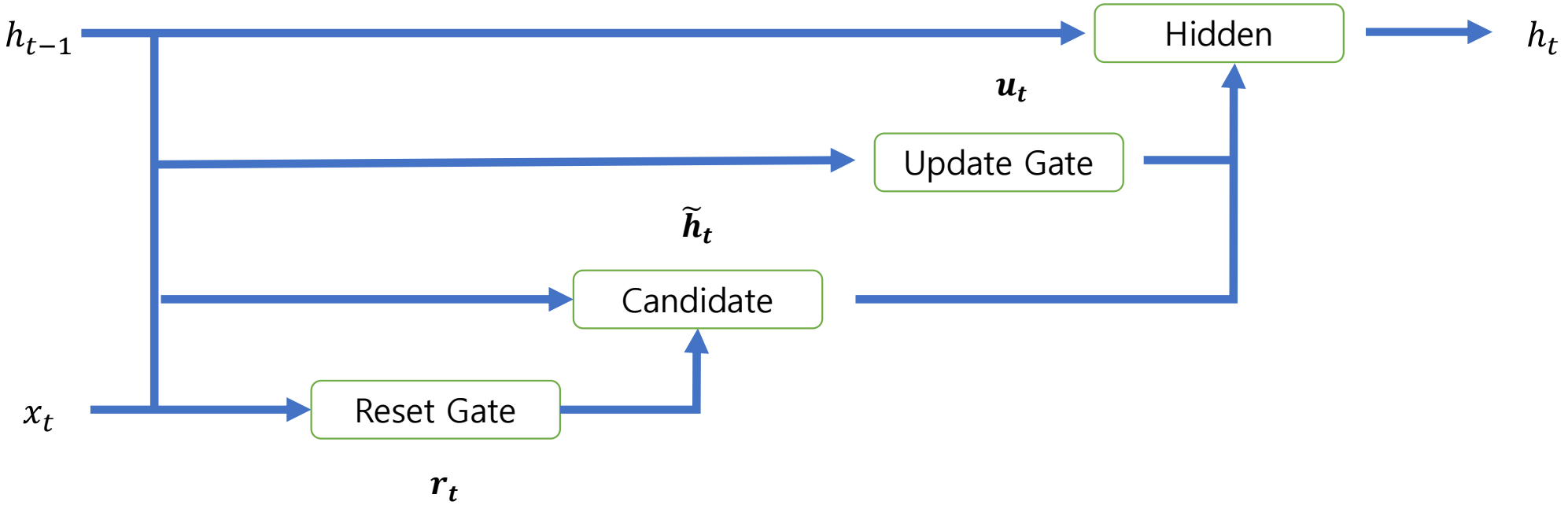
# Sequence to Sequence (S2S)

최연석

- 번역의 예시



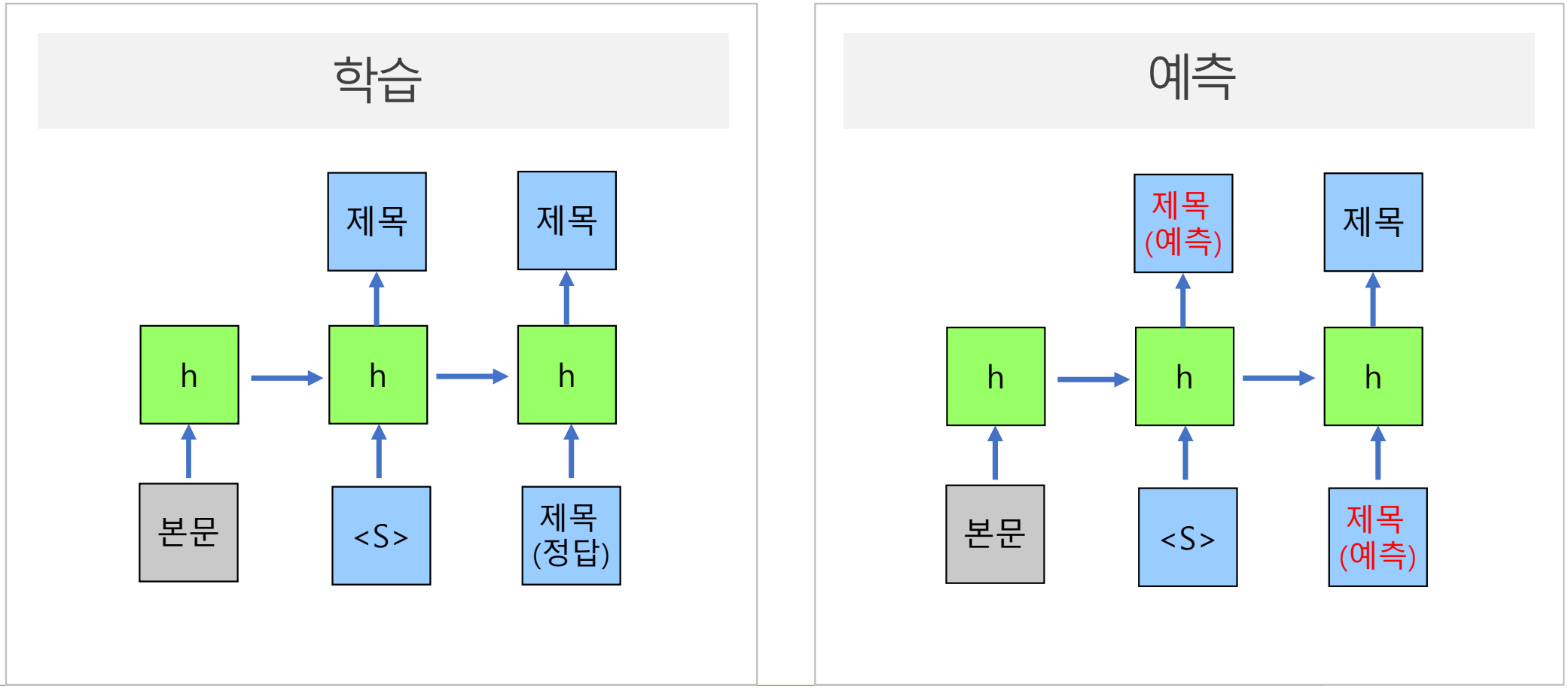
# Gated Recurrent Unit(GRU) 구조




1. Reset Gate : Candidate 계산 과정에서 과거의 정보를 어느정도 제거할지에 대한 값을 도출하는 역할을 함 (0~1 사이의 값으로 이루어진 벡터)
2. Candidate : 현시점의 정보($h_{t-1}$)와 Reset Gate를 통해 줄어든 과거 정보를 취합하여 정보 후보군을 계산하는 단계
3. Update Gate : Hidden을 계산하는 과정에서 과거의 정보와 현재 정보 결합 비율에 대한 값을 도출하는 역할 (0~1 사이의 값으로 이루어진 벡터)
4. Hidden : Update Gate를 통해 현재정보와 과거 정보를 조합하여 GRU의 최종 결과인 hidden vector를 계산

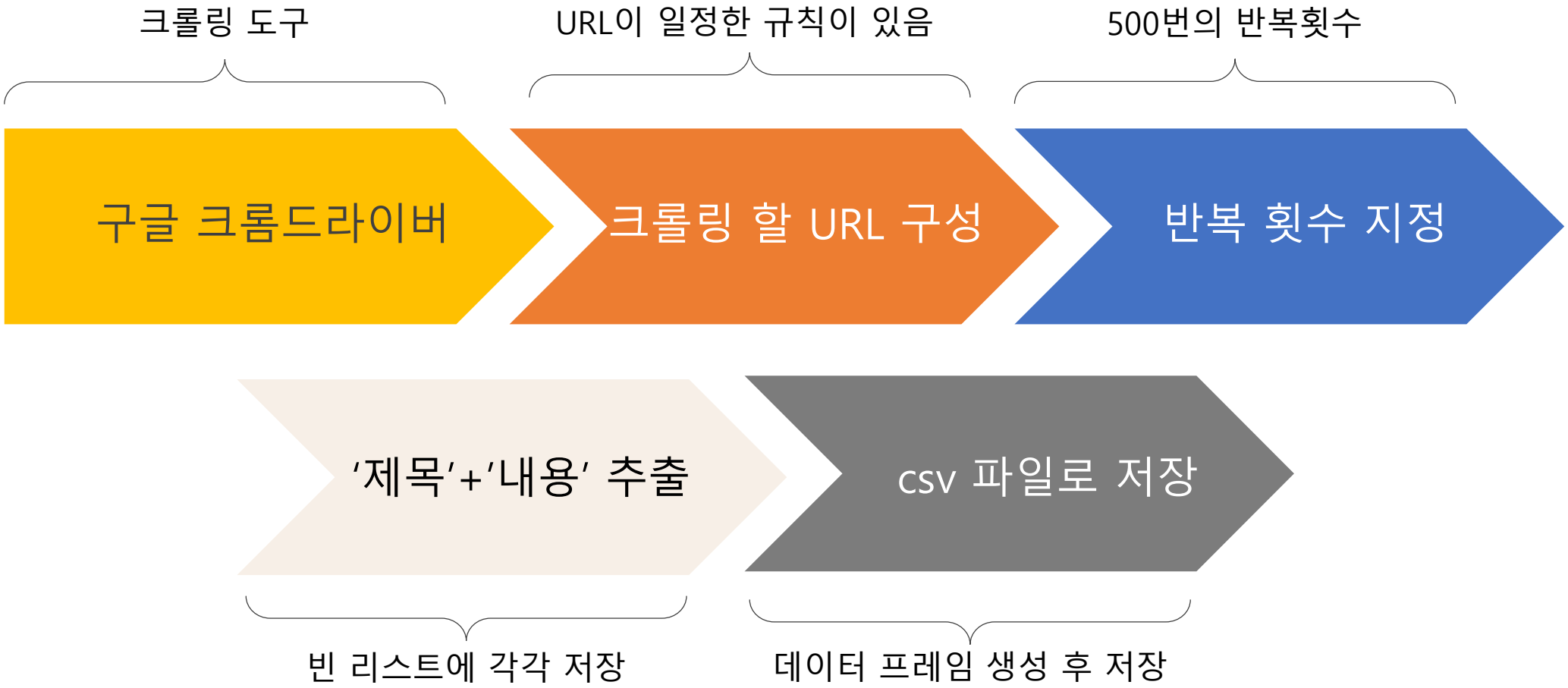
# 모델 구현(S2S)

- 주제: S2S를 이용하여 뉴스 제목 추출하기



# 크롤링

- '보안뉴스'의 최신기사를 바탕으로 크롤링



# 크롤링

- 크롤링한 내용(CSV) ▶ 482개의 데이터 수집 + 50분

| | A | B |
|---|---|---|
| 1 | | 0 |
| 2 | [부정경쟁방지법, 과태료 부과기준 '명확' 해진다] | 부정경쟁방지 및 영업비밀보호에 관한 법률(이하 부정경쟁방지법) 시행령 개정에 '부정경쟁행위 조사 방해 등에 관한 과태료 부과기준'이 명확해진다.<br><br>지식경제부는 부정경쟁방지법 시행령 일부를 개정하는 안을 16일 입법예고했다. 개정안에 따르면 현재 구체적이지 않은 과태료의 부과기준을 행위 정도 및 결과에 따라 네 가지 유형으로 정리했다. 변경 조항은 다음과 같다.<br><br>나. 부정경쟁행위 조사 방해 등에 관한 과태료 부과기준 명확화(안 제6조 제3항)<br>(1) 현행 조문은 과태료의 부과기준을 구체적으로 규정하고 있지 않아 행정행위의 투명성과 예측가능성을 저해하고 있으며, 이는 국민의 의무와 직결되는 사항이므로 |
| 3 | [녹소연 "이통사 개인정보 유용행위 내용 공개해야"] | 이동통신사의 불합리한 통화료 과금체계와 개인정보유출 사건에 대한 감사원의 보고에 대해 녹색소비자연대는 "요금체계와 수준에 대한 평가내용, 개인정보 유용행위 내용을 공개하라"고 주장했다.<br><br>녹소연은 16일 감사원의 (구)정통부 통신위원회에 대한 감사보고서 내용을 조목조목 분석하며 방송통신위원회에 적극 조치를 요구하는 성명서를 발표했다. 녹소연이 주장한 내용은 크게 두 가지다. 요금인가제도 존치・요금체계와 요금수준 평가내용 공개, 이통사의 개인정보 유용행위 수사와 내용 공개다.<br><br>녹소연은 "감사 보고서가 이통사들이 한해 평균 1조원 이상의 요금인하 여력을 갖고 있을 만큼 독과점 요금수준을 유지하고 있음을 지적했다"며 "이는 요금인가제도 때문이 아니라 요금인가제도를 악용한 민관유착의 그릇된 규제행태에 기인한 것"이라 |

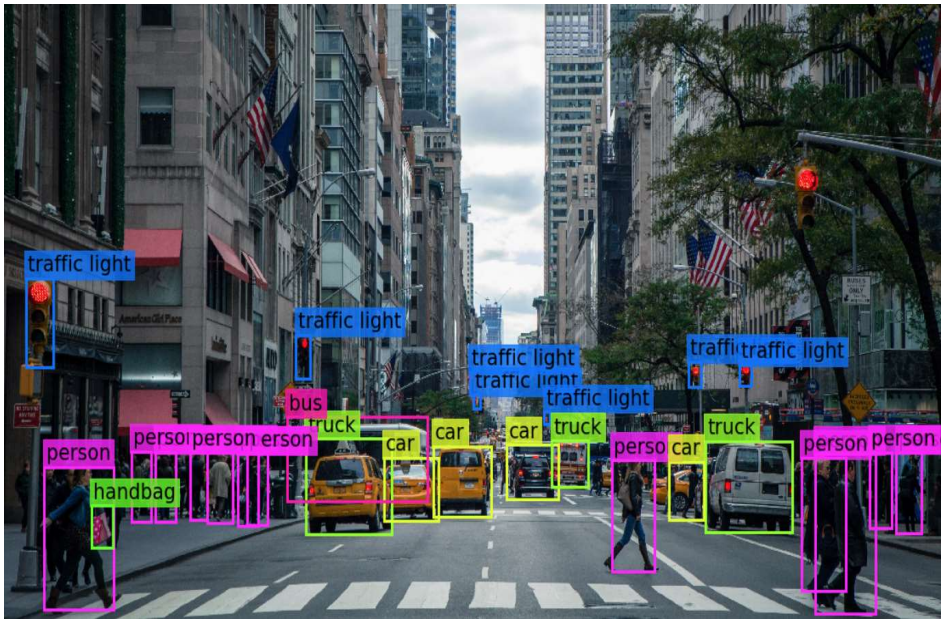
# 학습 및 결과

- 학습내용: 3-layers, encoder_size(100개), decoder_size(20개), GRU(300차원), batch_size(16)
- 데이터 개수: 482개의 기사 제목 및 내용 / 단어 개수: 23738개
- 학습 및 예측 시간: 약 4시간

```
----
 step : 18440
 time : 14744312165.150423
 LOSS : 3.352949446346611
 예측 : 보 이 시 파 보 공 < < <
 손질한 정답 : 보 이 시 일 무 보 시 < <
 정답 : 보안사고 이제 시작 일뿐 무선랜 보안대책 시급
----

----
 step : 18450
 time : 14744312281.49924
 LOSS : 11.413787854434121
 예측 : 데 대 내 오 오 서 < 오 <
 손질한 정답 : 데 소 내 봄 오 < < < <
 정답 : 데이터 소거기 내수시장 봄 오나
----

----
 step : 18460
 time : 14744312396.87547
 LOSS : 5.458987457805779
 예측 : 금 대 홈 홈 < < < < <
 손질한 정답 : 금 대 무 해 < < < < <
 정답 : 금융기관 대출정보 무더기 해킹
----

----
 step : 18470
 time : 14744312515.001822
 LOSS : 11.11860588152958
 예측 : 경 모 초 설 체 < < < <
 손질한 정답 : 경 모 초 설 < < < < <
 정답 : 경기도 모든 초중고교에 설치
----

----
 step : 18480
 time : 14744312632.117226
 LOSS : 2.6422752233200297
 예측 : 마 리 문 < < < < < <
 손질한 정답 : 마 리 인 < < < < < <
 정답 : 마켓 리더로 인정
----

----
 step : 18490
 time : 14744312748.179256
 LOSS : 3.495760653997422
 예측 : 보 제 회 해 모 < < < <
 손질한 정답 : 보 제 회 논 모 < < < <
 정답 : 보안인증사무국 제 회 논문 모집
----
```

# Object Detection(객체 검출)

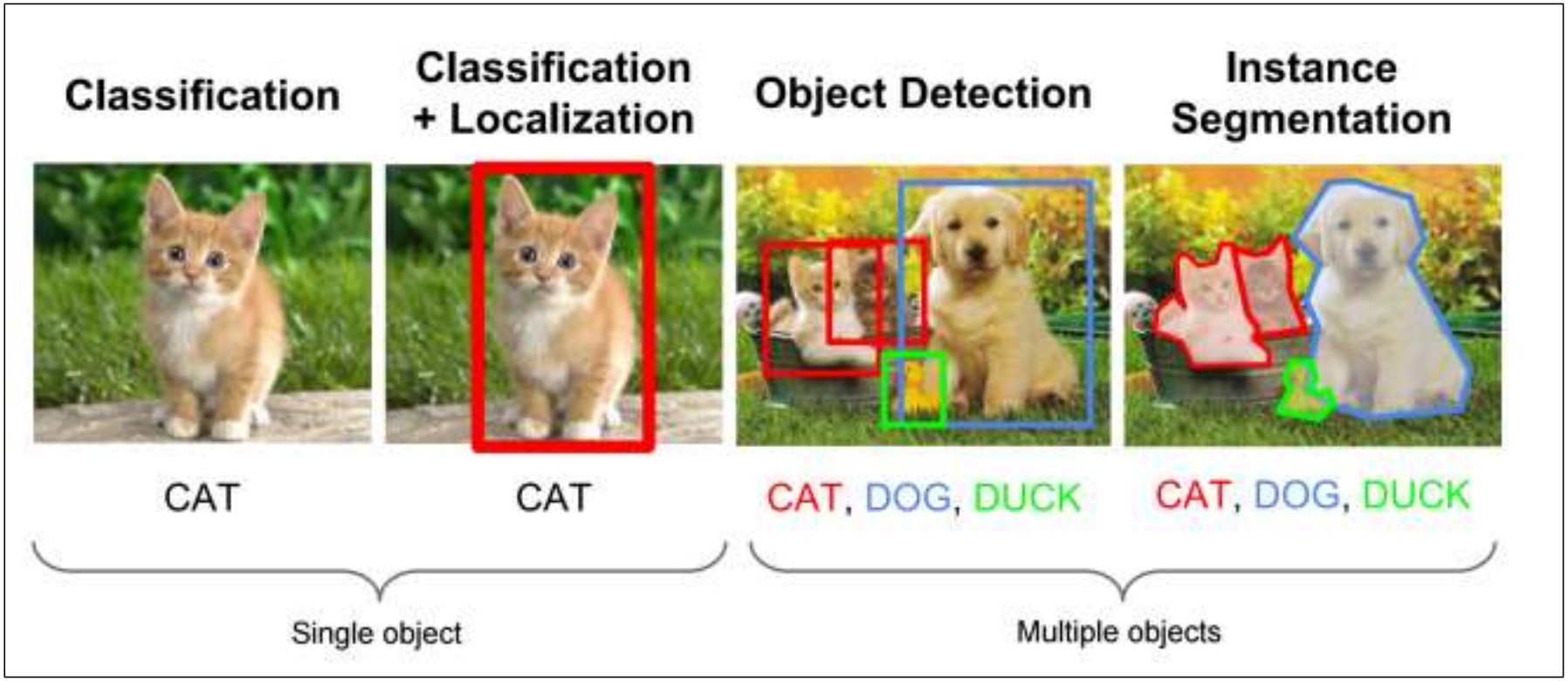
허지혜




http://mrkim.cloudy.so/board_DJpE79/174208

https://github.com/jihyeheo/People_Countiong_Object_Detection

# Object Detection(객체 검출)

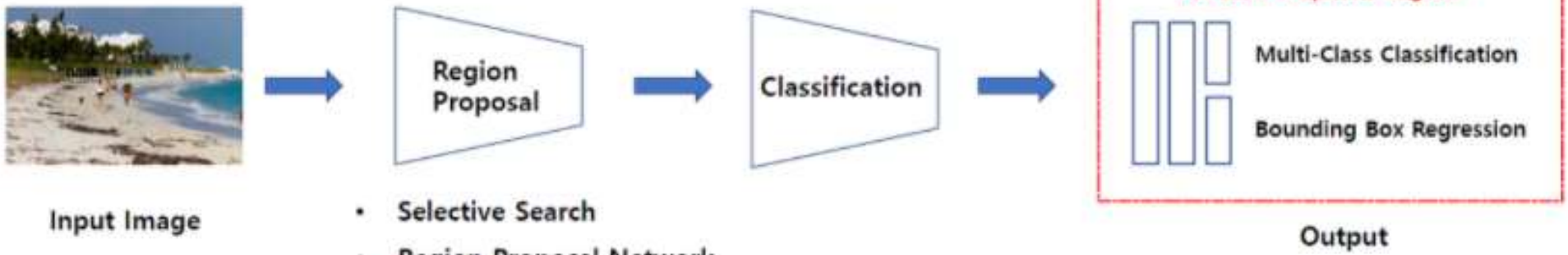



Multi-label **Classification** + Bounding Box Regression(**Localization**)

# Object Detection(객체 검출)

## 1-stage Detector

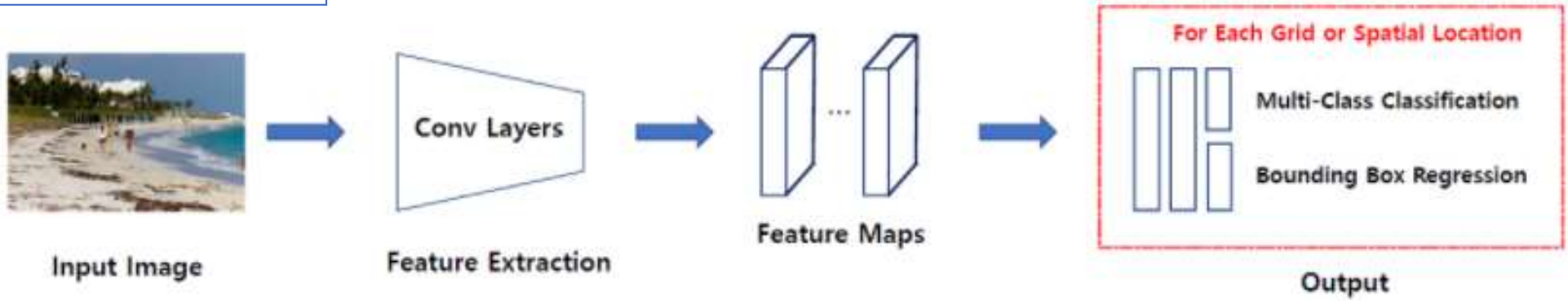
YOLO, SSD 등등



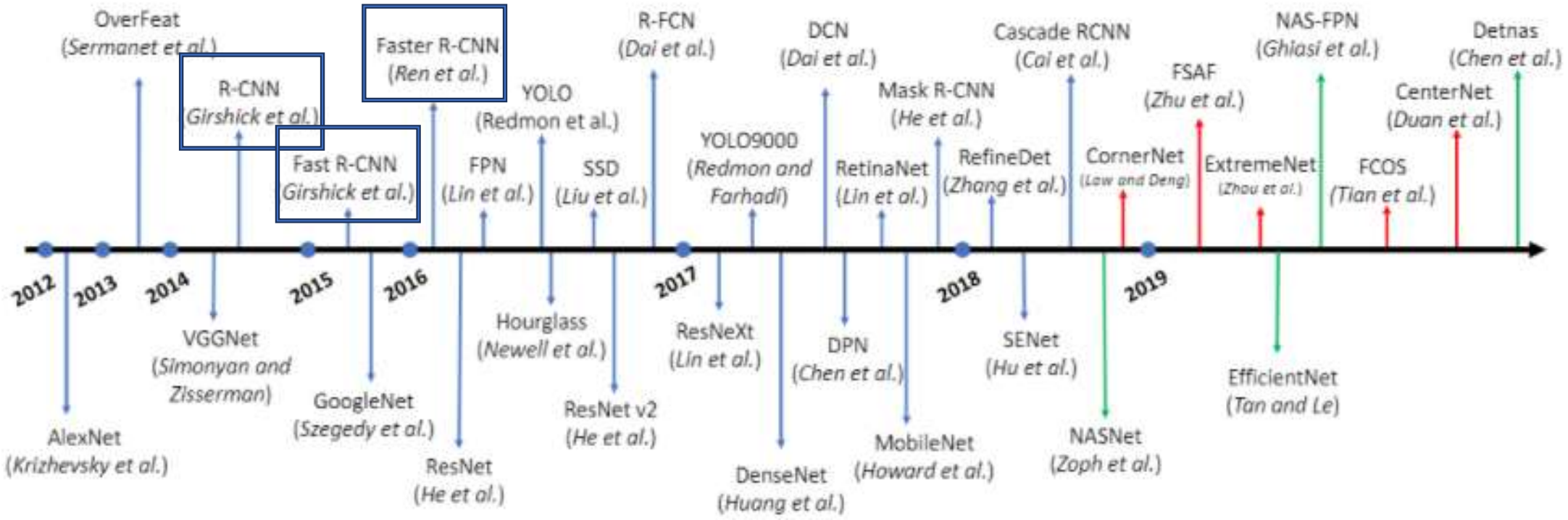

- Selective Search
- Region Proposal Network

## 2-stage Detector

R-CNN, Fast R-CNN 등등

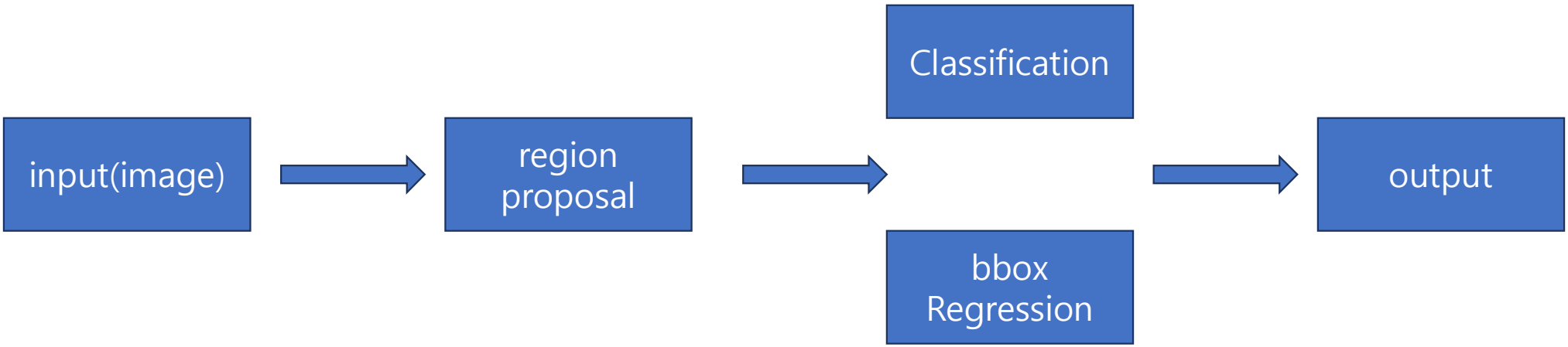


# Object Detection(객체 검출)



# Object Detection(객체 검출)

Architecture

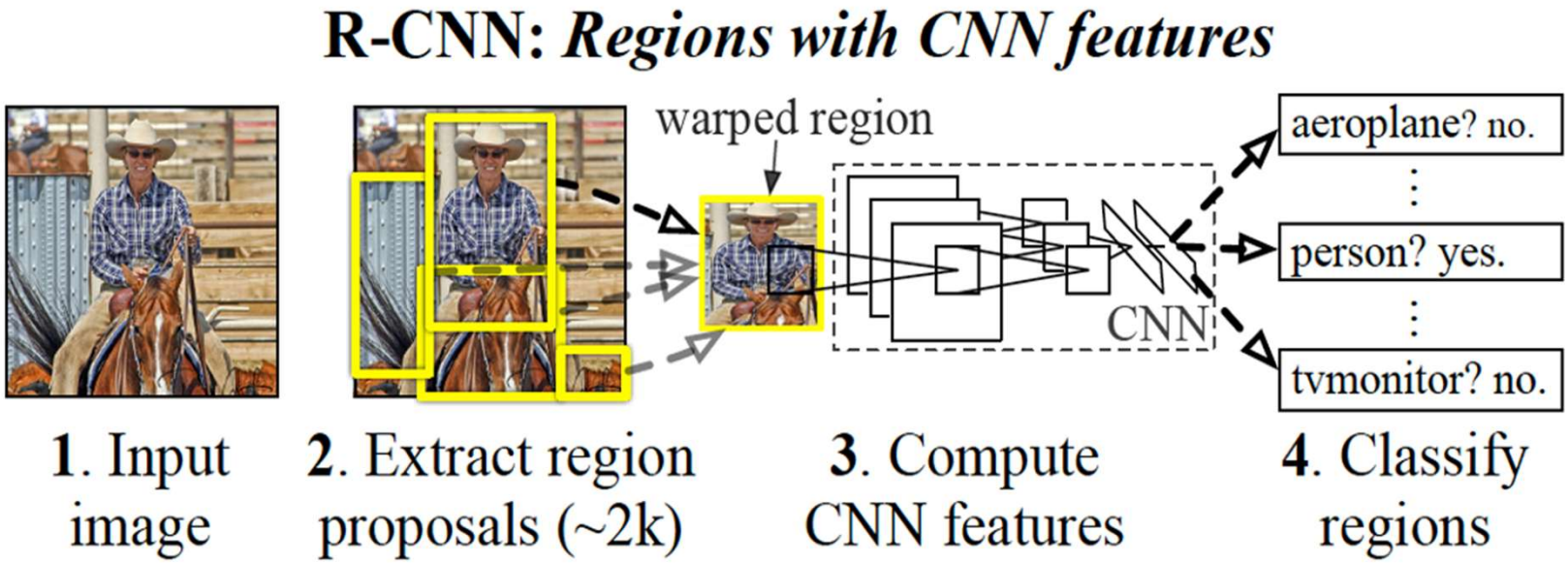


# Object Detection(객체 검출)

R-CNN

Selective Search Algorithm

SVM(Support Vector Machine Classification)

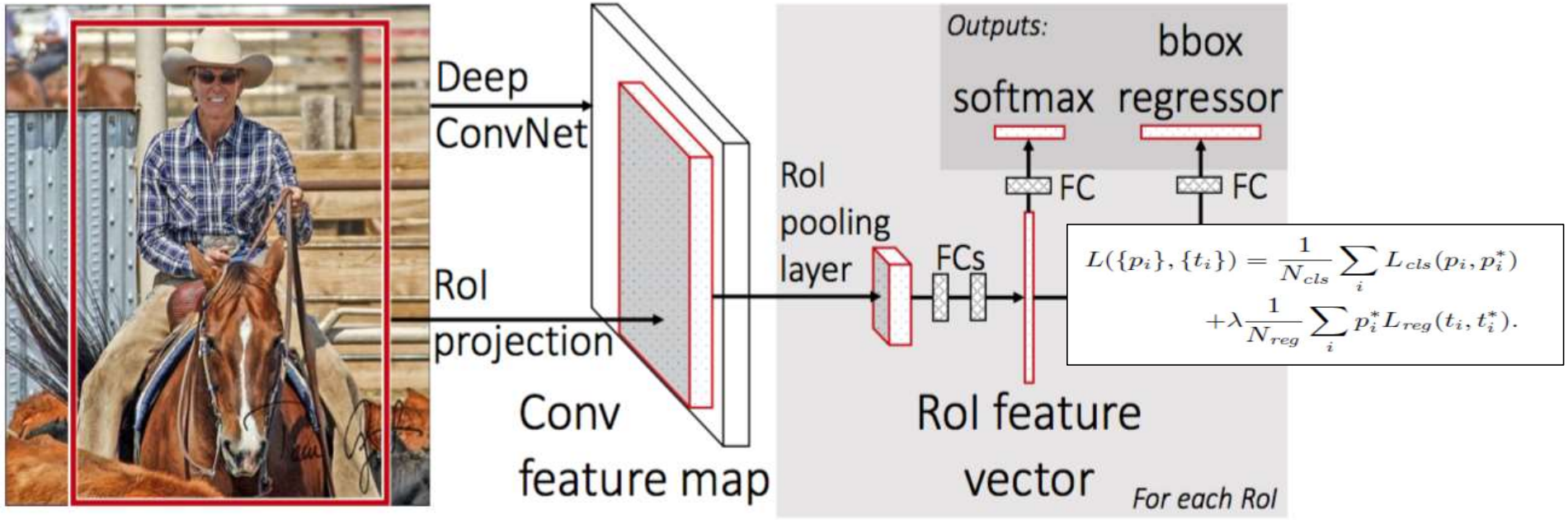



Fine tuning model(AlexNet) + Training

# Object Detection(객체 검출)

## Fast R-CNN

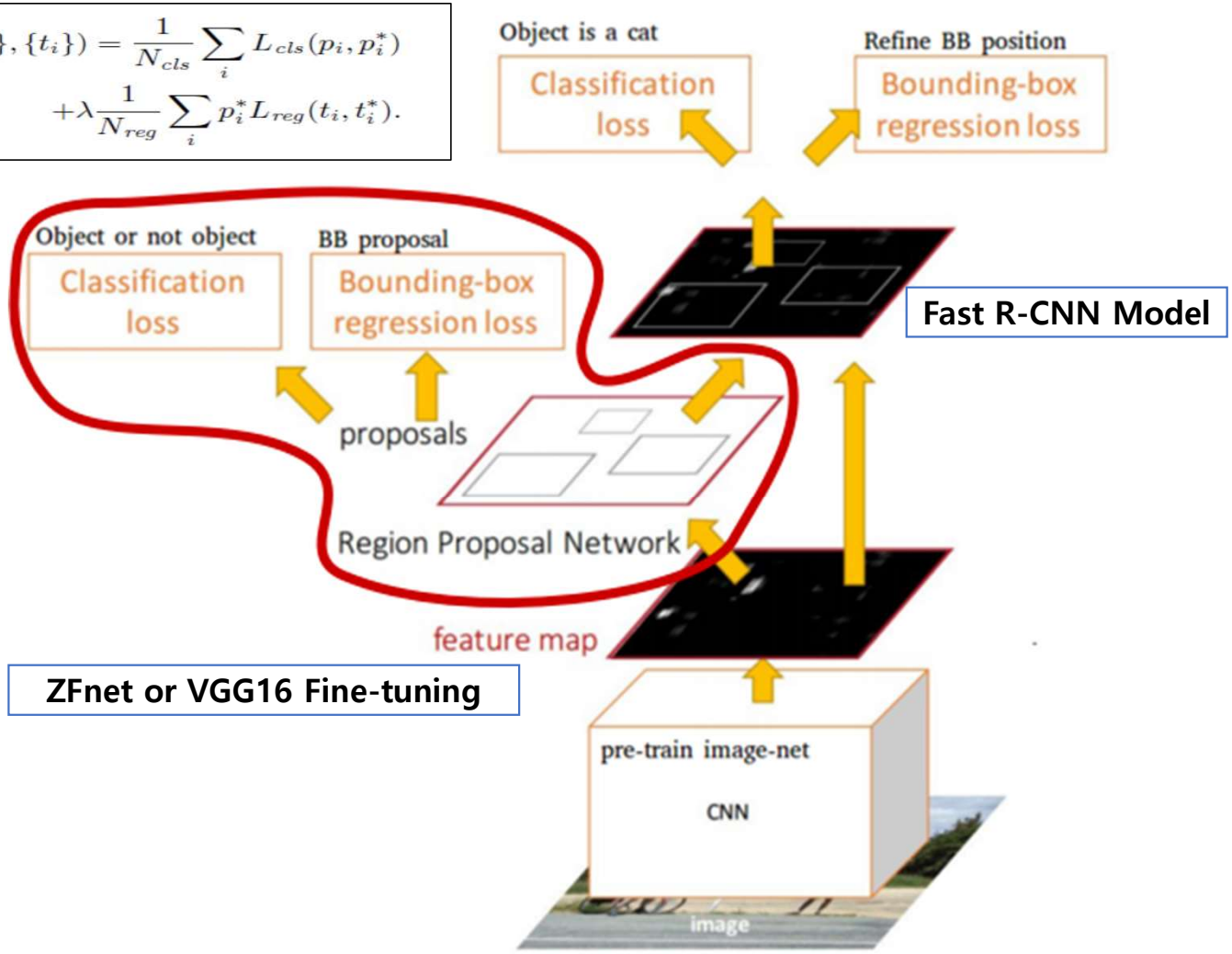
**Selective Search Algorithm**



# Object Detection(객체 검출)

Faster R-CNN

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$



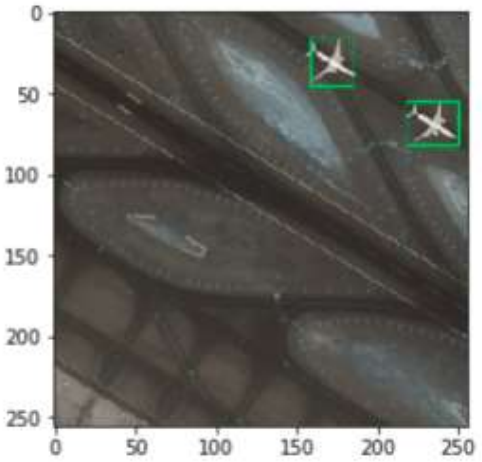

Fast R-CNN Model

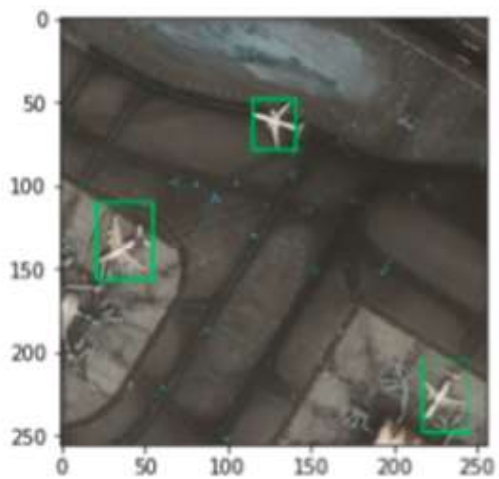
ZFnet or VGG16 Fine-tuning

# Object Detection(객체 검출)

```
for row in df.iterrows():
                x1 = int(row[1][0].split(" ")[0])
                 y1 = int(row[1][0].split(" ")[1])
                 x2 = int(row[1][0].split(" ")[2])
                 y2 = int(row[1][0].split(" ")[3])
                 gtvalues.append({"x1":x1,"x2":x2,"y1":y1,"y2":y2})
            ss.setBaseImage(image)
            ss.switchToSelectiveSearchFast()
            ssresults = ss.process()
            imout = image.copy()
            counter = 0
            falsecounter = 0
            flag = 0
            fflag = 0
            bflag = 0
            for e,result in enumerate(ssresults):
                if e < 2000 and flag == 0:
                    for gtval in gtvalues:
                        x,y,w,h = result
                        iou = get_iou(gtval,{"x1":x,"x2":x+w,"y1":y,"y2":y+h})
                        if counter < 30:
                            if iou > 0.70:
                                timage = imout[y:y+h,x:x+w]
                                resized = cv2.resize(timage, (224,224),
interpolation = cv2.INTER_AREA)
                                train_images.append(resized)
                                train_labels.append(1)
                                counter += 1
```

```
from keras.applications.vgg16 import VGG16
vggmodel = VGG16(weights='imagenet', include_top=True)

for layers in (vggmodel.layers)[:15]:
print(layers)
    layers.trainable = FalseX= vggmodel.layers[-2].output
predictions = Dense(2, activation="softmax")(X)
model_final = Model(input = vggmodel.input, output = predictions)
opt = Adam(lr=0.0001)
model_final.compile(loss = keras.losses.categorical_crossentropy, optimizer = opt, metrics=["accur
acy"])
model_final.summary()
```




https://towardsdatascience.com/step-by-step-r-cnn-implementation-from-scratch-in-python-e97101ccde55

```
model.train()
model.to(device)
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.01, momentum=0.9, weight_decay=0.000
01)
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=1, gamma=0.5
)
num_epochs = 5
loss_hist = Averager()
itr = 1

for epoch in range(num_epochs):
loss_hist.reset()
    for images, targets, image_ids in train_data_loader:

        images = list(image.to(device) for image in images)
        targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

        loss_dict = model(images, targets)

        losses = sum(loss for loss in loss_dict.values())
        loss_value = losses.item()

        loss_hist.send(loss_value)
        optimizer.zero_grad()
        losses.backward()
        optimizer.step()

        if itr % 50 == 0:
            print(f"Iteration #{itr} loss: {loss_value}")
        itr += 1
    if lr_scheduler is not None:
    lr_scheduler.step()
    print(f"Epoch #{epoch} loss: {loss_hist.value}")
```

```
model =
torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=False,
pretrained_backbone=False)
num_classes = 2 # 1 class (wheat), 0 background
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
model.load_state_dict(torch.load(WEIGHTS_FILE))
#model.eval()
#x = model.to(device)
```

https://www.kaggle.com/heojihye/getting-started-with-object-detection-with-pytorch/

# Object Detection(객체 검출)

Difference

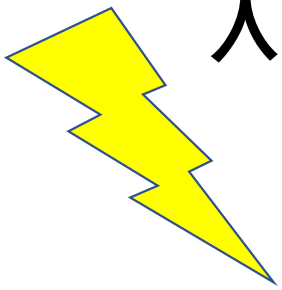
| | R-CNN | Fast R-CNN | Faster R-CNN |
|---|---|---|---|
| **Region proposal (영역 제안)** | selective search | selective search | RPN(Region proposal network) |
| **Loss** | Each loss | multi-task loss | multi-task loss |
| **Disadvantage (단점)** | 1. Long time<br>2. Complex<br>3. Back propagation impossible | 1. Better than R-CNN but takes longer | |

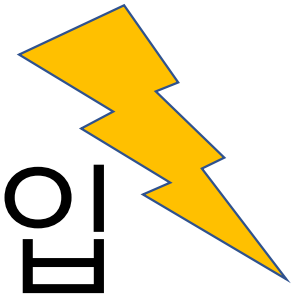
# Object Detection(객체 검출)

**[1]** An overview of gradient descent optimization algorithms, https://ruder.io/optimizing-gradient-descent/
**[2]** Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).
**[3]** Custom-Optimizer-on-Keras, https://github.com/angetato/Custom-Optimizer-on-Keras
**[4]** Ross Girshich, Jeff Donahue, Trevor Darrell, Jitendra Malik et al,
"Rich feature hierarchies for accurate onject detection and semantic segmentation"
**[5]** https://arxiv.org/abs/1311.2524 (R-CNN 논문)
**[6]** https://arxiv.org/abs/1504.08083 (Fast R-CNN 논문 v2)
**[7]** https://arxiv.org/abs/1506.01497 (Faster R-CNN 논문 v3)
**[8]** https://github.com/jihyeheo/Deep_Learning_Paper_review
**[9]** https://github.com/lovit/soy/blob/4c97e35cd78f2079897857c4ad4ec4a4d6a7c0f1/soy/nlp/hangle/_hangle.py
**[10]** https://ratsgo.github.io/natural%20language%20processing/2017/03/12/s2s/

긴 급

# 새 멤버 추가 영입





이런 분이 오시면 좋습니다.

- 대학원 진학 예정
- IT 개발 관련 취업 예정
- 수다쟁이