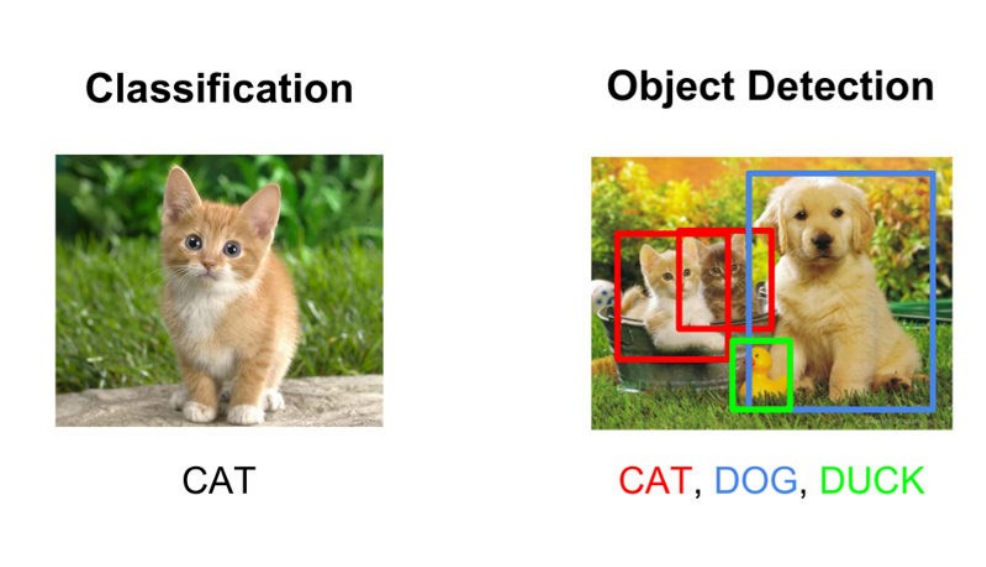
# VISI KOMPUTER

Prof. Dr.Eng. Fitri Utaminingrum, S.T., M.T.

brone.ub.ac.id

# REGION-BASED CNN

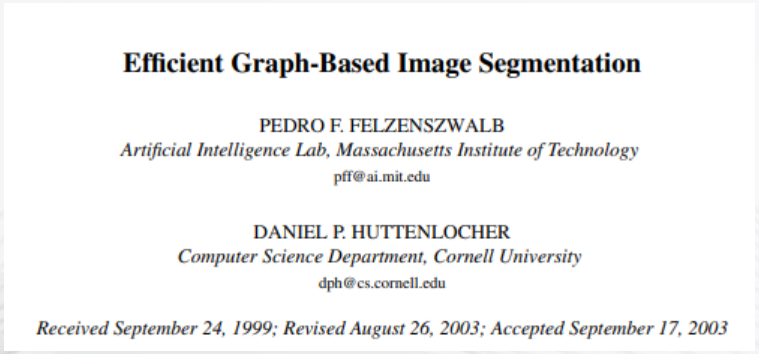



Region-Based CNN adalah pengembangan dari arsitektur CNN yang digunakan untuk melakukan deteksi objek dalam citra beserta dengan lokalisasi objek. Terdapat beberapa varian dari R-CNN yang telah dikembangkan untuk deteksi objek seperti :

- R-CNN
- Fast R-CNN
- Faster R-CNN

# R-CNN

R-CNN pertama kali diusulkan sebagai pengembangan dari AlexNet untuk membuat model deteksi objek berbasis CNN yang lebih efisien dibandingkan metode sebelumnya yang melokalisasikan objek menggunakan *Exhaustive Search.* R-CNN dapat melakukan lokalisasi objek secara lebih efisien dengan menggunakan *Selective Search* untuk membuat RoI (Region of Interest) sebagai kandidat *bounding box* dari objek

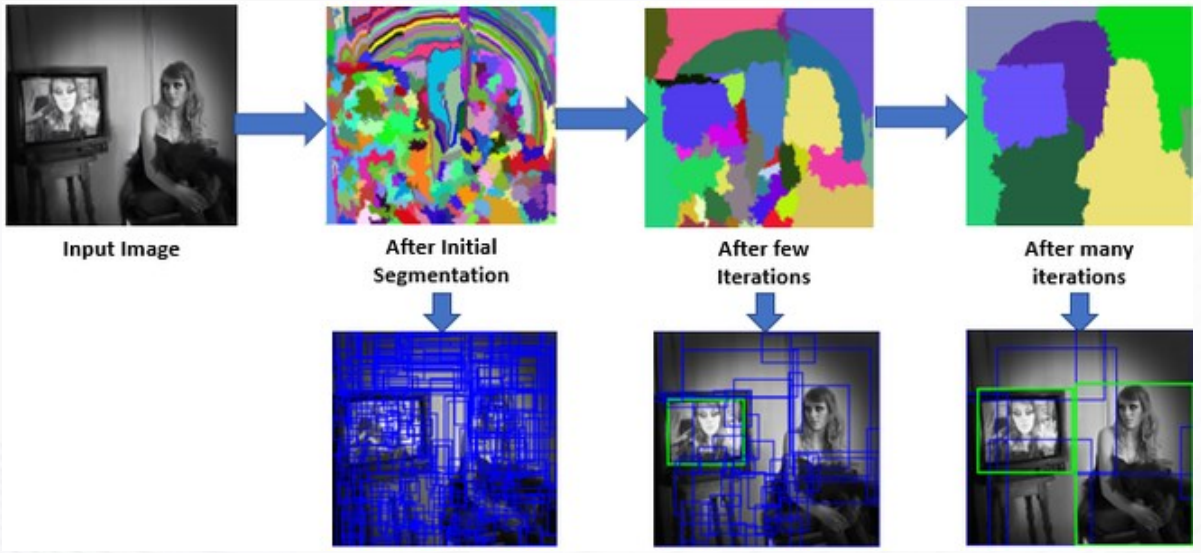


**Efficient Graph-Based Image Segmentation**

PEDRO F. FELZENSZWALB
*Artificial Intelligence Lab, Massachusetts Institute of Technology*
pff@ai.mit.edu

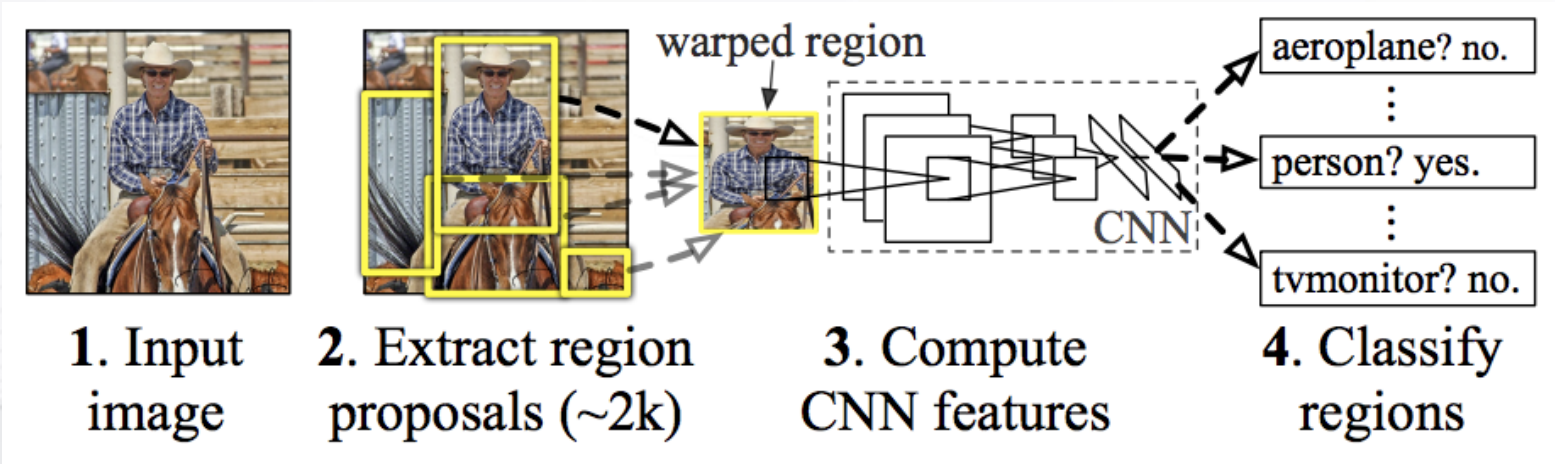
DANIEL P. HUTTENLOCHER
*Computer Science Department, Cornell University*
dph@cs.cornell.edu

*Received September 24, 1999; Revised August 26, 2003; Accepted September 17, 2003*



# R-CNN

Dari *Bounding Box* yang dihasilkan dari *Selective Search* tersebut, setiap RoI (~2000) akan dimasukkan ke *backbone* AlexNet untuk dilakukan ekstraksi fitur. Namun tidak seperti pada AlexNet pada umumnya, klasifikasi citra dilakukan dengan menggunakan SVM.

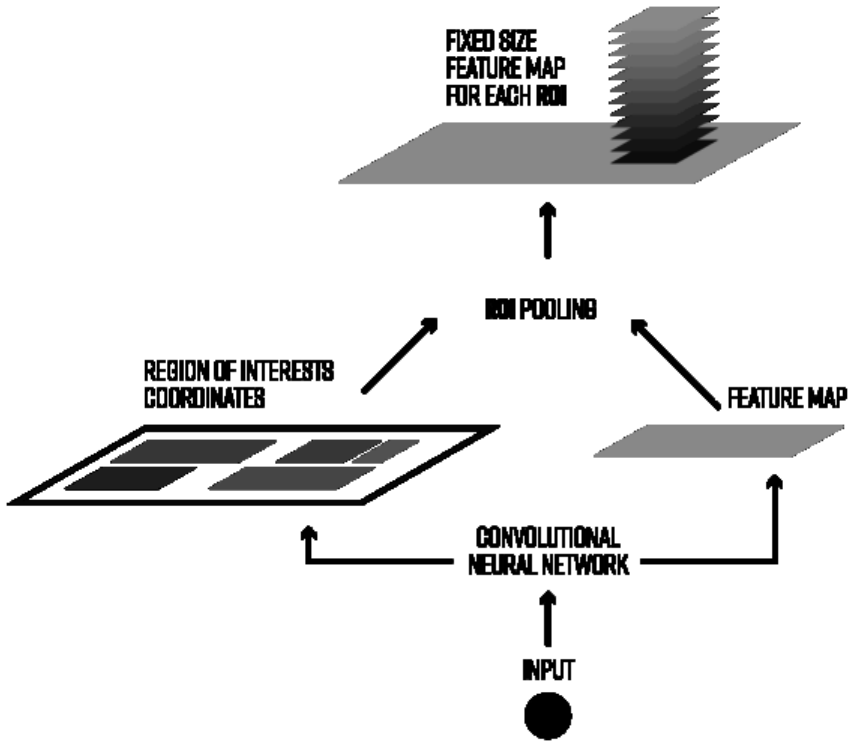


# FAST R-CNN

- Meskipun dengan pengembangan pada R-CNN yang dapat mempercepat proses lokalisasi objek, setiap citra perlu dikomputasi dalam CNN sejumlah dengan banyaknya RoI yang dihasilkan yang umumnya berjumlah sekitar **2000** RoI. Hal ini menyebabkan R-CNN memiliki waktu komputasi yang tinggi sehingga kurang cocok pada pengaplikasian *real-time*.
- Untuk mempercepat proses deteksi objek dari RoI yang dihasilkan, diusulkan metode Fast R-CNN
- Fast R-CNN hanya memerlukan citra dikomputasi dalam CNN sebanyak sekali dengan memanfaatkan layer RoIPooling

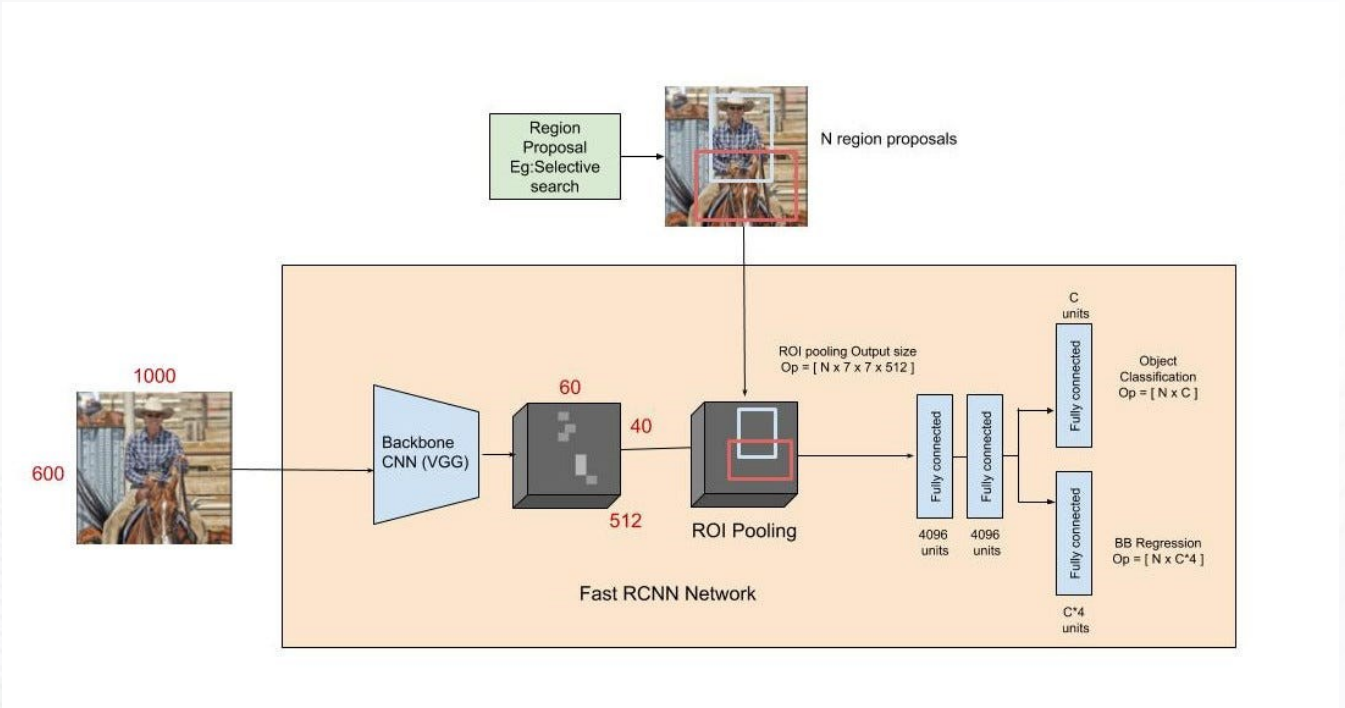
# FAST R-CNN

- RoIPooling menghasilkan feature r
  dari setiap objek dengan memoton
  feature map yang dihasilkan dari
  keseluruhan citra dengan RoI yang
  dihasilkan dari *Selective Search*.
- Hasil pemotongan kemudian di*poo*
  untuk menyesuaikan ukuran featur
  yang dihasilkan



# FAST R-CNN

- Hasil feature map dari setiap RoI kemudian diklasifikasikan dengan menggunakan *Fully-Connected* layer dengan aktivasi softmax
- Secara keseluruhan, dengan optimasi pada bidang ekstraksi fitur dan klasifikasi Fast R-CNN jauh lebih cepat dibandingkan R-CNN

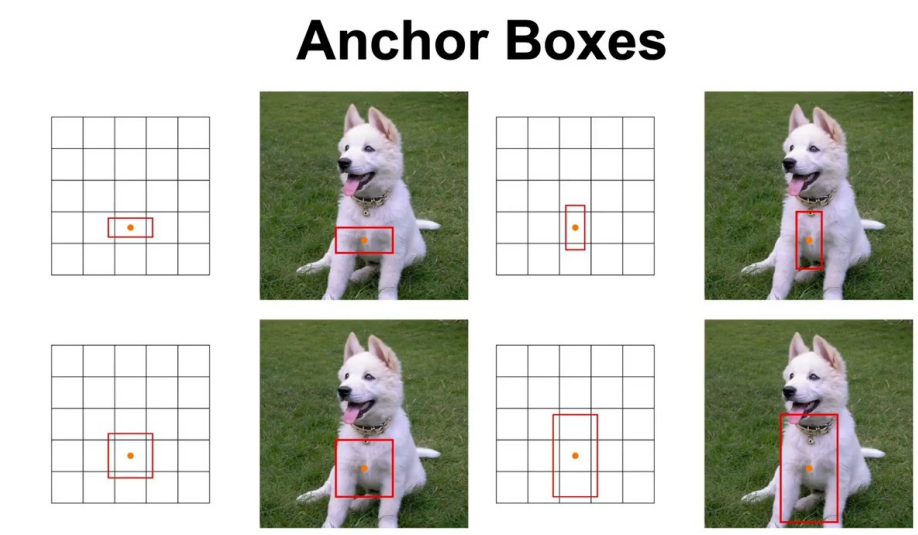


# FASTER R-CNN

- Meskipun Fast R-CNN sangat cepat dibandingkan dengan R-CNN, penggunaan *Selective Search* sebagai metode ekstraksi RoI masih menjadi permasalahan dikarenakan waktu dan kompleksitas komputasinya yang tinggi.
- Untuk mengatasi permasalahan ektraksi RoI tersebut, diusulkan metode Faster R-CNN yang menggunakan Region Proposal Network (RPN) untuk melakukan ekstraksi RoI

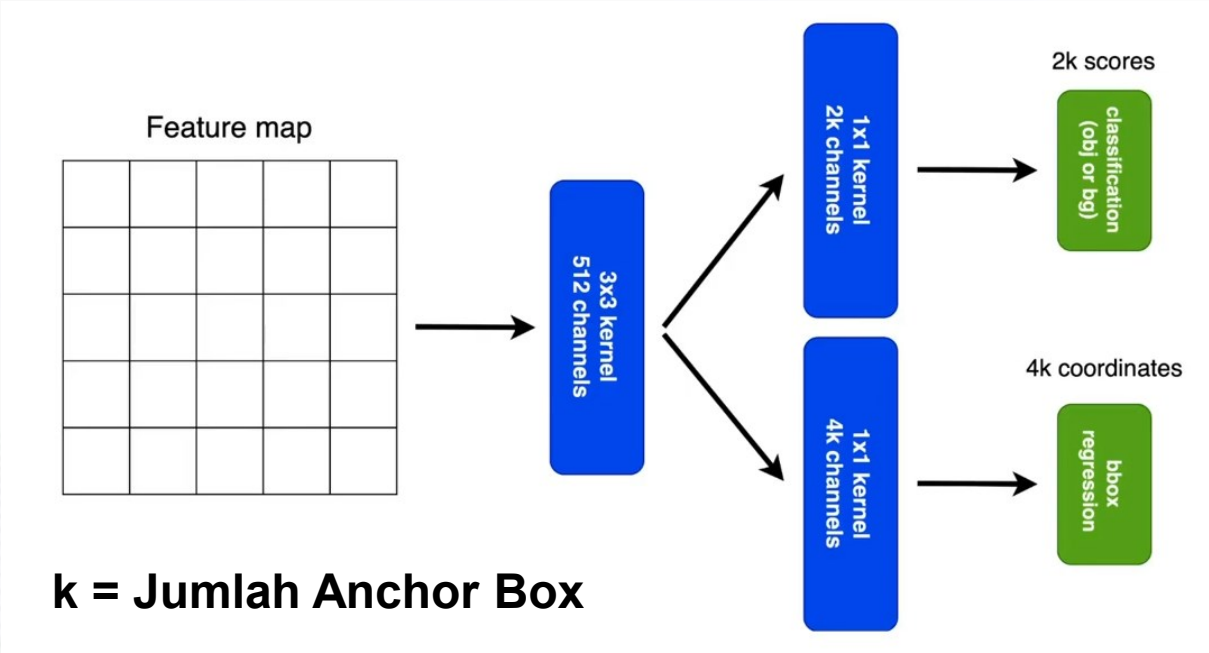
# FASTER R-CNN

- RPN merupakan sebuah *neural network* yang dapat ditambahkan pada CNN untuk melokalisasikan sebuah objek
- RPN memiliki performa yang jauh lebih cepat dibandingkan metode lokalisasi objek lainnya seperti *Selective Search* dan *EdgeBoxes*
- Untuk melokalisasikan objek, RPN menggunakan *Anchor Box* yang diaplikasikan dengan metode sliding window pada setiap posisi di feature map



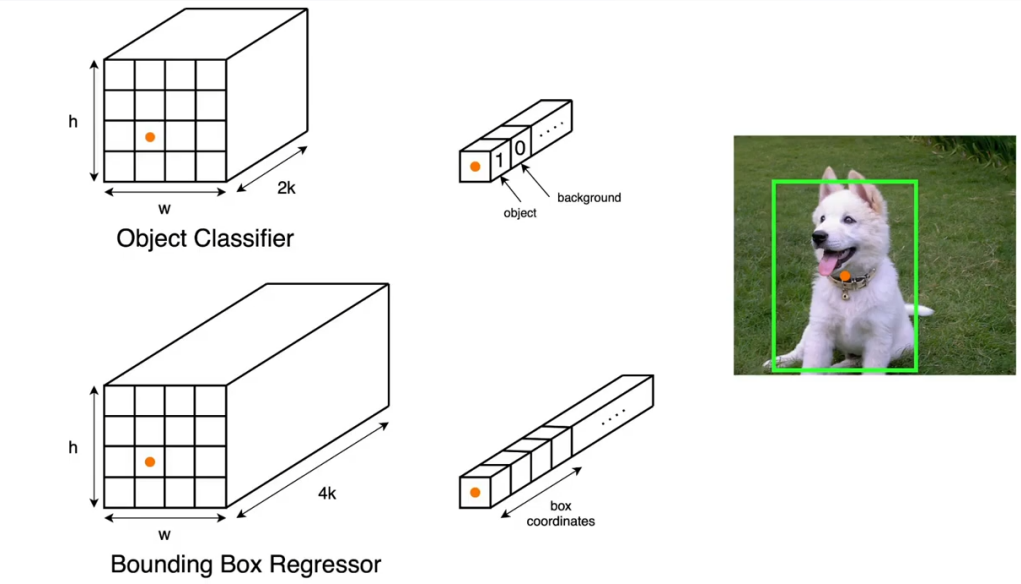
# FASTER R-CNN

- RPN akan mengambil input berupa output dari *backbone* yang digunakan untuk menghasilkan *region proposal*
- Untuk menghasilkan *region proposal* tersebut, RPN dibagi menjadi komponen klasifikasi yang menentukan apakah terdapat objek pada posisi tersebut dan koordinat yang berfungsi menentukan koordinat dan ukuran objek



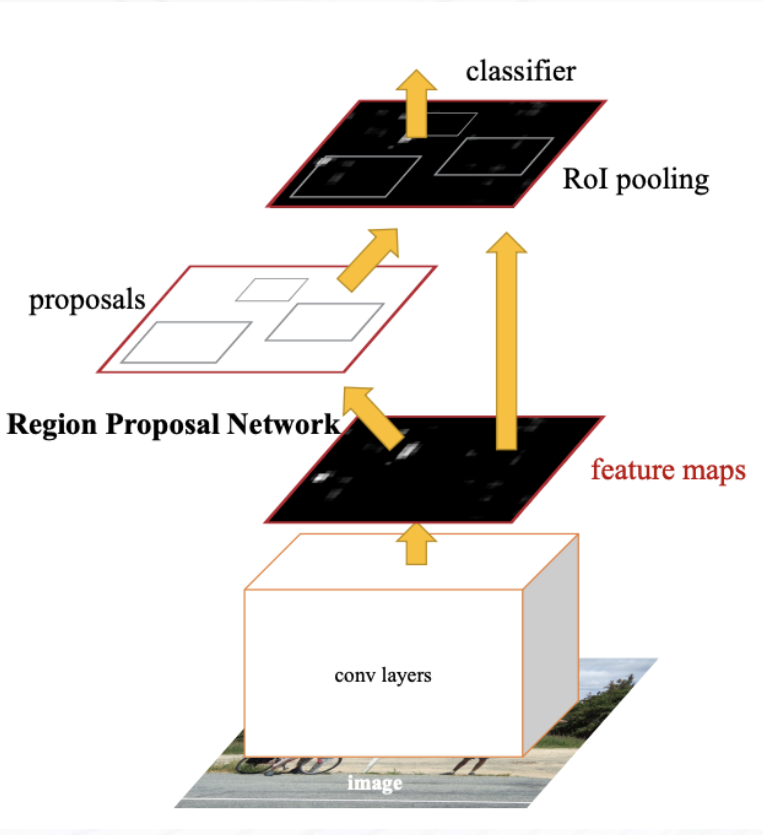
# FASTER R-CNN

- Hasil output dari RPN akan berupa 2 matriks 3 dimensi. Satu untuk menentukan apakah RoI dalam Anchor Box berupa objek atau tidak dan matriks untuk menentukan koordinat dan ukuran spesifik dari objek dalam citra.
- Lebar dan tinggi pada hasil matriks 3 dimensi tersebut akan menunjukkan Anchor Box pada citra, kemudian *depth* akan menandakan Anchor Box yang digunakan.

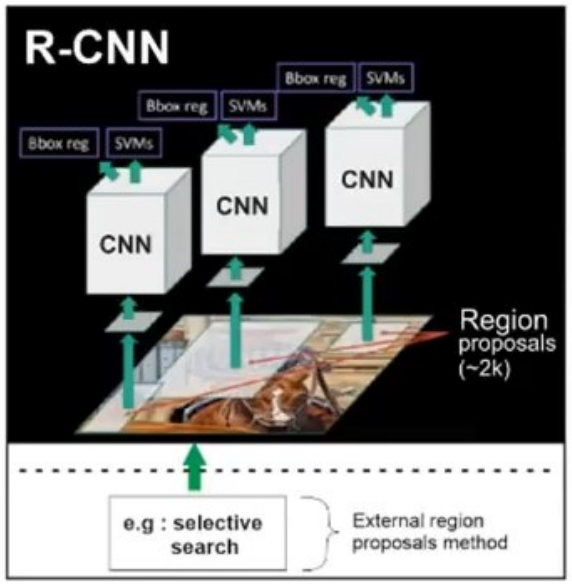


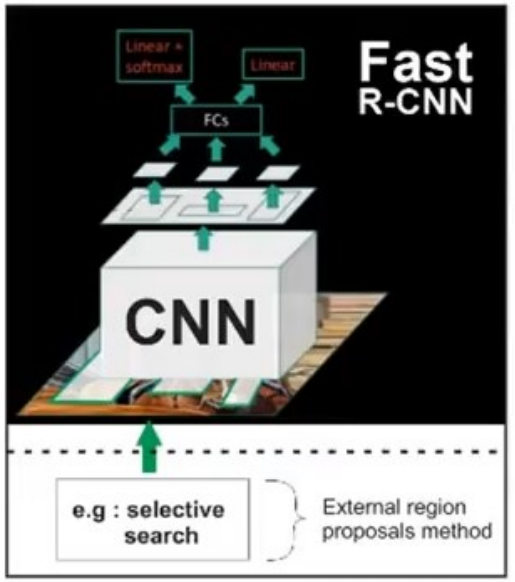
# FASTER R-CNN

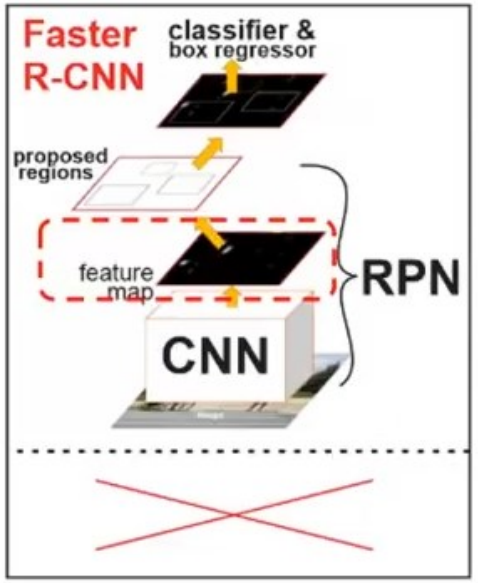
- Secara keseluruhan, Arsitektur Faster R-CNN mirip dengan arsitektur Fast R-CNN dengan perubahan pada metode untuk menghasilkan RoI
- Perubahan metode ekstraksi RoI dari *Selective Search* ke RPN membantu Faster R-CNN menjadi jauh lebih cepat dibandingkan R-CNN dan Fast R-CNN



# PERBANDINGAN R-CNN




| | R-CNN | Fast R-CNN | Faster R-CNN |
|---|---|---|---|
| Test time per image | 50 seconds | 2 seconds | 0.2 seconds |
| Speed-up | 1x | 25x | 250x |
| mAP (VOC 2007) | 66.0% | 66.9% | 66.9% |

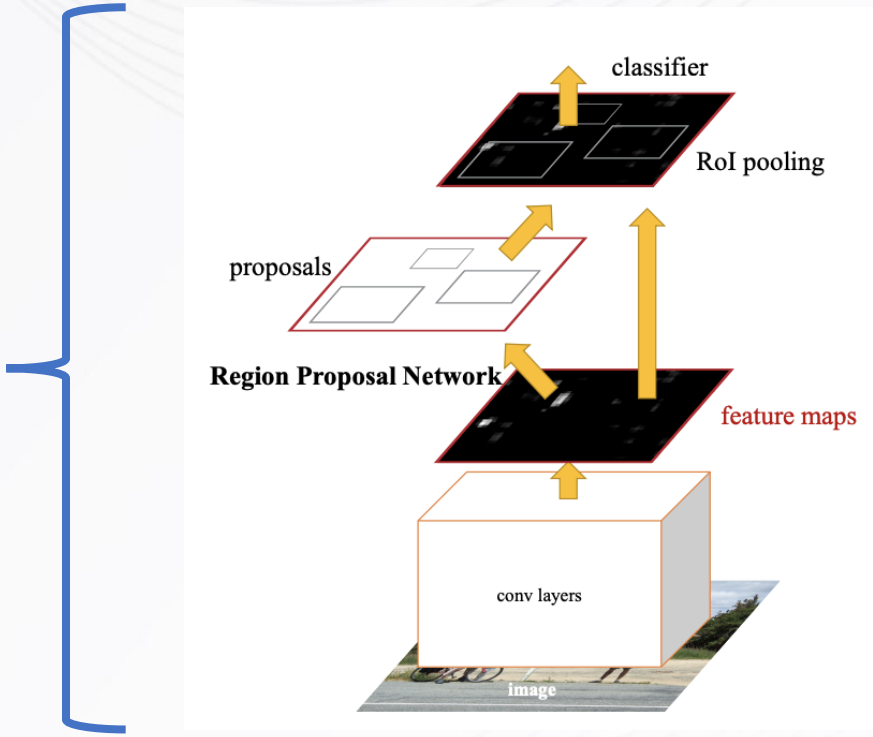
(Li & Karphaty, 2016)

# IMPLEMENTASI R-CNN

- R-CNN dapat diimplementasikan menggunakan berbagai *backbone* AlexNet, VGG, ResNet, Swin Transformer, dll.
- Sebagai contoh, akan diimplementasikan Faster R-CNN dengan menggunakan *backbone* ResNet50 dan library PyTorch

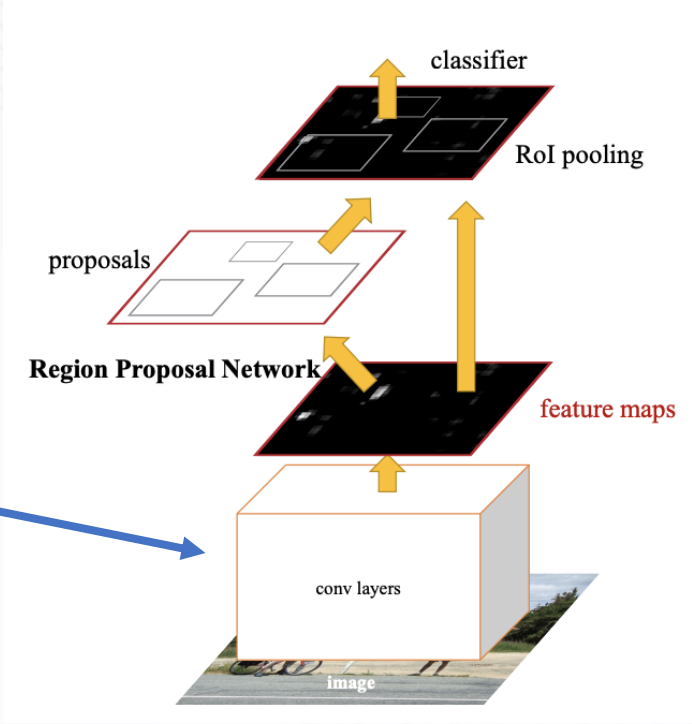
# IMPLEMENTASI R-CNN

```python
class FasterRCNN(nn.Module):
    def __init__(self, backbone, rpn_head, roi_pool, predictor):
        super(FasterRCNN, self).__init__()
        self.backbone = backbone
        self.rpn_head = rpn_head
        self.roi_pool = roi_pool
        self.predictor = predictor

    def forward(self, images, targets=None):
        feature_map_size = (50, 50)
        scales = [8, 16, 32]
        ratios = [0.5, 1, 2]
        stride = 16
        anchors = generate_anchors(scales, ratios, ratios, stride)
        features = self.backbone(images)
        rpn_logits, rpn_bbox_pred = self.rpn_head(features)
        proposals = generate_proposals(rpn_bbox_pred, anchors)
        pooled_features = self.roi_pool(features, proposals)
        scores, bbox_deltas = self.predictor(pooled_features)

        if self.training:

            rpn_loss = rpn_loss(rpn_logits, rpn_bbox_pred, targets['labels'], targets['boxes'])


            return rpn_loss
        else:

            return scores, bbox_deltas, proposals
```



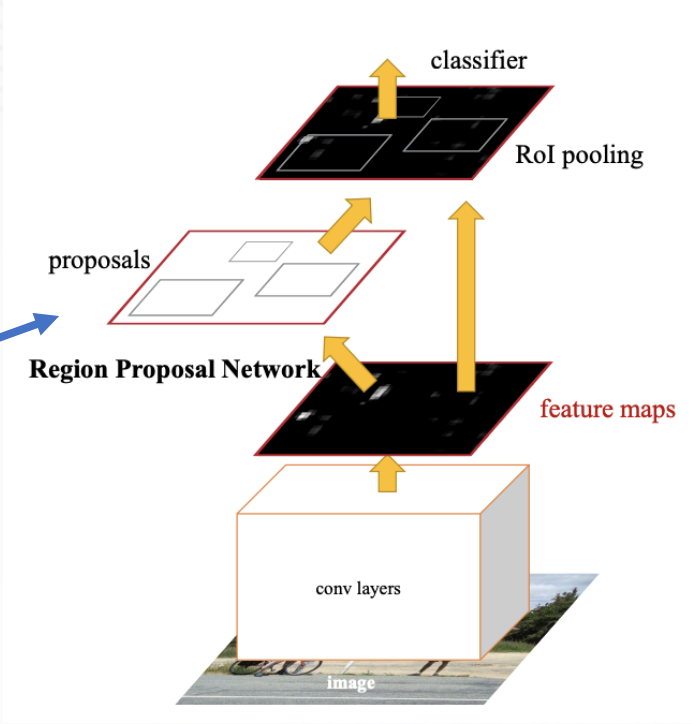
# IMPLEMENTASI R-CNN

```python
class Backbone(nn.Module):
    def __init__(self):
        super(Backbone, self).__init__()
        resnet = models.resnet50(pretrained=False)
        self.backbone = nn.Sequential(*list(resnet.children())[:-3])
        self.out_channels = 1024

    def forward(self, x):
        return self.backbone(x)
```



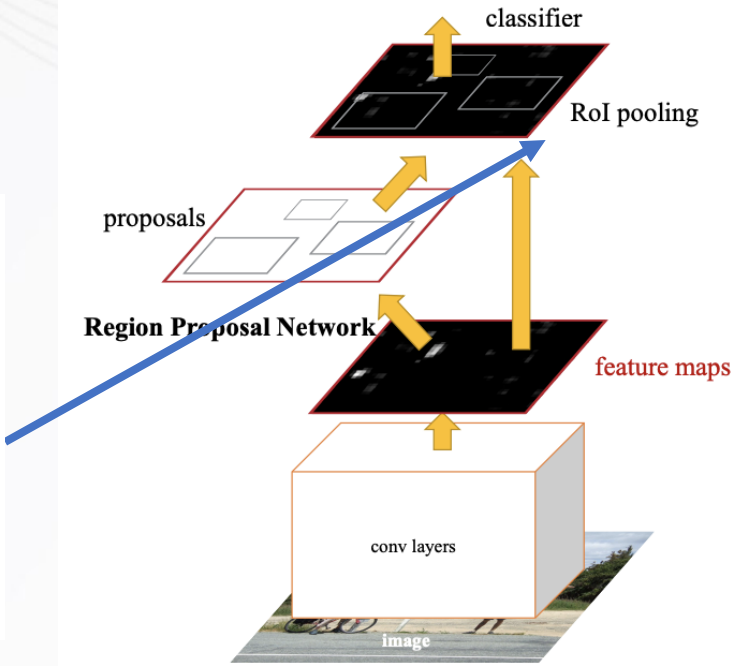
# IMPLEMENTASI R-CNN

```python
class RPNHead(nn.Module):
    def __init__(self, in_channels, num_anchors):
        super(RPNHead, self).__init__()
        self.conv = nn.Conv2d(in_channels, in_channels, kernel_size=3, stride=1, padding=1)
        self.cls_logits = nn.Conv2d(in_channels, num_anchors, kernel_size=1, stride=1)
        self.bbox_pred = nn.Conv2d(in_channels, num_anchors * 4, kernel_size=1, stride=1)

    def forward(self, x):
        x = self.conv(x)
        logits = self.cls_logits(x)
        bbox_pred = self.bbox_pred(x)
        return logits, bbox_pred
```



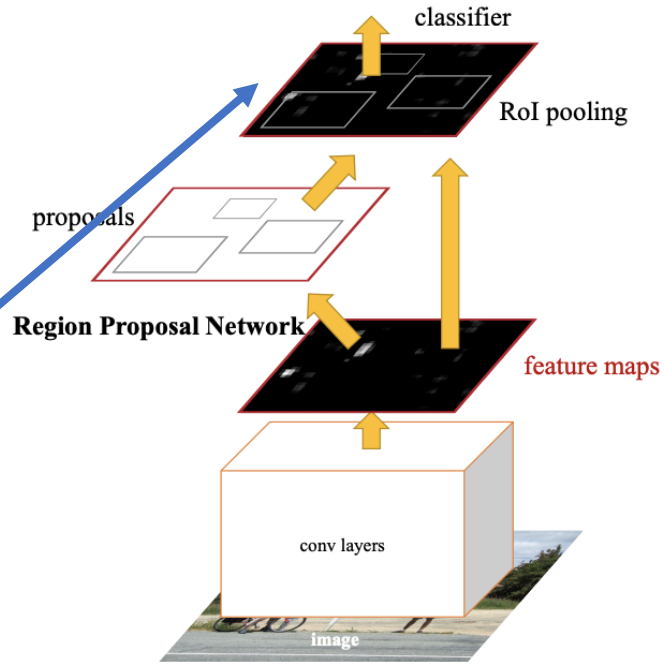
# IMPLEMENTASI R-CNN

```python
class RoIPool(nn.Module):
    def __init__(self, output_size, spatial_scale):
        super(RoIPool, self).__init__()
        self.output_size = output_size
        self.spatial_scale = spatial_scale

    def forward(self, features, proposals):

        pooled_features = []
        for proposal in proposals:
            roi = proposal * self.spatial_scale
            roi_feature = F.adaptive_max_pool2d(features[:, :, int(roi[1]):int(roi[3]), int(roi[0]):int(roi[2])], self.output_size)
            pooled_features.append(roi_feature)
        return torch.stack(pooled_features)
```



# IMPLEMENTASI R-CNN

```python
class FastRCNNPredictor(nn.Module):
    def __init__(self, in_channels, num_classes):
        super(FastRCNNPredictor, self).__init__()
        self.fc1 = nn.Linear(in_channels * 7 * 7, 1024)
        self.fc2 = nn.Linear(1024, 1024)
        self.cls_score = nn.Linear(1024, num_classes)
        self.bbox_pred = nn.Linear(1024, num_classes * 4)

    def forward(self, x):
        x = x.flatten(start_dim=1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        scores = self.cls_score(x)
        bbox_deltas = self.bbox_pred(x)
        return scores, bbox_deltas
```



# TERIMA KASIH