Road Map

Mimicking Very Efficient Network for Object Detection

http://openaccess.thecvf.com/content_cvpr_2017/papers/Li_Mimicking_Very_Efficient_CVPR_2017_paper.pdf

According to the experiments in [22], the Fast R-CNN [13] with AlexNet trained from scratch gets the 40.4% AP on Pascal VOC 2007, while with ImageNet pre-trained AlexNet gets 56.8% AP. Due to this phenomenon, nearly all the modern detection methods can only train networks which have been trained on ImageNet before and cannot train a network from scratch to achieve comparable results. The result is that we can only use networks designed for classification task such as AlexNet [23], ZFNet [35], VGGNet [30] and ResNet [17], which are not necessarily optimal for detection.

Moreover, in experiments we find that smaller networks always perform poor on ImageNet classification so that fine-tuning them on detection also leads to poor detection performance. However by mimicking method, smaller networks can achieve higher results by estimating complex functions learned by complex networks.

In this paper, we want to train more efficient detection networks without ImageNet pre-training. More importantly, we still need to achieve competitive performance as the large ImageNet pre-trained models. The basic idea is that if we already have a network that achieves satisfying performance for detection, the network can be used to supervise other network training for detection.

Every object detector has a similar structure:

- Extract feature maps from the whole input image
- Use a method in order to decode location from feature map

So a detector can be modeled as:

- A feature extractor
- A feature decoder

This way, mimicking supervision is used for feature extraction in smaller network and ground-truth supervision is used for detection part.

Extensions to feature map mimicking:

1. Mimic across scales: Define a simple transformation to up-sample the feature map to a large scale then mimic the transformed feature map.
2. Extend mimicking technique to a two-stage procedure???

Method	MR_{-2}	Parameters	test time (ms)
Inception R-FCN	7.15	2.5M	53.5
$1/2$-Inception Mimic R-FCN	7.31	625K	22.8
$1/2$-Inception finetuned from ImageNet	8.88	625K	22.8

Table 1: The parameters and test time of large and small models. Tested on TITANX with 1000×1500 input. The $1/2$-Inception model trained by mimicking outperforms that fine-tuned from ImageNet pre-trained model. Moreover, it obtains similar performance as the large Inception model with only $1/4$ parameters and achieves a $2.5\times$ speed-up.

It is stated in paper that: One shot methods such as SSD and YoLo consumes most of the when extracting the features from input image.

The soft targets(logits) carrying helpful information learned by the large complex model allow the small network to approximate the complex functions of the large network.

Summary:

1. Daha küçük bir networkü alıp baştan eğitmek ile başarılı sonuçlar elde edilemiyor.
2. Complex networklerdeki öğrenilmişlik, herhangi bir yöntemle küçük networklere aktarıldığı zaman küçük networkler başarılı sonuçlar üretebiliyor.
3. Object detectorler (including double shot and single shot ones) öyle veya böyle imajdan feature map çıkarıyorlar ve esas zaman bu aşamada harcanıyor. Bu paper'daki esas anlatılmak istenen, feature map extraction ve feature map decoding aşamalarında küçük networkler üretebilmek ve dolayısı ile test zamanında harcanan zamanı minimize etmeye çalışmak.

Mobile Video Object Detection with Temporally-Aware Feature Maps

https://arxiv.org/abs/1711.06368

Summary:

1. Frame by frame detection yapılan methotta temporal information detector tarafından kullanılmıyor.
2. Temporal information kullanabilmek için LSTM kullanmak gerekiyor. Paper'da Bottleneck LSTM diye bir LSTM mimarisi introduce ediliyor.
3. Aynı zamanda base networklerde bir takım computational save'lerden bahsediliyor. Küçük ve verimli networklerin kullanıldığından bahsediliyor.
    a. Küçük networkler:
        i. SqueezeNet
        ii. MobileNet
        iii. Xception
        iv. Shufflenet
    b. Knowledge Distillation
        i. NoScope
4. Approach:
    a. SSD framework with MobileNet architecture
    b. Replace all conv layers in SSD with depth-wise separable networks
    c. Inject LSTMs to single frame detector (Instead separation of detection & LSTM networks, LSTMs are directly injected to the network)

Bu makalede elde edilen başarı oranları promising görünüyor. Burada bahsi geçen Bottleneck LSTM'ler bizim yöntemimize entegre edilebilir.

NoScope: Optimizing Neural Network Queries over Video at Scale

Bu paper aslında tam da benim kafamdaki düşüncelere değinmiş. Temel olarak propose edilen şey bir sistematik ve bu sistematik sonucunda temel olarak binary olarak object var mı yok mu sorusunun cevabı veriliyor.
Bu yöntemi bi deneyip görmek lazım. Ama benim ön görüm bu yöntemin sadece video sequencelarinda target objelerin çok da görülmediği senaryolarda iş yapacağı yönünde. Yani bir mobese kamerasında araba binary classifier i çalıştırıyorsan eğer bu yöntem sanki çok da işe yarayacak gibi durmuyor.
Benim de aslında ilk başta kafamda olan, yani hocaya sunmak istediğim bu şekilde bir framework sağlamak idi. Yani bir metotdan ziyade, specific domainler için, domain-specific tune nasıl yapılır sorusuna cevap veren bir framework yaratmak. Bu açıdan bu paper benim kafamdakini somut olarak ortaya koyuyor.

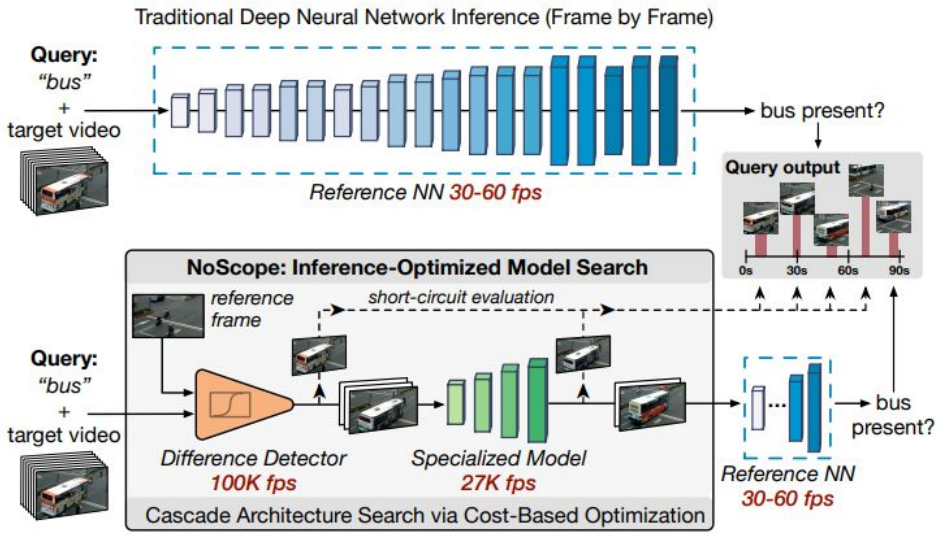

Figure 1: NoScope is a system for accelerating neural network analysis over videos via inference-optimized model search. Given an input video, target object, and reference neural network, NoScope automatically searches for and trains a cascade of models—including difference detectors and specialized networks—that can reproduce the binarized outputs of the reference network with high accuracy—but up to three orders of magnitude faster.

Summary:

1. Sistem kapsamında 3 farklı komponent tanıtılıyor:
    a. Model Specialization -> Bir task için özelleşmiş hızlı çalışan network
        i. Knowledge distillation gibi belirli bir öğrenilmişliği daha küçük bir networke aktarma probleminden ziyade burada güçlü networklerin outputları shallow networkleri eğitmek için kullanılıyor.

b. Difference Detection, Burada da klasik image processing yöntemlerinden türeyen difference detection yerine öğrenen sistemler tasvir edilmiş. Yani yine bir network eğitimi var ve network output olarak kendisine verilen referans frame ile target frame e göre target frame in referans frame ile aynı etikete sahip olup olmadığını hesaplar.
c. Cost based model search: Burada da esas amaç overall da bütün sistemin en iyi sonucu verecek şekilde farklı varyasyonlar arasından en iyinin aranmasıdır.

http://dawn.cs.stanford.edu/2017/06/22/noscope/
https://github.com/stanford-futuredata/noscope

MobileNet, EffNet, XCeption

Bütün bu paperlar depth-wise separable convolutionların kullanılması hakkında farklı farklı mimariler propose ediyorlar.
Burada propose edilen architecture, object detectorlerde kullanılan base feature extractorların computationally olarak daha verimli hale getirilmesinde kullanılabilir.
Temel olarak propose edilen yöntem, depth-wise separable convolution'lar kullanarak AddMultiplication operasyonlarının sayısını azaltmak. Örnek vermek gerekirse, MobileNet

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \quad (5)$$

which is the sum of the depthwise and 1×1 pointwise convolutions.

By expressing convolution as a two step process of filtering and combining we get a reduction in computation of:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

Yukarıda belirtilen oranda computation save ediyor. N filtre sayısı, Dk da kernel size ı.

SqueezeNet

Daha küçük network mimarilerinin tasarlanması için tavsiyeler ve propose edilen katman ile bir network'un tasarlanması ve eğitilmesi.
Burada propose edilen architecture, object detectorlerde kullanılan base feature extractorların computationally olarak daha verimli hale getirilmesinde kullanılabilir.
Bununla birlikte, araştırmalarım doğrultusunda, mevcut olarak bir çok embedded cihaz üzerinde squeezenet implementationları varmış. Mesela Ceva DSP bu network mimarisini kullanıyor.

SSD: Single Shot Detector

Summary:

1. Multi-scale feature maps for detection
2. Convolutional predictors for detection
   For a feature layer of size m × n with p channels, the basic element for predicting parameters of a potential detection is a **3 × 3 × p small kernel that produces either a score for a category, or a shape offset relative to the default box coordinates**. At each of the m × n locations where the kernel is applied, it produces an output value.
3. Default boxes and aspect ratios
   The default boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. At each feature map cell, we predict the offsets relative to the default box shapes in the cell, as well as the per-class scores that indicate the presence of a class instance in each of those boxes. ***Specifically, for each box out of k at a given location, we compute c class scores and the 4 offsets relative to the original default box shape. This results in a total of (c + 4)k filters that are applied around each location in the feature map, yielding (c + 4)kmn outputs for a m × n feature map.***

Yukarıdaki 3 maddede esasen SSD açıklanabiliyor. SSD adından da anlaşılabileceği gibi single shot bir detector. Yani R-CNN gibi proposallar üretip burada classification yapmak yerine (double shot) bütün bu processi tek bir network ile tek forward da çözüyor.

Performans arttırımı nerelerde yapılabilir?
Burada yazılanlar aslında bizim proje boyunca da deneyip raporlayabileceğimiz şeyleri oluşturuyor.

- Base feature extractor daha verimli bir şekilde değiştirilebilir. Standart implementation VGG-16 kullanıyor. Bunun yerine:
  - Depth-wise convolutions kullanan architecture'lara geçilebilir:
    - MobileNet
    - EffNet
    - XCeption
  - SqueezeNet 'deki tavsiyelere uygun Fire(Squeeze Expand) katmanları kullanılabilir. Buradaki 3x3 conv'lar depth-wise 1x1 kullanılabilir.
  - Overall'da elde edilen network'un başarı oranı Knowledge Distillation ile arttırılabilir (**Mimicking Very Efficient Network for Object Detection**).
  - BaseNetworkler'de 1k object classification taski olduğu için, en sondaki FC layer'lardaki parametre sayısı ve dolayısı AddMult operasyonları fazla. Domain-specific bir veya birden fazla classli classifier ihtiyacı olduğu durumda son katmandaki FC boyutları küçültülebilir.

- Base networkten çıkan feature map scale ediliyor. Bu aslında klasik object detectorlardaki pyramid olayı gibi düşünülürse, domain-specific problemlerde, object aspect ratio'suna göre pyramid size küçültülebilir. Örneğin sadece w,h boyutundaki objelerin bulunması gibi
- Aspect ratio'lara göre default box size'ları ayarlanabilir. Specific objeler için aspect ratio'lar aynı olacaktır.
- Yukarıda bahsedilen her iki madde için de convolution based detection da feature map üzerinde uygulanan toplam filtre sayısı düşecektir. Yani yukarıda **k** ile belirtilen parametre azalacaktır. Dolayısı ile toplam uygulanan filtre sayısı azalacaktır. Dolayısı ile de computation azalmış olacaktır.
- ***3 × 3 × p small kernel that produces either a score for a category, or a shape offset relative to the default box coordinates*** burada bahsi edilen convolution'lar depth-wise 'a çevrilebilir. Bu duruma örnek ***Mobile Video Object Detection with Temporally-Aware Feature Maps*** paperinda da kullanılmış.
- Overallda da single image detector unu urettikten sonra LSTM'ler kullanılarak temporal bilgi eklenebilir. (***Mobile Video Object Detection with Temporally-Aware Feature Maps***)