Paper review

Feature Pyramid Networks for Object Detection (CVPR 2017)

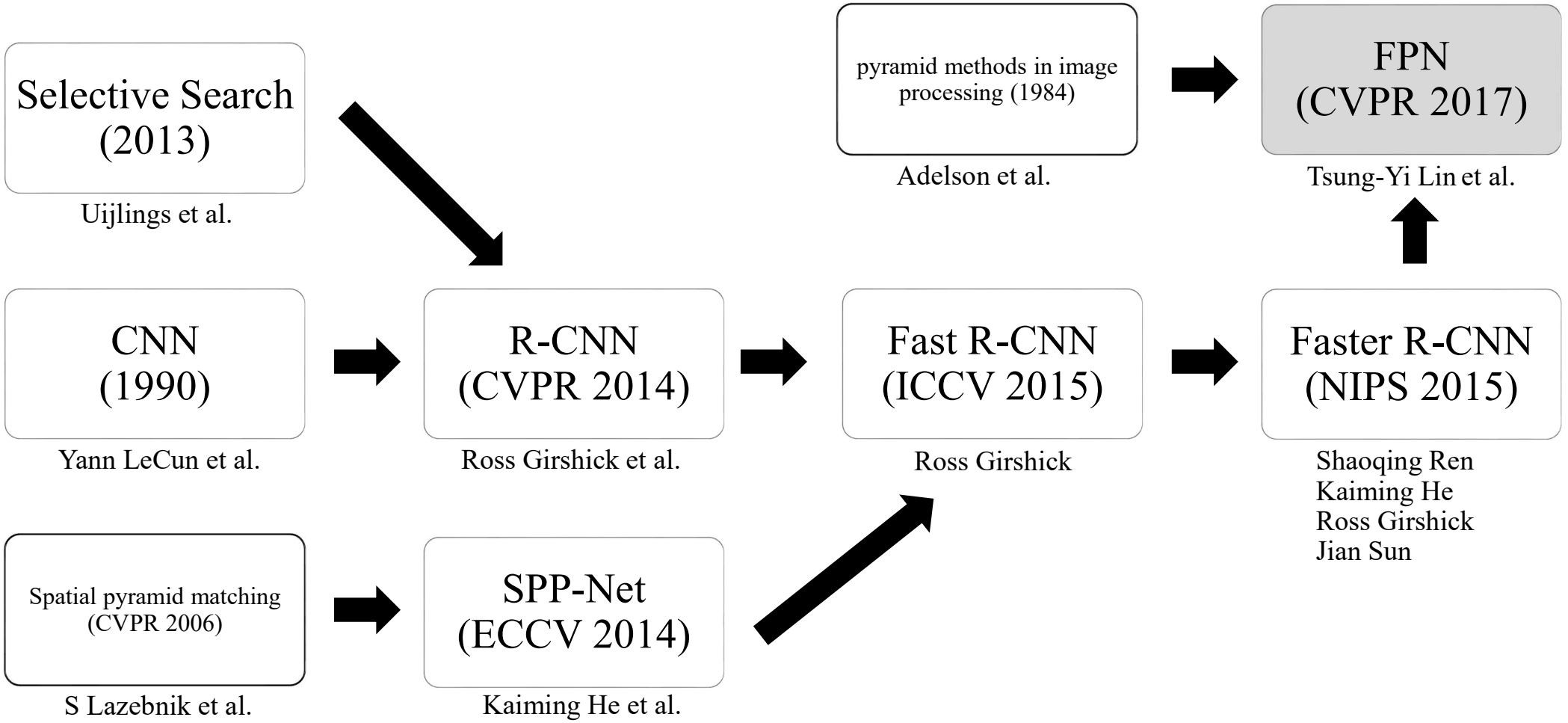
Tsung-Yi Lin[1,2], Piotr Dollár[1], Ross Girshick[1], Kaiming He[1], Bharath Hariharan[1], Serge Belongie[2]

[1]Facebook AI Research (FAIR)

[2]Cornell University and Cornell Tech

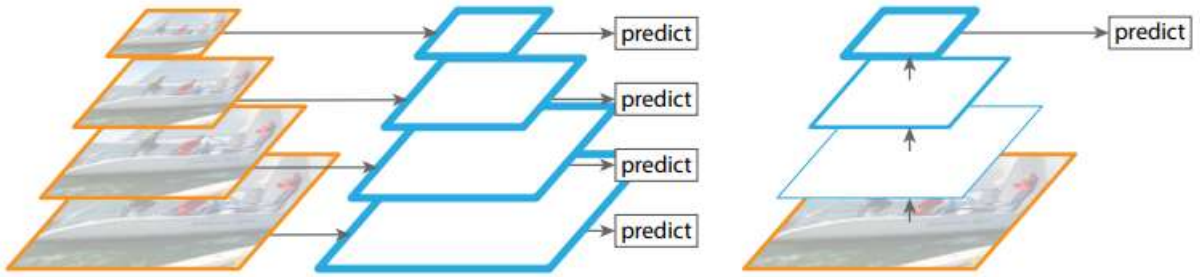
geunbae.lee1@gmail.com
OVERFITTING.AI

CNN-based object detection papers



1. Introduction

FPN 전체 구조 설명: https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c



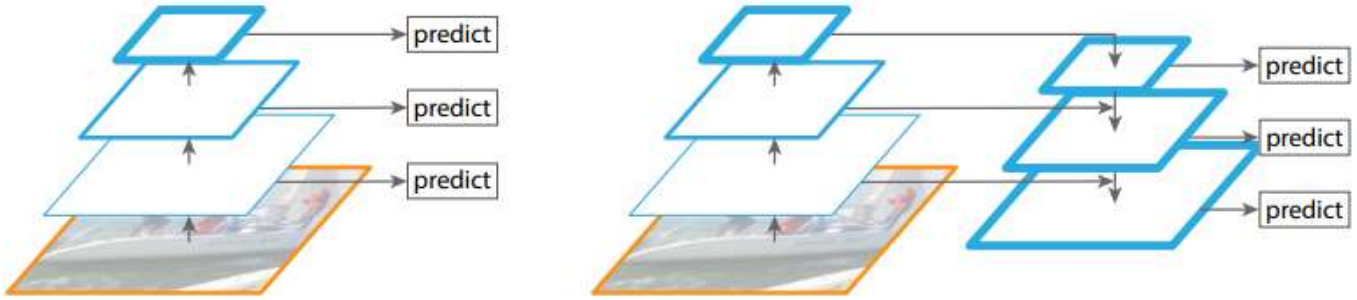
(a) Featurized image pyramid (b) Single feature map

- Recognizing objects at different scales is a fundamental challenge in computer vision
- Feature pyramids built upon image pyramids form the basis of a standard solution (1(a))
- These pyramids are scale-invariant → object's scale change is offset by shifting its level in the pyramid
- this property enables a model to detect objects across a large range of scales by scanning the model over both positions and pyramid levels.
- For recognition, featurized image pyramids have been replaced by deep ConvNets
- ConvNets are capable of representing higher-level semantics & more robust to variance in scale → recognize features computed on a single input scale (1(b))

1. Introduction

- But even with this robustness, pyramids are still needed to get the most accurate results
- top entries in the ImageNet and COCO challenges use multi-scale testing on featurized image pyramids
- The principle advantage of featurizing each level of an image pyramid
    1. produces a multi-scale feature representation in which
    2. all levels are semantically strong, including the high-res levels.
- Nevertheless, featurizing each level of an image pyramid has obvious limitations
    1. Inference time increases considerably → impractical for real applications
    2. training deep net end-to-end on an image pyramid is infeasible in terms of memory
- A deep ConvNet computes a feature hierarchy layer by layer, and with subsampling layers the feature hierarchy has an inherent multiscale, pyramidal shape.
- This in-net feature hierarchy produces feature maps of different spatial resolutions
    1. but introduces large semantic gaps caused by different depths
    2. The high-res maps have low-level features that harm their representational capa. for object recognition

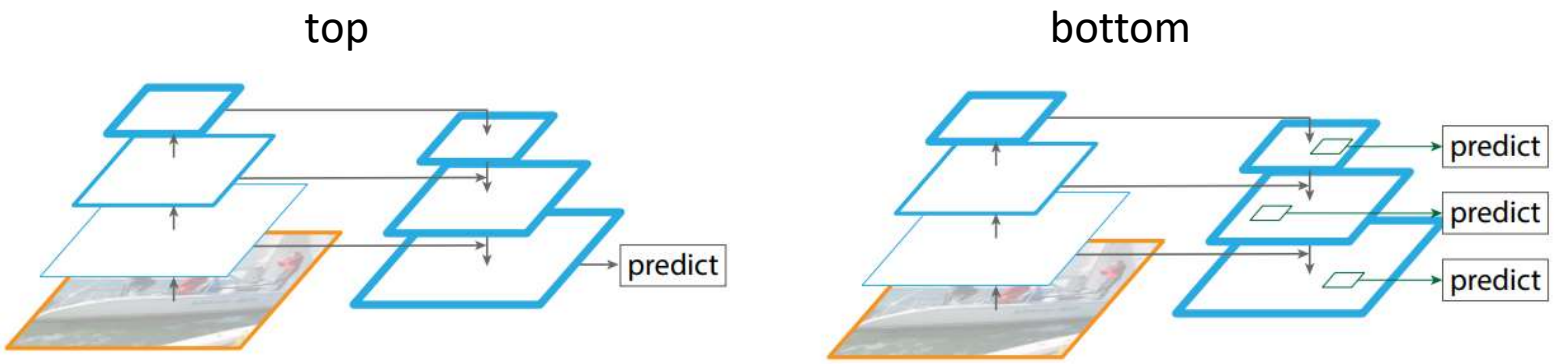
1. Introduction



(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

- SSD is one of the first attempts at using a ConvNet's pyramidal feature hierarchy (c)
- SSD-style pyramid reuse the multi-scale feature maps from different layers computed in the forward pass and thus come free of cost.
- But to avoid using low-level features, SSD reuse already computed layers and instead builds the pyramid starting from high up in the net and then by adding several new layers
  → it misses the opportunity to reuse the higher-res maps of the feature hierarchy
- FPN leverages the pyramidal shape of a ConvNet's feature hierarchy while creating a feature pyramid that has strong semantics at all scales
- FPN rely on an architecture combines: low-res, semantically strong features + high-res, semantically weak features via a top-down pathway and lateral connections (d)

1. Introduction



- FPN has rich semantics at all levels and is built quickly from a single input image scale → can replace featurized image pyramids without sacrificing representational power, speed, or memory
- Similar architectures adopting top-down and skip connections are popular in recent research
- Their goals are to produce a single high-level feature map of a fine res on which the predictions are to be made (top)
- On the contrary, FPN leverages the architecture as a feature pyramid where predictions are independently made on each level (bottom)

1. Introduction

- *Without bells and whistles*, we report a SOTA single-model result on the challenging COCO detection benchmark simply based on FPN and a basic Faster R-CNN detector
- In ablation experiments, we find that
    1. for BB proposals: increases AR +8.0 pts
    2. for object detection: improves COCO-style AP +2.3 pts, PASCAL-style AP + 3.8 pts
- FPN can be trained end-to-end with all scales and is used consistently at train/test time
- FPNs are able to achieve higher accuracy than all existing SOTA methods
- improvement is achieved without increasing the testing time over the single-scale baseline

2. Related Work

Hand-engineered features and early neural net

- SIFT features were originally extracted at scale-space extrema and used for feature point matching
- HOG features and later SIFT features as well, were computed densely over entire image pyramids
- HOG and SIFT pyramids have been used in many works for classification, detection, pose estimation
- There has also been significant interest in computing featurized image pyramids quickly
- Dollar´ et al. demonstrated fast pyramid computation by first computing a sparsely sampled (in scale) pyramid and then interpolating missing levels
- Before HOG and SIFT, early work on face detection with ConvNets computed shallow networks over image pyramids to detect faces across scales.

2. Related Work

Deep ConvNet object detectors

- With the modern deep ConvNets, object detectors showed dramatic improvements in accuracy
    1. OverFeat adopted a strategy similar to early neural net face detectors by applying a ConvNet as a sliding window detector on an image pyramid
    2. R-CNN adopted a region proposal in which each proposal was scale-normalized before classifying with a ConvNet
    3. SPPnet demonstrated that region-based detectors pplied much more efficiently on feature maps extracted on a single image scale
    4. Recent accurate detection methods (Fast R-CNN, Faster R-CNN) advocate using features computed from a single scale because it offers a good trade-off between accuracy and speed
- Multi-scale detection, however, still performs better, especially for small objects

2. Related Work

Methods using multiple layers

- recent approaches improve detection and segmentation by using different layers in a ConvNet
    1. FCN sums partial scores for each category over multiple scales to compute semantic segmentations.
    2. Hypercolumns uses a similar method for object instance segmentation.
    3. Several other approaches concatenate features of multiple layers before computing predictions, which is equivalent to summing transformed features
    4. SSD & MS-CNN predict objects at multiple layers of the feature hierarchy w/out combining features or scores
- There are recent methods exploiting lateral/skip connections that associate low-level feature maps across res and semantic levels
    1. U-Net and SharpMask for segmentation
    2. Recombinator networks for face detection
    3. Stacked Hourglass networks for keypoint estimation
    4. Laplacian pyramid presentation for FCNs to progressively refine segmentation
- Although these methods adopt architectures with pyramidal shapes, they are unlike featurized image pyramids where predictions are made independently at all levels

3. Feature Pyramid Networks

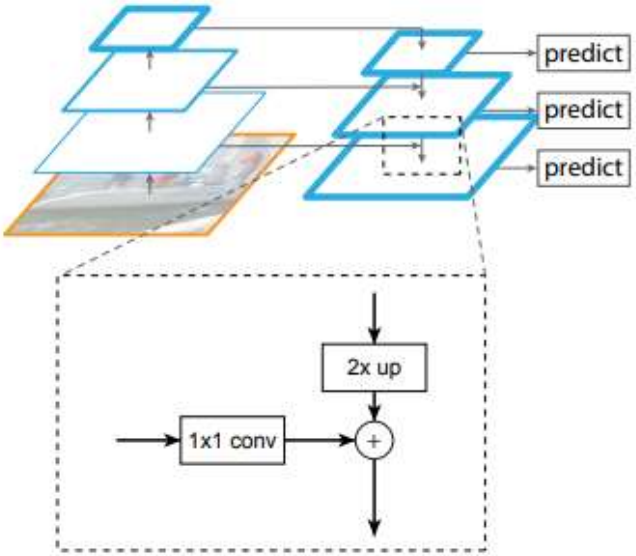
- Our goal is to leverage ConvNet's pyramidal feature hierarchy, which has semantics from low to high levels, and build a feature pyramid with high-level semantics throughout
- FPN is general purpose and we focus on sliding window proposers and region-based detectors
- **FPN takes a single-scale image of arbitrary size as input, and outputs proportionally sized feature maps at multiple levels** → independent of the backbone conv architectures (ResNet)
- FPN involves a bottom-up pathway, a top-down pathway, and lateral connections

3. Feature Pyramid Networks

Bottom-up pathway

- feedforward computation of the backbone ConvNet, which computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2.
- There are often many layers producing output maps of the same size → these layers are in the same network stage.
- define one pyramid level for each stage.
- choose the output of the last layer of each stage as reference set of feature maps, which we will enrich to create our pyramid. → the deepest layer of each stage should have the strongest features
- for ResNets we use the feature activations output by each stage's last residual block.
- denote the output of these last residual blocks as $\{C_2, C_3, C_4, C_5\}$ for conv2, conv3, conv4, conv5 outputs
- they have strides of {4, 8, 16, 32} pixels with respect to the input image
- do not include conv1 into the pyramid → too large memory footprint

3. Feature Pyramid Networks



Top-down pathway and lateral connections

- *hallucinates* higher res features by upsampling spatially coarser (semantically stronger) feature maps from higher pyramid levels
- These features are then enhanced with features from the bottom-up pathway via lateral connections
- Each lateral connection merges feature maps of the same spatial size from the bottom-up pathway and the top-down pathway
- bottom-up feature map is of lower-level semantics, but its activations are more accurately localized as it was subsampled fewer times.
- With a coarser-resolution feature map, upsample the spatial res by a factor of 2 (using nearest neighbor upsampling for simplicity)
- The upsampled map is then merged with the corresponding bottom-up map (1×1 conv layer to reduce channel) by element-wise addition

피쳐 맵에서 원래 크기의 맵에서 앨리어싱 효과를 줄이기 위해 1x1 합성곱을 추가 후 3x3 필터를 적용 → official code는 아니지만 꽤 스타가 있는 ms resercher가 만든 깃헙에 3x3 convnet은 stride=1, padding=1 로 확인되어 인풋과 동일한 아웃풋 피쳐맵이 나옵니다 https://github.com/jwyang/fpn.pytorch

3. Feature Pyramid Networks

Top-down pathway and lateral connections

- This process is iterated until the finest resolution map is generated.
- To start the iteration, attach a 1×1 con layer on C_5 to produce the coarsest resolution map.
- Finally, append a 3×3 conv on each merged map to generate the final feature map, which is to reduce the *aliasing effect* of upsampling.
- final set of feature maps is called $\{P_2, P_3, P_4, P_5\}$, corresponding to $\{C_2, C_3, C_4, C_5\}$ that are respectively of the same spatial sizes
- Because all levels of the pyramid use shared classifiers/regressors as in a traditional featurized image pyramid, we fix the feature dim (# of channels, denoted as d) in all the feature maps
- set $d = 256$ → all extra conv layers have 256-channel outputs
- There are no non-linearities in these extra layers
- Designing better connection modules is not the focus of this paper

FPN 개념 자체가 중요하므로, implementation detail 은 다루지 않음?

4. Applications

- we adopt our method in RPN for BB proposal generation and in Fast R-CNN for object detection
- To demonstrate the simplicity and effectiveness of our method, we make minimal modifications to the original systems of when adapting them to our feature pyramid

4. Applications

4.1. Feature Pyramid Networks for RPN

- In the original RPN design, a small subnetwork is evaluated on dense 3×3 sliding windows, on top of a single scale conv feature map, performing object/non-object binary classification and BB regression
- This is realized by a 3×3 conv layer followed by two sibling 1×1 conv for classification and regression, which we refer to as a network head
- The object/non-object criterion and BB regression target are defined w/ respect to a set of reference boxes (anchors) → anchors are of multiple pre-defined scales and aspect ratios in order to cover objects of different shapes
- adapt RPN by replacing the single-scale feature map with our FPN
- attach a head of the same design (3×3 conv and two siblings 1×1 convs) to each level on our feature pyramid
- Because the head slides densely over all locations in all pyramid levels, it is not necessary to have multi-scale anchors on a specific level.

4. Applications

4.1. Feature Pyramid Networks for RPN

- we assign anchors of a single scale to each level.
- define the anchors to have areas of $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ pixels on $\{P_2, P_3, P_4, P_5, P_6\}$
- use anchors of multiple aspect ratios {1:2, 1:1, 2:1} at each level → total 15 anchors over the pyramid
- assign training labels to the anchors based on their IoU ratios w/ ground-truth BB
- an anchor is assigned a POS label if it has the highest IoU for a given ground truth box or an IoU > 0.7 with any ground-truth box, and;
- an anchor is assigned a NEG label if it has IoU < 0.3 for all ground-truth boxes → no extra rules
- the parameters of the heads are shared across all feature pyramid level
- The good performance of sharing parameters indicates that all levels of our pyramid share similar semantic levels → using a featurized image pyramid, where a common head classifier can be applied to features computed at any image scale.

4. Applications

4.2. Feature Pyramid Networks for Fast R-CNN

- To use Fast R-CNN with our FPN, we need to assign RoIs of different scales to the pyramid levels
- We view our feature pyramid as if it were produced from an image pyramid.
- Thus we can adapt the assignment strategy of region-based detectors in the case when they are run on image pyramids. Formally, we assign an RoI of (w, h) on the input image to the network to the level P_k of our feature pyramid by $k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor$
- Here 224 is the canonical ImageNet pre-training size, and k_0 is the target level on which an RoI with w × h = 2242 should be mapped into.
- Analogous to the ResNet-based Faster R-CNN that uses C4 as the single-scale feature map, we set k_0 to 4. Eqn. means that if the RoI's scale becomes smaller (say, 1/2 of 224), it should be mapped into a finer-resolution level (say, k = 3).
- We attach predictor heads (in Fast R-CNN the heads are class-specific classifiers and BB regressors) to all RoIs of all levels.
- Again, the heads all share parameters, regardless of their levels. a ResNet's conv5 layers (a 9-layer deep subnetwork) are adopted as the head on top of the conv4 features, but FPN has already harnessed conv5 to construct the feature pyramid.
- adopt RoI pooling to extract 7×7 features and attach two hidden 1,024-d FC layers (each followed by ReLU) before the final classification and bounding box regression layers.

5. Experiments on Object Detection

- experiments on the 80 category COCO detection dataset
- train using the union of 80k train images and a 35k subset of val images and report ablations on a 5k subset of val images
- We also report final results on the standard test set which has no disclosed labels.
- all network backbones are pre-trained on the ImageNet1k classification set and then fine-tuned on the detection dataset
- We use the pre-trained ResNet-50 and ResNet-101 models

5. Experiments on Object Detection

RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR^{1k}_s	AR^{1k}_m	AR^{1k}_l
(a) baseline on conv4	C_4	47k			36.1	48.3	32.0	58.7	62.2
(b) baseline on conv5	C_5	12k			36.3	44.9	25.3	55.5	64.2
(c) **FPN**	$\{P_k\}$	200k	✓	✓	**44.0**	**56.3**	**44.9**	**63.4**	66.2
Ablation experiments follow:									
(d) bottom-up pyramid	$\{P_k\}$	200k	✓		37.4	49.5	30.5	59.9	**68.0**
(e) top-down pyramid, w/o lateral	$\{P_k\}$	200k		✓	34.5	46.1	26.5	57.4	64.7
(f) only finest level	P_2	750k	✓	✓	38.4	51.3	35.1	59.7	67.6

5.1. Region Proposal with RPN

- evaluate the COCO-style AR and AR on small, medium, and large objects (AR_s, AR_m, and AR_l)
- report results for 100 and 1000 proposals per images (AR^{100} and AR^{1k})

Implementation details.

- All architectures in Table 1 are trained end-to-end.
- The input image is resized such that its shorter side has 800 pixels.
- We adopt synchronized SGD training on 8 GPUs.
- A mini-batch involves 2 images per GPU and 256 anchors per image.
- weight decay = 0.0001, momentum = 0.9, LR = 0.02 for first 30k mini-batches & 0.002 for next 10k.
- For all RPN experiments, include the anchor boxes that are outside the image for training
- Training RPN with FPN on 8 GPUs takes about 8 hours on COCO

5. Experiments on Object Detection

RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR^{1k}_s	AR^{1k}_m	AR^{1k}_l
(a) baseline on conv4	C_4	47k			36.1	48.3	32.0	58.7	62.2
(b) baseline on conv5	C_5	12k			36.3	44.9	25.3	55.5	64.2
(c) **FPN**	$\{P_k\}$	200k	✓	✓	**44.0**	**56.3**	**44.9**	**63.4**	66.2

5.1.1 Ablation Experiments

Comparisons with baselines.

- For fair comparisons with original RPNs, run two baselines (Table 1(a, b)) using the single-scale map of C4 or C5, both using the same hyper-parameters as FPN, including using 5 scale anchors of $\{32^2, 64^2, 128^2, 256^2, 512^2\}$.
- (b) shows no advantage over (a) → a single higher level feature map is not enough because there is a trade-off between coarser resolutions and stronger semantics
- Placing FPN in RPN improves AR^{1k} to 56.3 (c), + 8.0 pts over the single-scale RPN baseline (48.3→56.3)
- the performance on small objects (AR^{1k}_s) is boosted by 12.9 pts (32.0 → 44.9)

  → pyramid representation greatly improves RPN's robustness to object scale variation

5. Experiments on Object Detection

RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR^{1k}_s	AR^{1k}_m	AR^{1k}_l
(a) baseline on conv4	C_4	47k			36.1	48.3	32.0	58.7	62.2
(b) baseline on conv5	C_5	12k			36.3	44.9	25.3	55.5	64.2
(c) **FPN**	$\{P_k\}$	200k	✓	✓	**44.0**	**56.3**	**44.9**	**63.4**	66.2
Ablation experiments follow:									
(d) bottom-up pyramid	$\{P_k\}$	200k	✓		37.4	49.5	30.5	59.9	**68.0**
(e) top-down pyramid, w/o lateral	$\{P_k\}$	200k		✓	34.5	46.1	26.5	57.4	64.7
(f) only finest level	P_2	750k	✓	✓	38.4	51.3	35.1	59.7	67.6

5.1.1 Ablation Experiments

How important is top-down enrichment?

왜 성능이 잘 나오지 않는지 따로 설명이 없음..

- (d) shows the results of FPN without the top-down pathway
- With this modification, the 1×1 lateral connections followed by 3×3 convolutions are attached to the bottom-up pyramid.
- This architecture simulates the effect of reusing the pyramidal feature hierarchy
- The results in (d) are just on par with the RPN baseline and lag far behind FPN
  → because there are large semantic gaps between different levels on the bottom-up pyramid
- a variant of (d) w/out sharing the parameters of the heads similarly degraded performance, This issue cannot be simply remedied by level-specific heads.

5. Experiments on Object Detection

RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR^{1k}_s	AR^{1k}_m	AR^{1k}_l
(a) baseline on conv4	C_4	47k			36.1	48.3	32.0	58.7	62.2
(b) baseline on conv5	C_5	12k			36.3	44.9	25.3	55.5	64.2
(c) **FPN**	$\{P_k\}$	200k	✓	✓	**44.0**	**56.3**	**44.9**	**63.4**	66.2
Ablation experiments follow:									
(d) bottom-up pyramid	$\{P_k\}$	200k	✓		37.4	49.5	30.5	59.9	**68.0**
(e) top-down pyramid, w/o lateral	$\{P_k\}$	200k		✓	34.5	46.1	26.5	57.4	64.7
(f) only finest level	P_2	750k	✓	✓	38.4	51.3	35.1	59.7	67.6

5.1.1 Ablation Experiments

How important are lateral connections?

- (e) shows the ablation results of a top-down feature pyramid without the 1×1 lateral connections. This top-down pyramid has strong semantic features and fine resolutions
- But the locations of these features are not precise, because these maps have been downsampled and upsampled several times
- More precise locations of features can be directly passed from the finer levels of the bottom-up maps via the lateral connections to the top-down maps. As a results, FPN has an AR^{1k} score 10 pts higher than (e)

5. Experiments on Object Detection

RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR^{1k}_s	AR^{1k}_m	AR^{1k}_l
(a) baseline on conv4	C_4	47k			36.1	48.3	32.0	58.7	62.2
(b) baseline on conv5	C_5	12k			36.3	44.9	25.3	55.5	64.2
(c) **FPN**	$\{P_k\}$	200k	✓	✓	**44.0**	**56.3**	**44.9**	**63.4**	66.2
Ablation experiments follow:									
(d) bottom-up pyramid	$\{P_k\}$	200k	✓		37.4	49.5	30.5	59.9	**68.0**
(e) top-down pyramid, w/o lateral	$\{P_k\}$	200k		✓	34.5	46.1	26.5	57.4	64.7
(f) only finest level	P_2	750k	✓	✓	38.4	51.3	35.1	59.7	67.6

5.1.1 Ablation Experiments

How important are pyramid representations?

- Instead of resorting to pyramid representations, attach the head to the highest-res, strongly semantic feature maps of P_2 (i.e., the finest level in our pyramids).
- Similar to the single-scale baselines, we assign all anchors to the P_2 feature map.
- This variant (f) is better than the baseline but inferior to FPN

  → RPN is a sliding window detector with a fixed window size, so scanning over pyramid levels can increase its robustness to scale variance
- In addition, we note that using P_2 alone leads to more anchors (750k, (f)) caused by its large spatial resolution → suggests that a larger # of anchors is not sufficient in itself to improve accuracy

5. Experiments on Object Detection

5.2. Object Detection with Fast/Faster R-CNN

- investigate FPN for region-based (non-sliding window) detectors.
- evaluate object detection by the COCO-style AP and PASCAL-style AP (at a single IoU threshold = 0.5)
- also report COCO AP on objects of the small, medium, and large sizes (namely, AP_s, AP_m, and AP_l)

Implementation details.

- The input image is resized such that its shorter side has 800 pixels.
- Synchronized SGD is used to train the model on 8 GPUs.
- Each mini-batch involves 2 image per GPU and 512 RoIs per image.
- weight decay of 0.0001 and a momentum of 0.9.
- The learning rate is 0.02 for the first 60k mini-batches and 0.002 for the next 20k.
- use 2000 RoIs per image for training and 1000 for testing.
- Training Fast R-CNN with FPN takes about 10 hours on the COCO dataset

5. Experiments on Object Detection

Fast R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(a) baseline on conv4	RPN, $\{P_k\}$	C_4	conv5			54.7	31.9	15.7	36.5	45.5
(b) baseline on conv5	RPN, $\{P_k\}$	C_5	*2fc*			52.9	28.8	11.9	32.4	43.4
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	**45.8**
Ablation experiments follow:										
(d) bottom-up pyramid	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓		44.9	24.9	10.9	24.4	38.5
(e) top-down pyramid, w/o lateral	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*		✓	54.0	31.3	13.3	35.2	45.3
(f) only finest level	RPN, $\{P_k\}$	P_2	*2fc*	✓	✓	56.3	33.4	17.3	37.3	45.6

5.2.1 Fast R-CNN (on fixed proposals)

- freeze the proposals as computed by RPN on FPN, because it has good performance on small objects that are to be recognized by the detector.
- For simplicity, do not share features between Fast R-CNN and RPN, except when specified
- As a ResNet-based Fast R-CNN baseline, adopt RoI pooling w/ an output size of 14×14 and attach all conv5 layers as the hidden layers of the head → gives an AP of 31.9 (a)
- (b) is a baseline exploiting an MLP head w/ 2 hidden FC layers, similar to the head in FPN architecture.
- It gets an AP of 28.8, → the 2-fc head does not give us any orthogonal advantage over the baseline in (a)

5. Experiments on Object Detection

Fast R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(a) baseline on conv4	RPN, $\{P_k\}$	C_4	conv5			54.7	31.9	15.7	36.5	45.5
(b) baseline on conv5	RPN, $\{P_k\}$	C_5	2*fc*			52.9	28.8	11.9	32.4	43.4
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	2*fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	**45.8**
Ablation experiments follow:										
(d) bottom-up pyramid	RPN, $\{P_k\}$	$\{P_k\}$	2*fc*	✓		44.9	24.9	10.9	24.4	38.5
(e) top-down pyramid, w/o lateral	RPN, $\{P_k\}$	$\{P_k\}$	2*fc*		✓	54.0	31.3	13.3	35.2	45.3
(f) only finest level	RPN, $\{P_k\}$	P_2	2*fc*	✓	✓	56.3	33.4	17.3	37.3	45.6

5.2.1 Fast R-CNN (on fixed proposals)

- Comparing with the baseline in (a), FPN improves AP +2.0 pts & small AP_s +2.1 pts
- Comparing with the baseline that also adopts a 2fc head (b), FPN improves AP +5.1 pts
  → indicate that FPN is superior to single-scale features for a region-based object detector
- (d) and (e) show that removing top-down or lateral connections leads to inferior results
- removing top-down connections (d) significantly degrades the accuracy, suggesting that Fast R-CNN suffers from using the low-level features at the high-resolution maps.
- In (f), adopt Fast R-CNN on the single finest scale feature map of P_2. Its result (33.4 AP) is marginally worse than that of using all pyramid levels, 33.9 AP (c)
  → because RoI pooling is a warping-like op, which is less sensitive to the region's scales.
- Despite the good accuracy of this variant, it is based on the RPN proposals of $\{P_k\}$ and has thus already benefited from the pyramid representation.

5. Experiments on Object Detection

Faster R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(*) baseline from He *et al.* [16]†	RPN, C_4	C_4	conv5			47.3	26.3	-	-	-
(a) baseline on conv4	RPN, C_4	C_4	conv5			53.1	31.6	13.2	35.6	**47.1**
(b) baseline on conv5	RPN, C_5	C_5	*2fc*			51.7	28.0	9.6	31.9	43.1
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	45.8

5.2.2 Faster R-CNN (on consistent proposals)

- in a Faster R-CNN, the RPN and Fast R-CNN must use the same backbone in order to make feature sharing possible
- (a) shows reproduction of the baseline Faster R-CNN. Under controlled settings, FPN (c) is better than this strong baseline +2.3 pts AP and +3.8 pts AP@0.5.
- (a) and (b) are baselines that are much stronger than the baseline (*) because:
    1. use an image scale of 800 pixels instead of 600;
    2. train with 512 ROI / image which accelerates convergence, in contrast to 64 ROI
    3. use 5 scale anchors instead of 4 (adding 32^2);
    4. At test time, use 1000 proposals / image instead of 300

5. Experiments on Object Detection

	ResNet-50		ResNet-101	
share features?	$AP_{@0.5}$	AP	$AP_{@0.5}$	AP
no	56.9	33.9	58.0	35.0
yes	**57.2**	**34.3**	**58.2**	**35.2**

5.2.2 Faster R-CNN (on consistent proposals)

Sharing features.

- for simplicity, do not share the features between RPN and Fast R-CNN
- evaluate sharing features following the 4-step training → improves accuracy by a small margin
- Feature sharing also reduces testing time

Running time

- With feature sharing, FPN-based Faster R-CNN has an inference time of 0.165 s/ image on a single NVIDIA M40 GPU for ResNet-50 and 0.19 s / image for ResNet-101
- As a comparison, the single-scale ResNet-50 baseline runs at 0.32 seconds.
- FPN introduces a small extra cost by the extra layers but has a lighter weight head.
- Overall FPN is faster than the ResNet-based Faster R-CNN counterpart.

5. Experiments on Object Detection

5.2.3 Comparing with COCO Competition Winners

- ResNet-101 model is not sufficiently trained with the default learning rate schedule
- So increase the # of mini-batches by 2× at each learning rate when training the Fast R-CNN step
- This increases AP on minival to 35.6, without sharing features.
- We have not evaluated its feature-sharing version due to limited time

5. Experiments on Object Detection

method	backbone	competition	image pyramid	test-dev					test-std				
				$AP_{@.5}$	AP	AP_s	AP_m	AP_l	$AP_{@.5}$	AP	AP_s	AP_m	AP_l
ours, Faster R-CNN on **FPN**	ResNet-101	-		**59.1**	**36.2**	**18.2**	**39.0**	48.2	**58.5**	**35.8**	**17.5**	**38.7**	47.8
*Competition-winning **single-model** results follow:*													
G-RMI†	Inception-ResNet	2016		-	34.7	-	-	-	-	-	-	-	-
AttractioNet‡ [10]	VGG16 + Wide ResNet§	2016	✓	53.4	35.7	15.6	38.0	**52.7**	52.9	35.3	14.7	37.6	**51.9**
Faster R-CNN +++ [16]	ResNet-101	2015	✓	55.7	34.9	15.6	38.7	50.9	-	-	-	-	-
Multipath [40] (on minival)	VGG-16	2015		49.6	31.5	-	-	-	-	-	-	-	-
ION‡ [2]	VGG-16	2015		53.4	31.2	12.8	32.9	45.2	52.9	30.7	11.8	32.8	44.8

5.2.3 Comparing with COCO Competition Winners

- W/out adding bells and whistles, our single-model entry has surpassed these strong, heavily engineered competitors.
- On the test-dev set, FPN increases over the existing best results + 0.5 pts of AP (36.2 vs. 35.7) and +3.4 pts of $AP_{@0.5}$ (59.1 vs. 55.7).
- FPN does not rely on image pyramids and only uses a single input image scale, but still has outstanding AP on small-scale objects → achieved by high res image inputs with previous methods.
- FPN does not exploit many popular improvements, such as iterative regression, hard negative mining, context modeling, stronger data augmentation