第7回全日本コンピュータビジョン勉強会
「CVPR2021読み会」（前編）

You Only Look One-level Feature の解説と見せかけた物体検出のよもやま話

株式会社Mobility Technologies
内田 祐介

Mobility Technologies

自己紹介

■Yusuke Uchida

- -2017年　：通信キャリアの研究所で画像認識・検索の研究に従事
- -2016年　：社会人学生として博士号を取得（情報理工学）
- 2017年-　：DeNA中途入社、深層学習を中心とした
  コンピュータビジョン技術の研究開発に従事
- 2019年-　：　Mobility Technologiesへ移籍

Twitter: https://twitter.com/yu4u
GitHub: https://github.com/yu4u
Qiita: https://qiita.com/yu4u
SlideShare: https://www.slideshare.net/ren4yu/
Kaggle: https://www.kaggle.com/ren4yu

Mobility Technologies

You Only Look One-level Feature (YOLOF)

■みんな大好きYOLO！

YOLO*?

- YOLO: Single shot object detectioの火付け役
  - J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in Proc. Of CVPR, 2016.
- YOLOv2: FCN化、k-meansにより作成されたアンカーベースの検出
  - J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” in Proc. Of CVPR, 2017.
- YOLOv3: より強力なバックボーン、FPN的構造、複数解像度の特徴からの検出
  - J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” in arXiv, 2018.
- YOLOv4: ベストプラクティス全部入りみたいなやつ
  - A. Bochkovskiy, C. Wang, and H. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” in arXiv, 2020.
  - https://github.com/AlexeyAB/darknet
- YOLOv5: Ultralytics社のOSS実装。最早手法とかではなくて学習・推論を含めたフレームワークと言ったほうが良い。何故かKagglerが大好き
  - https://github.com/ultralytics/yolov5

YOLO*?

■YOLO: Single shot object detectioの火付け役

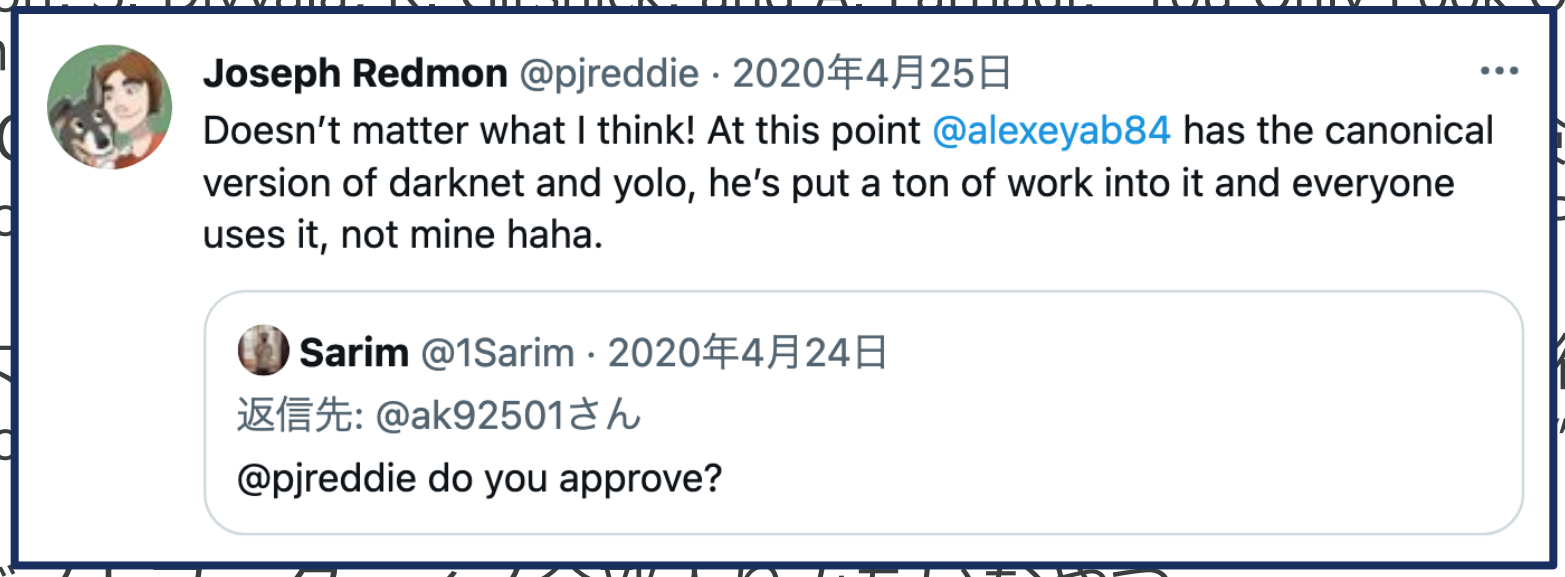
- J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Tim

■YOLOv2: FC 出

- J. Redmo Proc. Of CVPR, 2017.

■YOLOv3: よ 徴からの検出

- J. Redmo ” in arXiv, 2018.



■YOLOv4: ベストプラクティス全部入りみたいなやつ

- **A. Bochkovskiy**, C. Wang, and H. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” in arXiv, 2020.
- https://github.com/AlexeyAB/darknet

■YOLOv5: Ultralytics社のOSS実装。最早手法とかではなくて学習・推論を含めたフレームワークと言ったほうが良い。何故かKagglerが大好き

- https://github.com/ultralytics/yolov5

YOLO*?

■YOLO: Single shot object detectioの火付け役

- J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in Proc. Of CVPR, 2016.

■YOLOv2: FCN化、k-meansにより作成されたアンカーベースの検出

- J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” in Proc. Of CVPR, 2017.

■YOLOv3: より強力なバックボーン、FPN的構造、複数解像度の特徴からの検出

- J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” in arXiv, 2018.

■YOLOv4: ベストプラクティス全部入りみたいなやつ

- **A. Bochkovskiy**, C. Wang, and H. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” in arXiv, 2020.

Alexey Bochkovskiy @alexeyab84 · 2020年5月23日

I am an AI developed by @pjreddie to complete his AI without his participation

学習・推論を含め
き

https://github.com/ultralytics/yolov5

YOLO*?

■YOLO: Single shot object detectioの火付

- J. Redmon, S. Divvala, R. Girshick, and A. F
- al-Time Object Detection," in Proc. Of CV

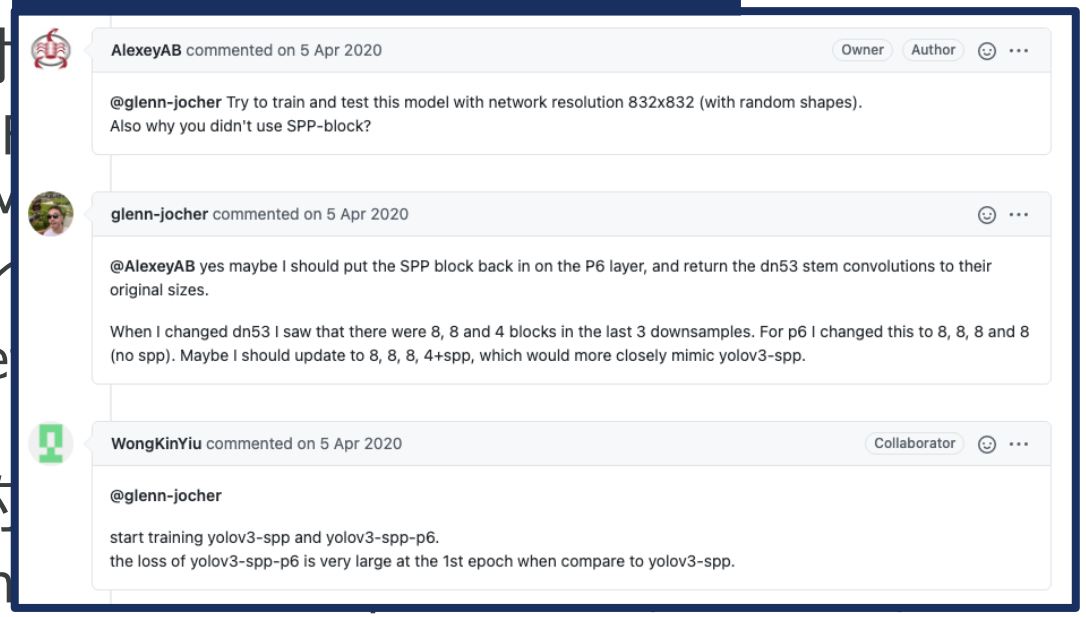
v4論文

7. Acknowledgements ↓Ultralytics CEO

The authors wish to thank Glenn Jocher for the ideas of Mosaic data augmentation, the selection of hyper-parameters by using genetic algorithms and solving the grid sensitivity problem https://github.com/ultralytics/yolov3.

2018.

AlexeyAB/darknet の issue



■YOLOv4: ベストプラクティス全部入りみたいなやつ

- A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," in arXiv, 2020.
- https://github.com/AlexeyAB/darknet

■YOLOv5: Ultralytics社のOSS実装。最早手法とかではなくて学習・推論を含めたフレームワークと言ったほうが良い。何故かKagglerが大好き

https://github.com/ultralytics/yolov5

YOLOv5のablation study by @hirotomusiker

- https://www.kaggle.com/c/global-wheat-detection/discussion/172436

condition	private (best conf threshold)	public (best conf threshold)	EMA	lr decay	GIOU object-ness	plus 2 target grids	target assign-ment using w, h ratios	filter tiny boxes (w>2, h>2)	mosaic	hsv	scaling
w/o pseudo labeling	0.642 (0.5)	0.737 (0.5)	-		-	-	-	-	-	-	-
pseudo labeling, default	0.672 (0.7)	0.760 (0.6)	✔	✔	✔	✔	✔	✔	✔	✔	✔
w/o EMA	0.671 (0.7)	0.761 (0.6)		✔	✔	✔	✔	✔	✔	✔	✔
w/o lr decay	0.673 (0.7)	0.756 (0.6)	✔		✔	✔	✔	✔	✔	✔	✔
w/o GIOU objectness	**0.663 (0.9)**	0.758 (0.8)	✔	✔		✔	✔	✔	✔	✔	✔
w/o 2 neighboring target grids	**0.665 (0.5)**	**0.750 (0.4)**	✔	✔	✔		✔	✔	✔	✔	✔
w/o GIOU objectness w/o 2 neighbor target grids	**0.656 (0.6)**	**0.751 (0.4)**	✔	✔			✔	✔	✔	✔	✔
anchor assignment using IoU	0.672 (0.7)	0.759 (0.6)	✔	✔	✔	✔		✔	✔	✔	✔
w/o filtering tiny boxes	0.674 (0.7)	0.761 (0.6)	✔	✔	✔	✔	✔		✔	✔	✔
w/o mosaic	0.672 (0.7)	0.757 (0.6)	✔	✔	✔	✔	✔	✔		✔	✔
w/o hsv aug	0.668 (0.7)	0.760 (0.6)	✔	✔	✔	✔	✔	✔	✔		✔
w/o scaling	0.673 (0.7)	0.762 (0.6)	✔	✔	✔	✔	✔	✔	✔	✔	

YOLO*?

■PP-YOLO: PaddlePaddle版YOLO

- X. Long, et al., "PP-YOLO: An Effective and Efficient Implementation of Object Detector," in arXiv, 2020.
- X. Huang, et al., "PP-YOLOv2: A Practical Object Detector," in arXiv, 2021.

■Scaled-YOLOv4

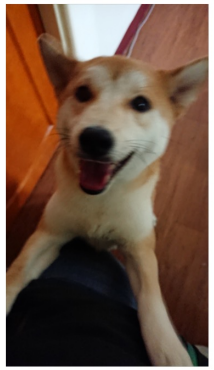
- C. Wang, A. Bochkovskiy, and H. Liao, "Scaled-YOLOv4: Scaling Cross Stage Partial Network," in Proc. of CVPR, 2021.
- https://github.com/WongKinYiu/ScaledYOLOv4

■YOLOR

- C. Wang, I. Yeh, and H. Liao, "You Only Learn One Representation: Unified Network for Multiple Tasks," in arXiv, 2021.
- https://github.com/WongKinYiu/yolor

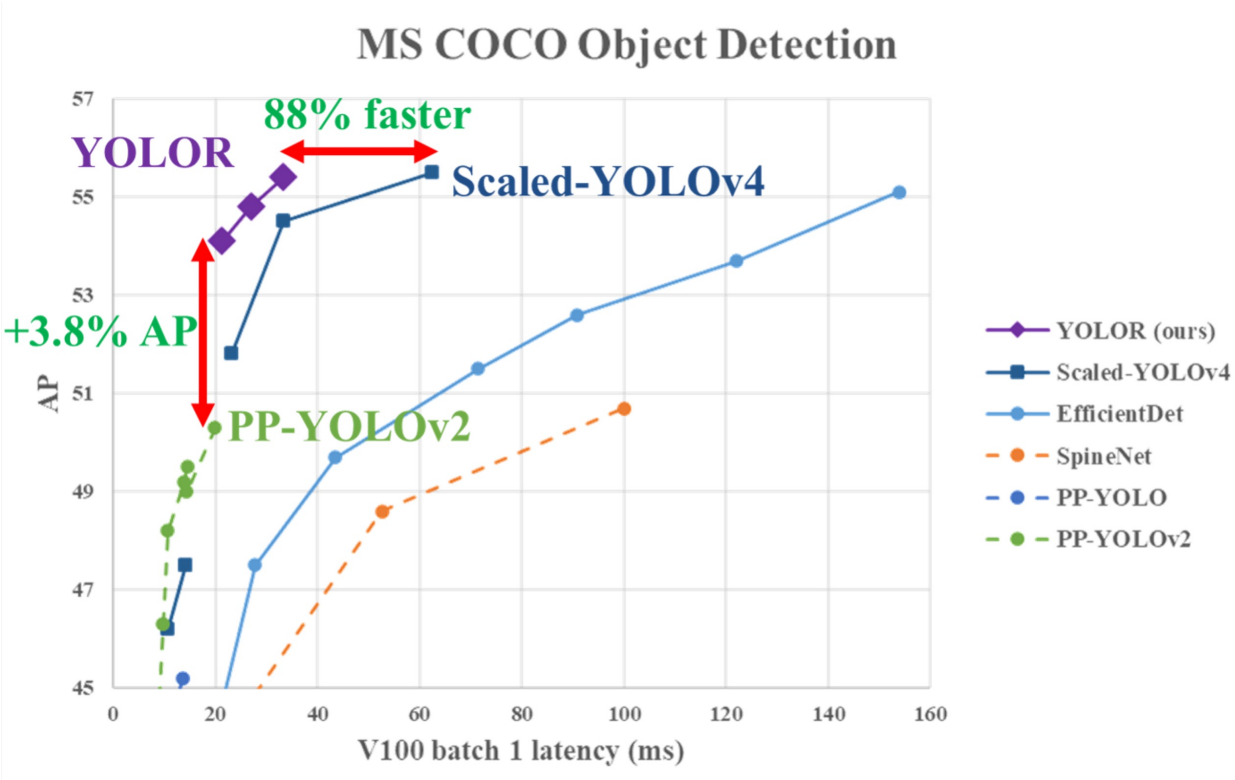
YOLOR

■論文のIntroはエモいのにめっちゃdetection結果推し



→What is this? →A Shiba Inu.
→Where is the Shiba Inu? →In a room.
→Where is she? →In a room.
→What is she doing? →LOL.
→What is her name? →A I.
→Do you love her? →Yes! Sure! Of course!

Figure 1: Human beings can answer different questions from the same input. Our aim is to train a single deep neural network that can serve many tasks.



そういえばYOLOFでした

You Only Look One-level Feature

Qiang Chen[1,2*], Yingming Wang[4], Tong Yang[4], Xiangyu Zhang[4], Jian Cheng[1,2,3†], Jian Sun[4]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]CAS Center for Excellence in Brain Science and Intelligence Technology
[4]MEGVII Technology
{qiang.chen, jcheng}@nlpr.ia.ac.cn, {wangyingming, yangtong, zhangxiangyu, sunjian}@megvii.com

Q. Chen, et al., "You Only Look One-level Feature," in Proc. of CVPR, 2021.

論文の主張

- Feature Pyramids Networks (FPN) はマルチスケールの特徴を融合することにより性能が向上していると思われているがポイントはそこやないで
- 物体検出における最適化問題を（multi-scaleのアンカーを使うことで）分割統治的に解いているところが一番ポイントやで
- でもマルチスケールの特徴を使った検出は複雑かつ低速なので、single-scaleでmulti-scaleに匹敵する検出器をつくるお！

ちなみに

■YOLOFは

ちなみに

■YOLOFはYOLOではありません！

ちなみに

■YOLOFはYOLOではありません！
- これまでの前フリは…

■こいつはRetinaNetです
- ちなみに何を持ってYOLOだ、RetinaNetだというのは個人的に好きな議論
  - 意味はないけど
- BackboneがDarknetならYOLO?
- 後述のアンカーがkmeansで作られていたらYOLO?
- Headにクラス毎の確率ではなくてbboxの信頼度もあったらYOLOで bboxとclass分類が別々のbranchになってたらRetinaNet?
- Loss? 後述のAnchor matchingの手法？

物体検出モデルの汎用的な表現

■Backbone, Neck, Headの組み合わせで物体検出モデルは表現できる

Tutorial 4: Customize Models

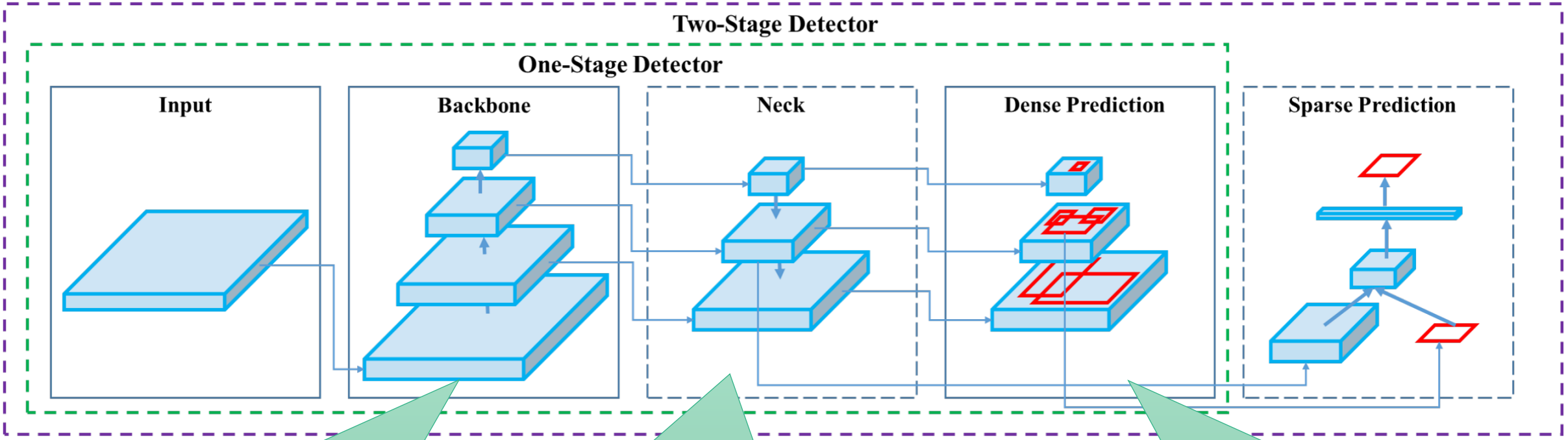
We basically categorize model components into 5 types.

- backbone: usually an FCN network to extract feature maps, e.g., ResNet, MobileNet.
- neck: the component between backbones and heads, e.g., FPN, PAFPN.
- head: the component for specific tasks, e.g., bbox prediction and mask prediction.
- roi extractor: the part for extracting RoI features from feature maps, e.g., RoI Align.
- loss: the component in head for calculating losses, e.g., FocalLoss, L1Loss, and GHMLoss.

https://mmdetection.readthedocs.io/en/latest/tutorials/customize_models.html

物体検出モデルの汎用的な表現

■Backbone, Neck, Headの組み合わせで物体検出モデルは表現できる



Backbone:
ベースとなるクラス分類モデル。Multi-scaleの特徴マップを出力
(e.g. ResNet, Darknet)

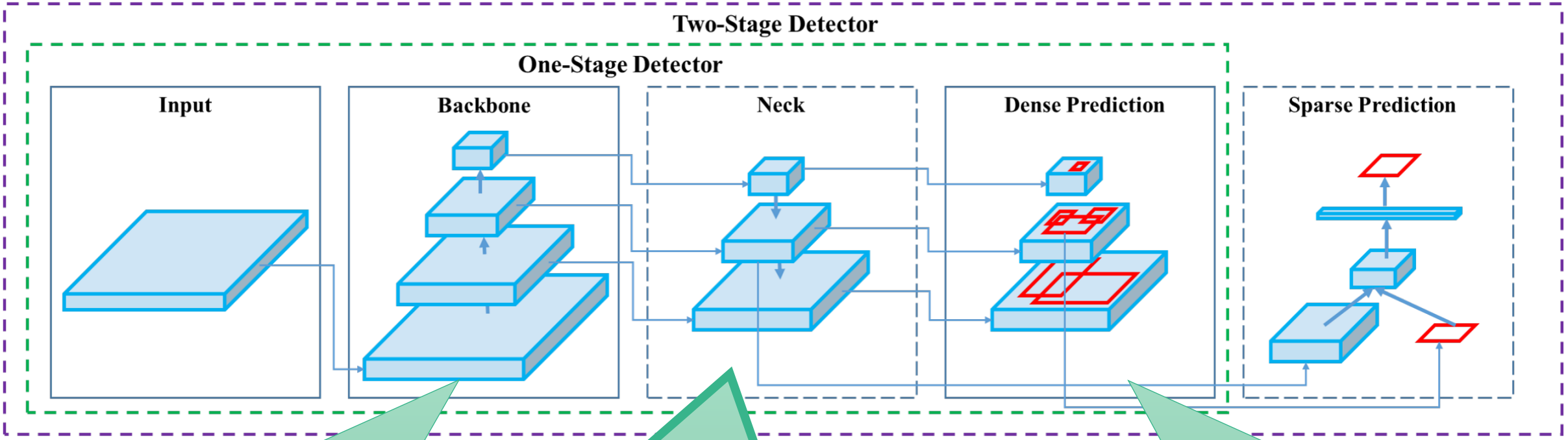
Neck:
Multi-scaleの特徴マップを入力してコネコネして出力
（e.g. FPN, BiFPN）

Head:
Multi-scaleの特徴マップを入力して検出結果を出力
(e.g. YOLO/Retina head)

A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," in arXiv, 2020.

物体検出モデルの汎用的な表現

■Backbone, Neck, Headの組み合わせで物体検出モデルは表現できる



Backbone:
ベースとなるクラス分類モデル。Multi-scaleの特徴マップを出力
(e.g. ResNet, Darknet)

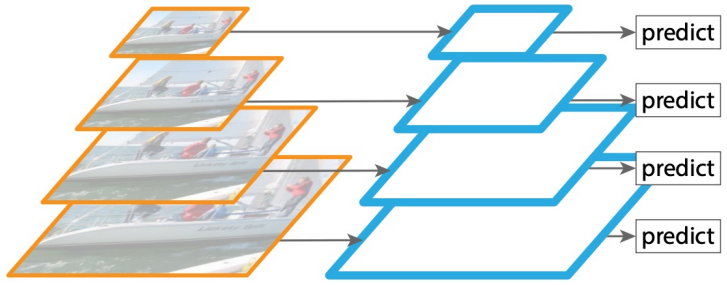
Neck:
Multi-scaleの特徴マップを入力してコネコネして出力
（e.g. FPN, BiFPN）

Head:
Multi-scaleの特徴マップを入力して検出結果を出力
(e.g. YOLO/Retina head)

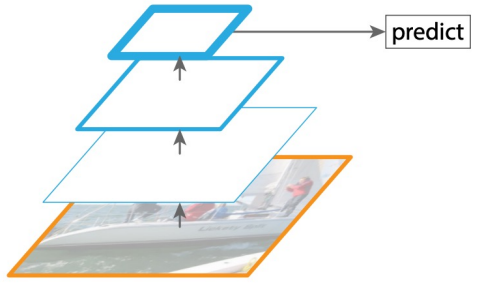
A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," in arXiv, 2020.

Feature Pyramid Network (FPN)

■出力層付近の特徴を入力層付近の特徴へと徐々に統合することで特徴の強さと特徴マップの解像度を両立



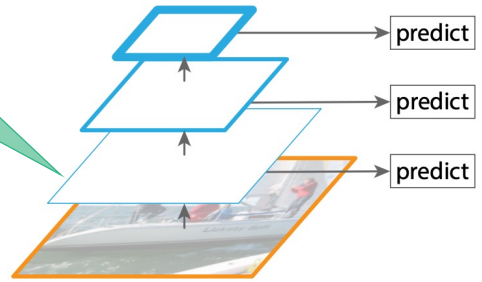
(a) Featurized image pyramid



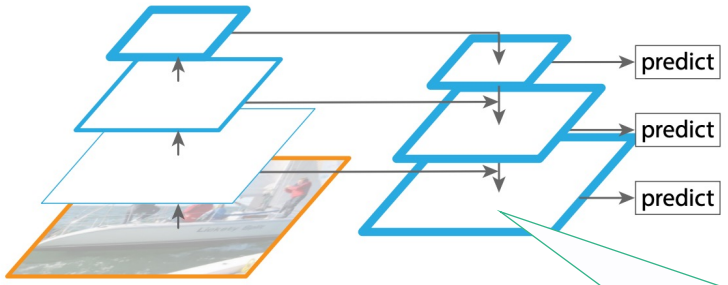
(b) Single feature map

特徴の強さ：**強**
解像度：**低**
e.g. Faster R-CNN, YOLO

特徴の強さ：**弱**
解像度：高
e.g. SSD



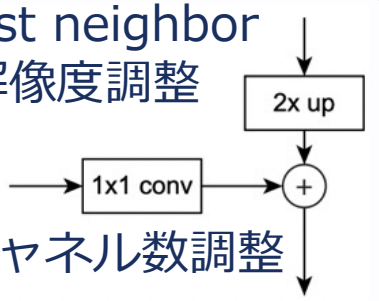
(c) Pyramidal feature hierarchy



(d) Feature Pyramid Network

特徴の強さ：**強**
解像度：**高**
FPN

Nearest neighbor で解像度調整

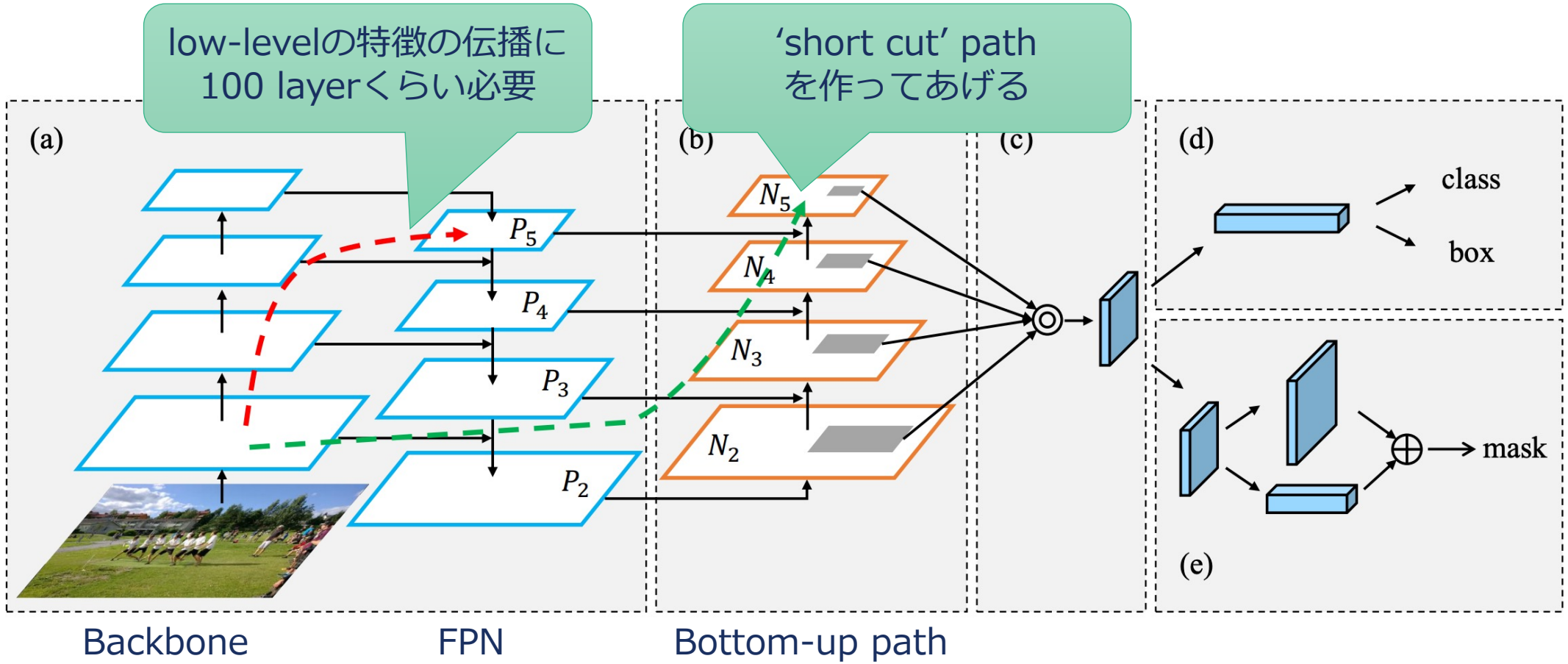


1x1でチャネル数調整

T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in Proc. of CVPR, 2017.

Path Aggregation Network (PANet)

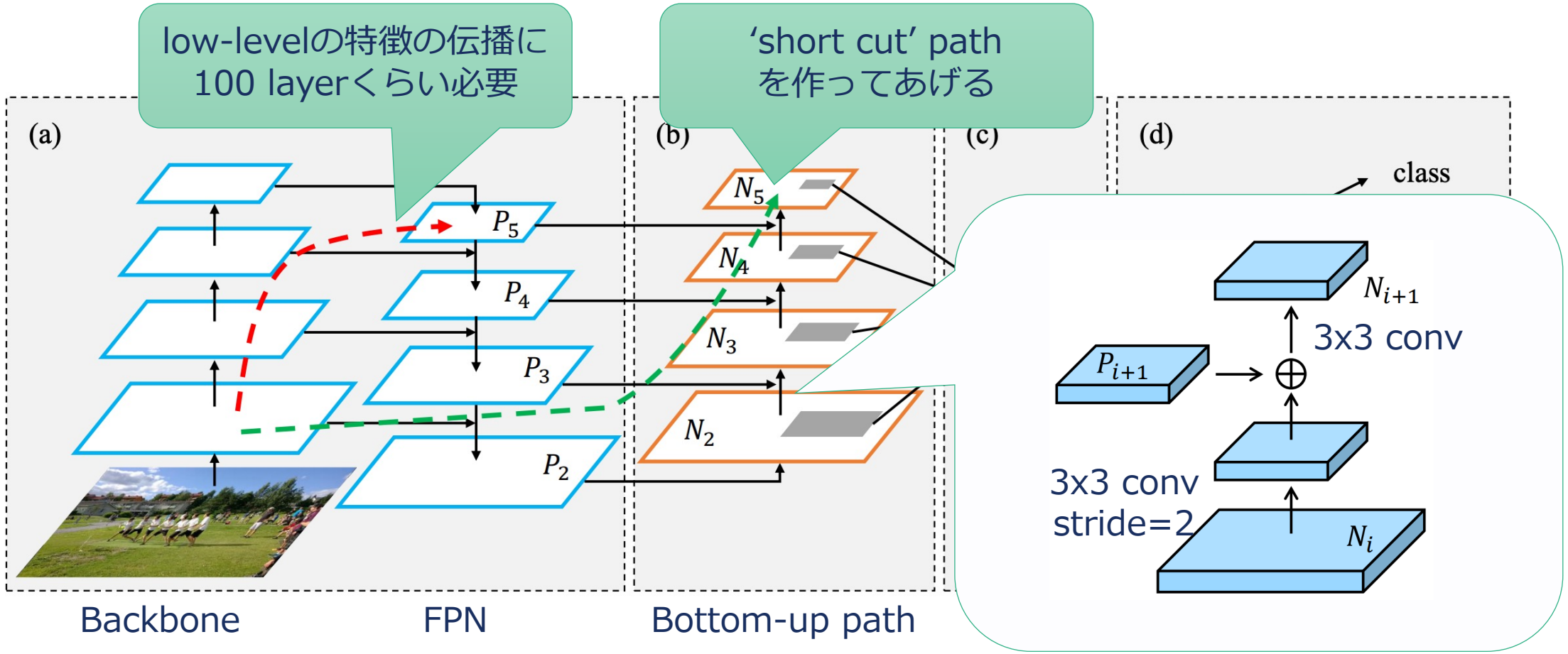
■エッジ等のlow-levelの情報をネットワーク全体に伝播させる



S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in Proc. of CVPR, 2018.

Path Aggregation Network (PANet)

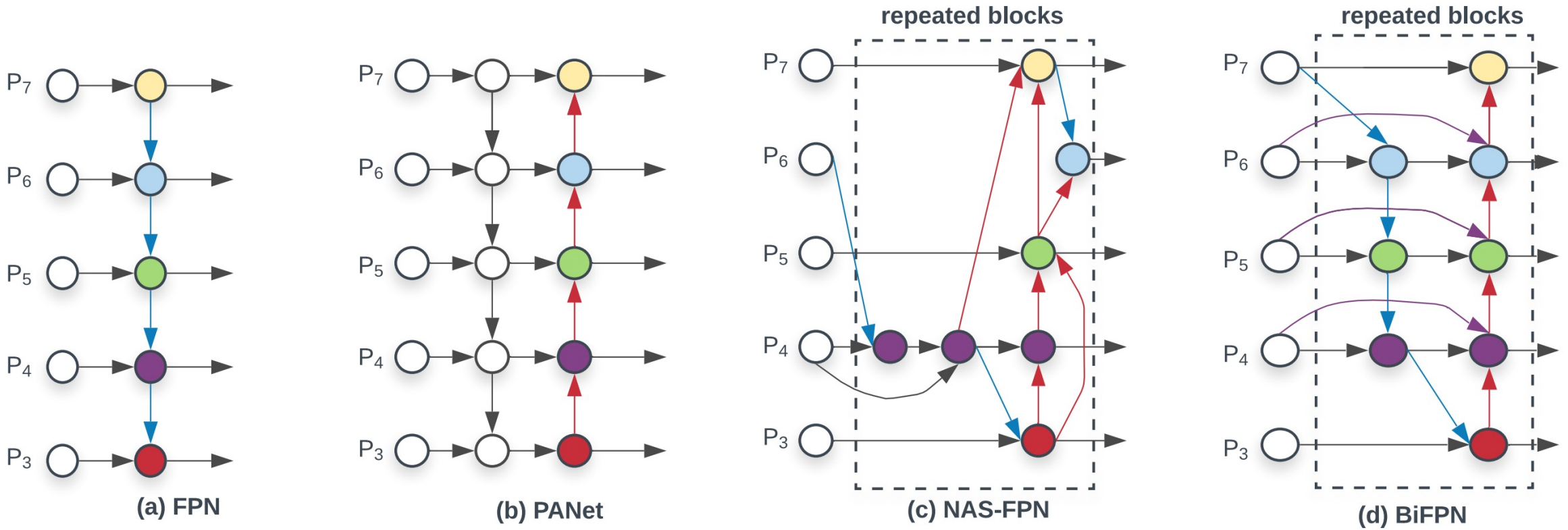
■エッジ等のlow-levelの情報をネットワーク全体に伝播させる



S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in Proc. of CVPR, 2018.

Bi-directional Feature Pyramid Network (BiFPN)

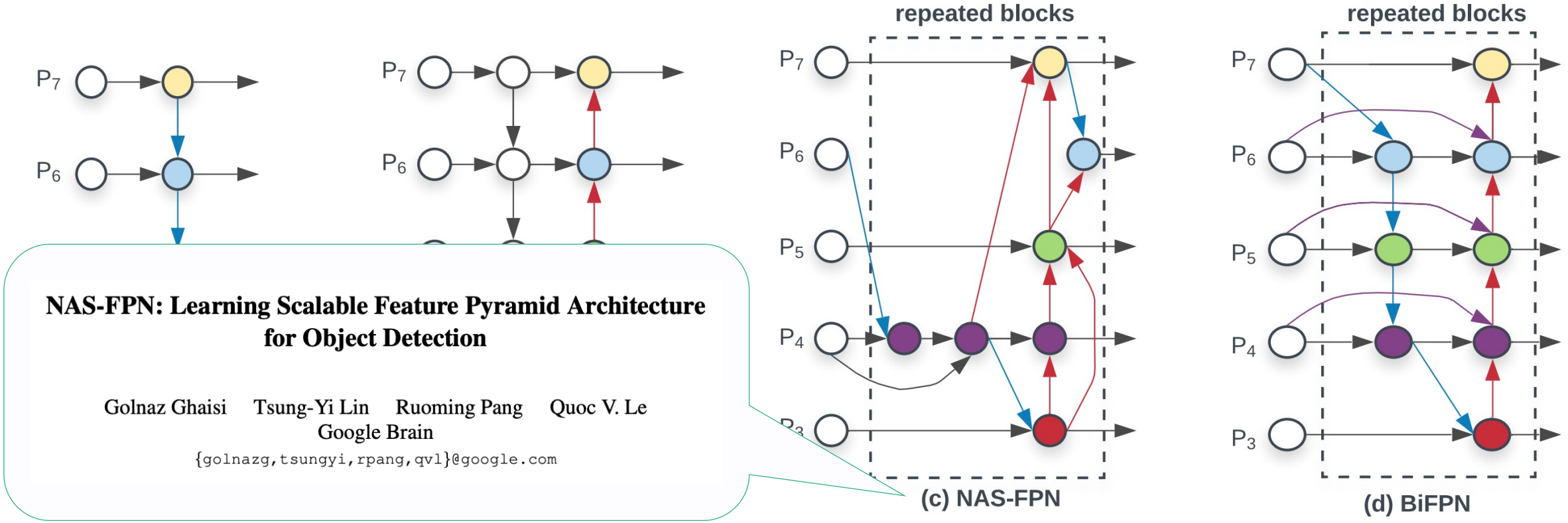
■PANetを簡略化、同一解像度のskip connection、top-down+bottom-upを1モジュールとして繰り返す（単一モジュールとして考えることで簡略化が可能に）



M. Tan, R. Pang, and Quoc V. Le, "EfficientDet: Scalable and Efficient Object Detection," in Proc. of CVPR, 2020.

Bi-directional Feature Pyramid Network (BiFPN)

■PANetを簡略化、同一解像度のskip connection、top-down+bottom-upを1モジュールとして繰り返す（単一モジュールとして考えることで簡略化が可能に）



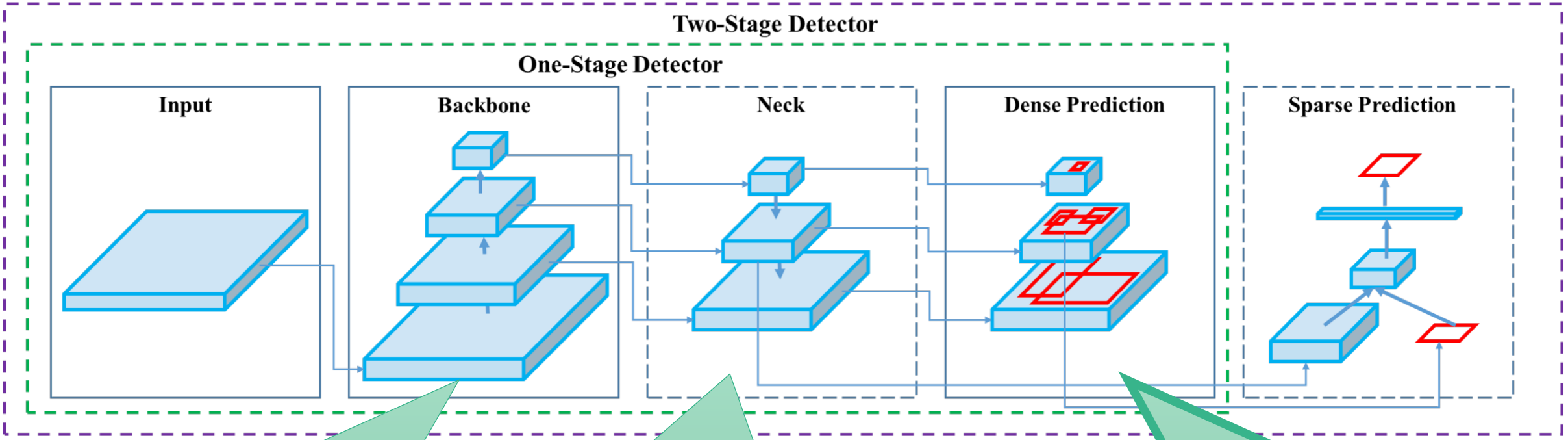
NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection

Golnaz Ghaisi Tsung-Yi Lin Ruoming Pang Quoc V. Le
Google Brain
{golnazg,tsungyi,rpang,qvl}@google.com

M. Tan, R. Pang, and Quoc V. Le, "EfficientDet: Scalable and Efficient Object Detection," in Proc. of CVPR, 2020.

物体検出モデルの汎用的な表現

■Backbone, Neck, Headの組み合わせで物体検出モデルは表現できる



Backbone:
ベースとなるクラス分類モデル。Multi-scaleの特徴マップを出力
(e.g. ResNet, Darknet)

Neck:
Multi-scaleの特徴マップを入力してコネコネして出力
（e.g. FPN, BiFPN）

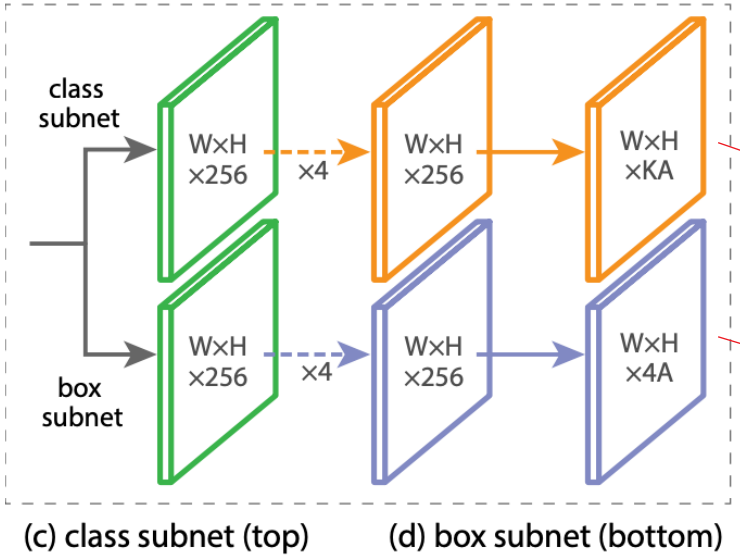
Head:
Multi-scaleの特徴マップを入力して検出結果を出力
(e.g. YOLO/Retina head)

A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," in arXiv, 2020.

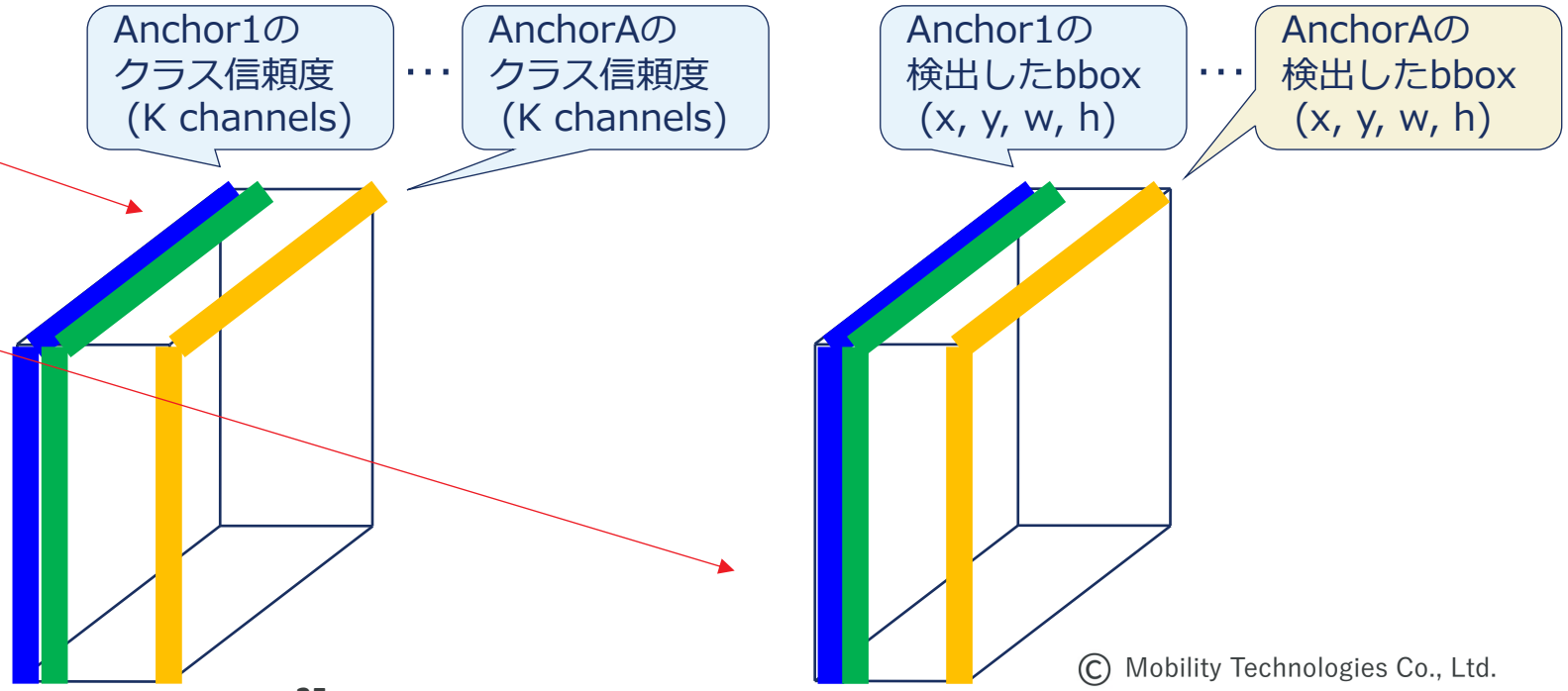
Anchor

■各スケールのHeadの特徴マップの座標毎にA個の「Anchor」が定義されている
- Anchor: 特定の条件の物体のみを検出する部品
- Bounding box (bbox) のサイズで定義される。YOLOv3はA=3, RetinaNetはA=9

■Anchorのbboxとmatchingルールによって「各Anchorの守備範囲」が決まる
- 各Anchorがどういうサイズの物体を検出すべきか（&検出すべきでないか）



RetinaNetのHead



AnchorとのMatching

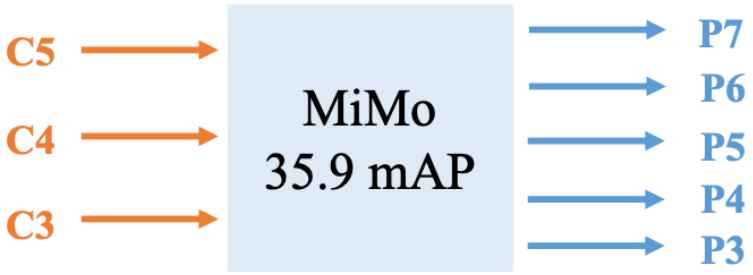
- AnchorとのMatchingとは、ground truth (GT) の各objectをどのAnchorが検出すべき（&すべきでない）かを決めるプロセス
  - これにより特徴マップのどこにどういうロスをかけるかが決まる
- RetinaNetでは…
  - IoUが0.5以上のAnchorが検出すべき（positive Anchor）
  - IoUが0.4以下のAnchorは検出すべきではない（negative Anchor）
  - どちらでもないAnchorを残すことは重要（個人的意見）
    - ギリギリのAnchorにはどちら側のロスをかけることも不適切
- 手法によってMatching手法にかなり細かい違いがある
  - Digging into Sample Assignment Methods for Object Detection
  - https://speakerdeck.com/hirotohonda/digging-into-sample-assignment-methods-for-object-detection
  - The devil is in the details…

論文の主張

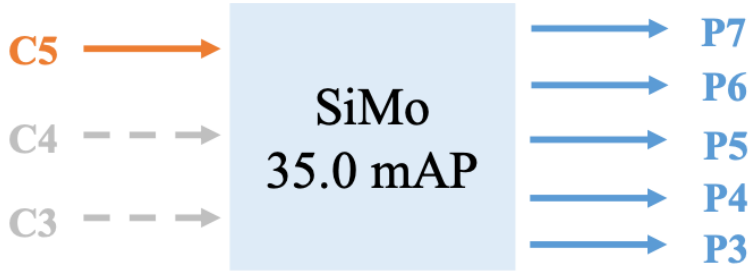
- Feature Pyramids Networks (FPN) はマルチスケールの特徴を融合することにより性能が向上していると思われているがポイントはそこやないで
- 物体検出における最適化問題を（multi-scaleのアンカーを使うことで）分割統治的に解いているところが一番ポイントやで
- でもマルチスケールの特徴を使った検出は複雑かつ低速なので、single-scaleでmulti-scaleに匹敵する検出器をつくるお！

色々なNeckを比較

FPN



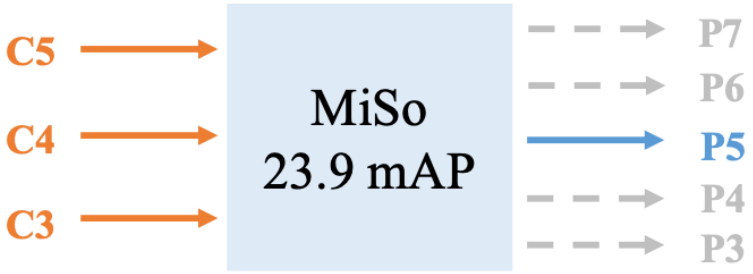
(a) Multiple-in-Multiple-out



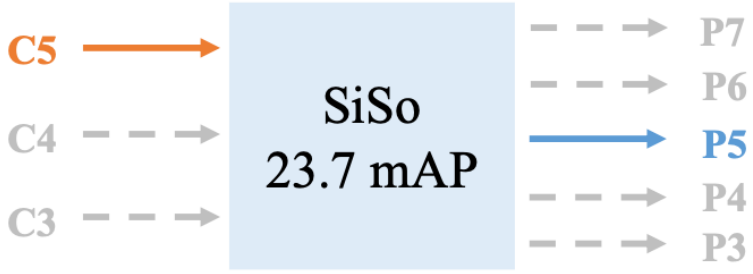
(b) Single-in-Multiple-out

単一スケールから無理やり複数スケールの特徴を出力

複数スケールを統合し単一スケールの特徴を出力



(c) Multiple-in-Single-out



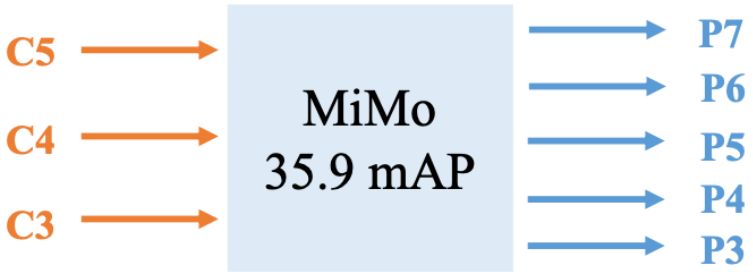
(d) Single-in-Single-out

単一スケールの特徴をそのまま出力

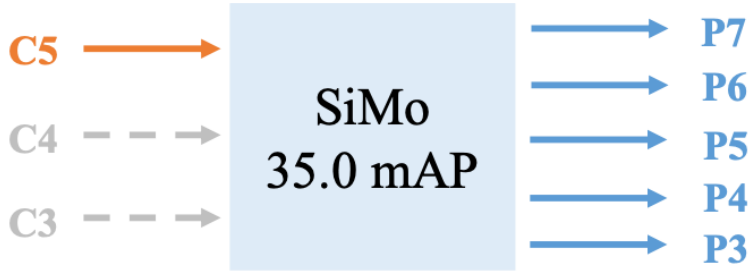
- (a) と (b) を比較すると、マルチスケールの特徴を融合することによる影響はそこまで大きくない
- (a) と (c)、(b) と (d) を比較するとsingle outputによる性能低下が著しい

色々なNeckを比較

FPN



(a) Multiple-in-Multiple-out

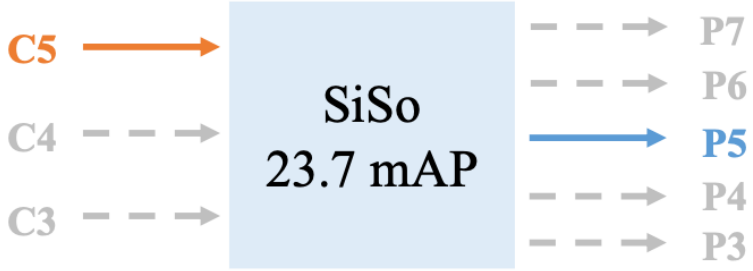


(b) Single-in-Multiple-out

単一スケールから無理やり複数スケールの特徴を出力

複数スケールを統合し単一スケールの特徴を出力

(c) Multiple-in-Single-out



(d) Single-in-Single-out

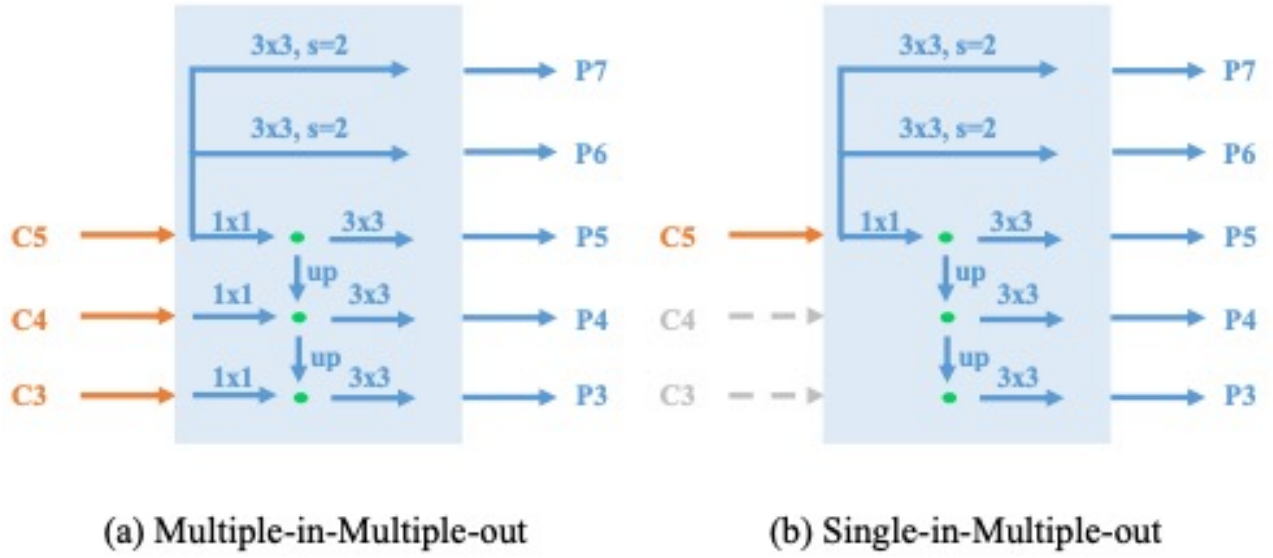
単一スケールの特徴をそのまま出力

■C5特徴は様々なスケールの特徴を検出する情報を十分に持っている

■FPNにおけるマルチスケールの特徴を融合するメリットは、multiple outputにより実現される分割統治のメリットには遠く及ばない

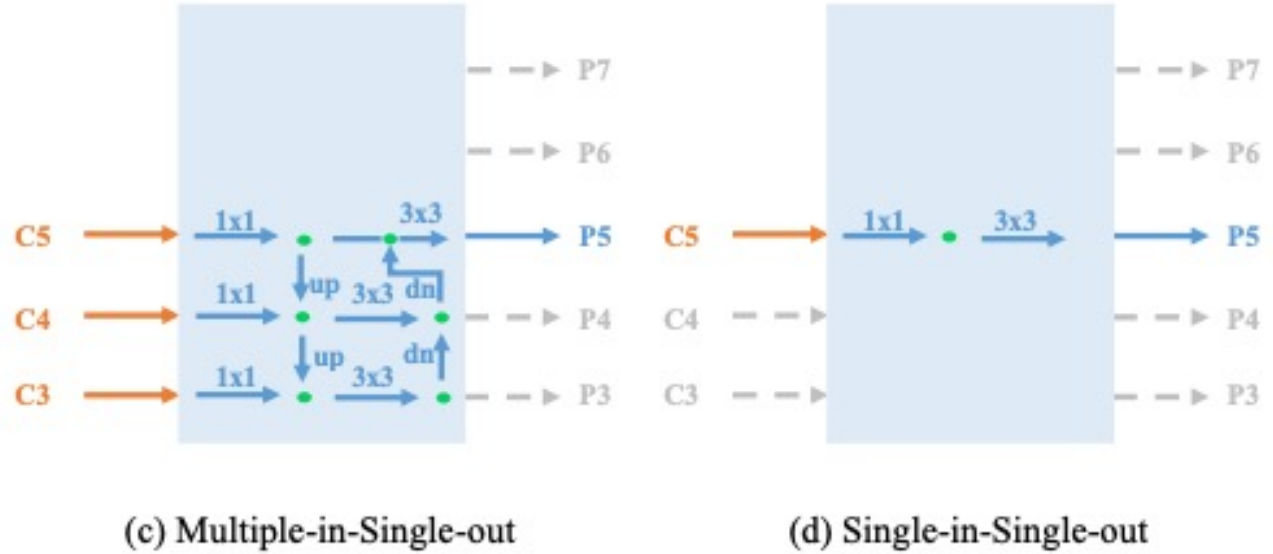
色々なNeckを比較

FPN



単一スケールから無理やり複数スケールの特徴を出力

複数スケールを統合し単一スケールの特徴を出力

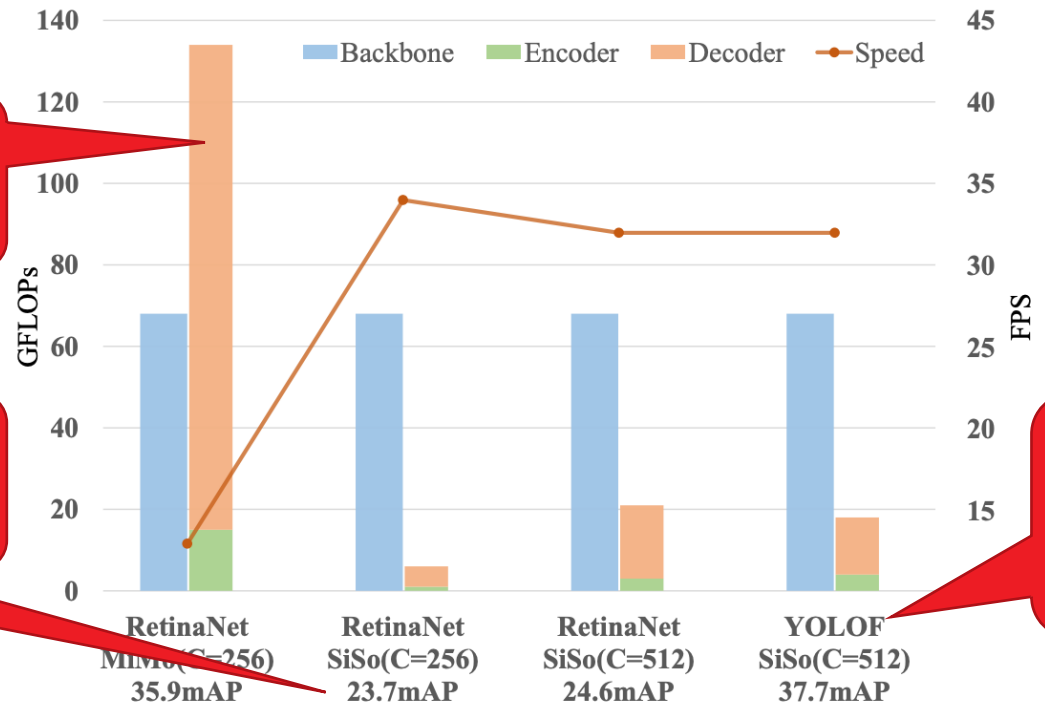


単一スケールの特徴をそのまま出力

じゃあmultiple outputでええやん？

■Multiple outputは計算量が大きい

Backbone → Encoder (Neck) → Decoder (Head) →



Multiple output はHeadが重い

Single outputにすると精度が下がる

Single outputでも精度が維持できる手法を提案するお

他の論文でも

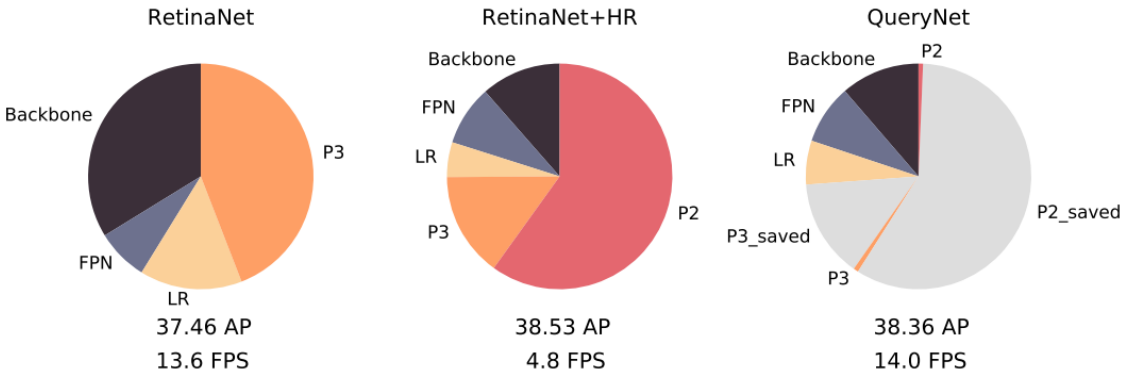
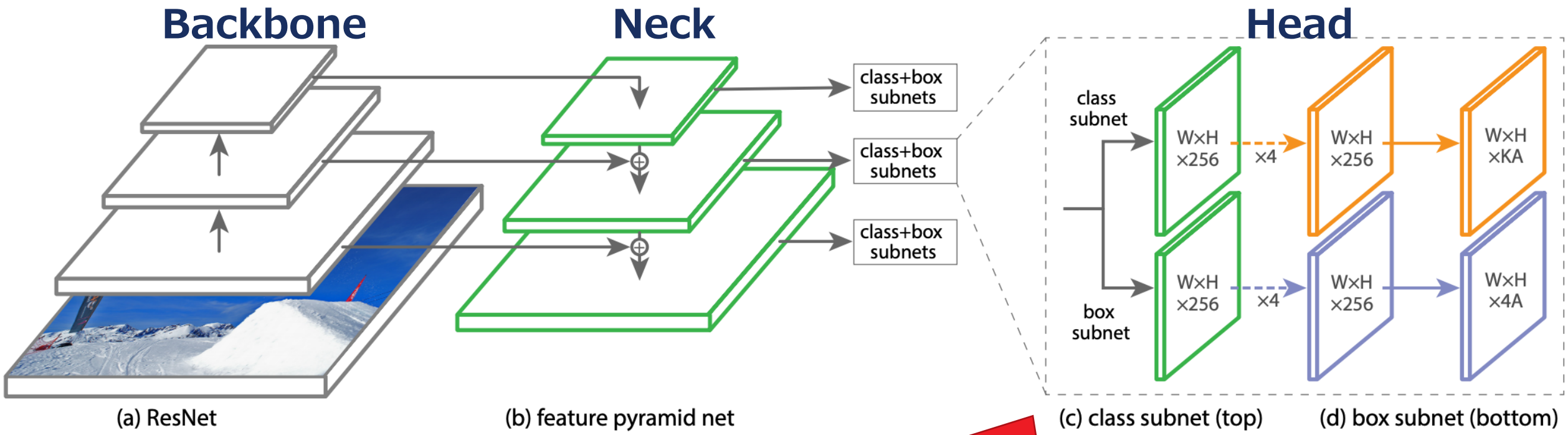


Figure 2. The FLOPs distribution of different module when using ResNet-50 backbone. In RetinaNet, the computational cost on the high-resolution P_3 accounts for 43% of the total cost; when the higher-resolution P_2 is added, they together account for 74% of the total cost. Our QueryDet can effectively reduce the computation on those features by 99%, leading to a fast inference speed and keeping a high detection accuracy. Note LR stands for the low-resolution P_4 to P_7.

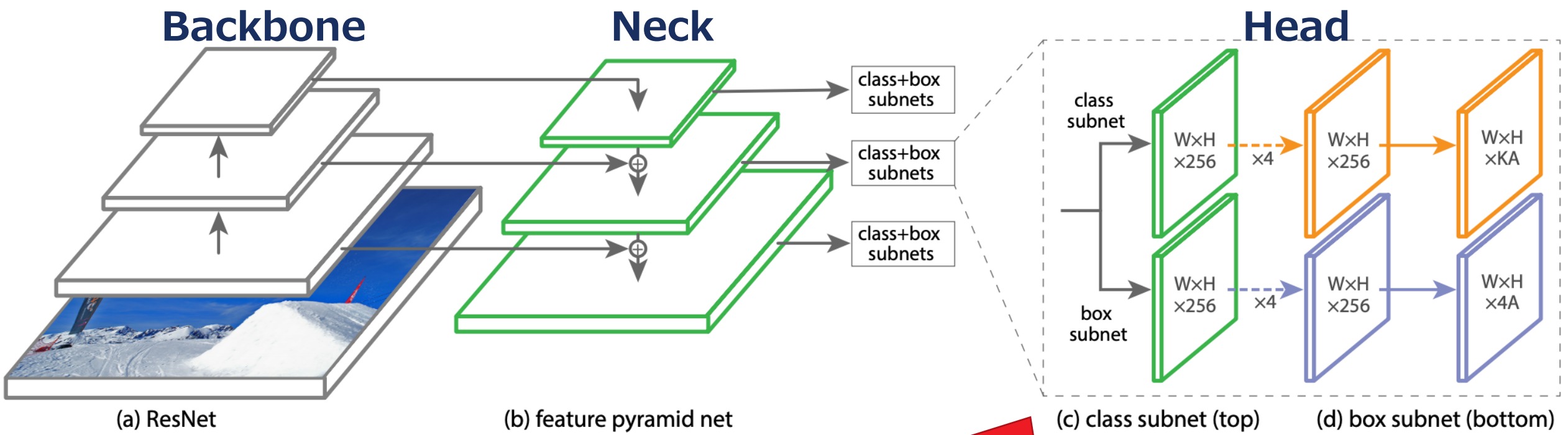
C. Yang, Z. Huang, and N. Wang, "QueryDet: Cascaded Sparse Query for Accelerating High-Resolution Small Object Detection," in arXiv, 2021.
https://speakerdeck.com/keiku/querydet-cascaded-sparse-query-for-accelerating-high-resolution-small-object-detection

Multiple outputのHeadが何故重いかというと



複数スケール間で重み共有のhead
channel数256

Multiple outputのHeadが何故重いかというと



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

複数スケール間で重み共有のhead channel数256

ResNetのC3特徴のchannel数は128（計算量は4倍）

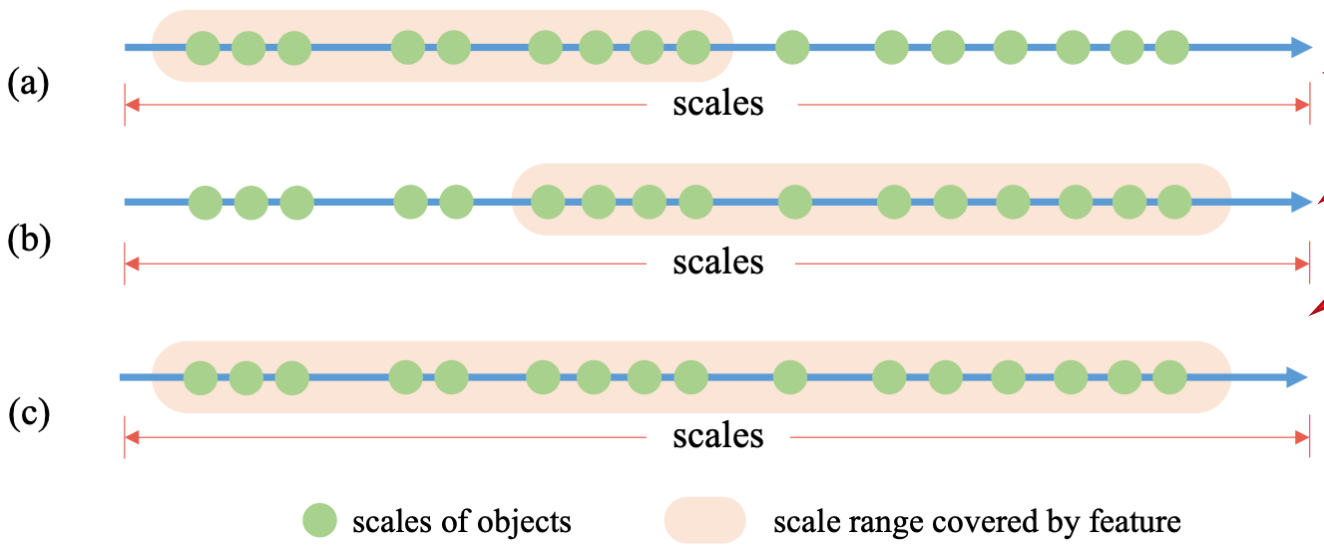
SiSoにおける課題

■C5特徴が対応できる物体の大きさが限られている

■Positive anchorの不均衡問題

C5特徴が対応できる物体の大きさが限られている

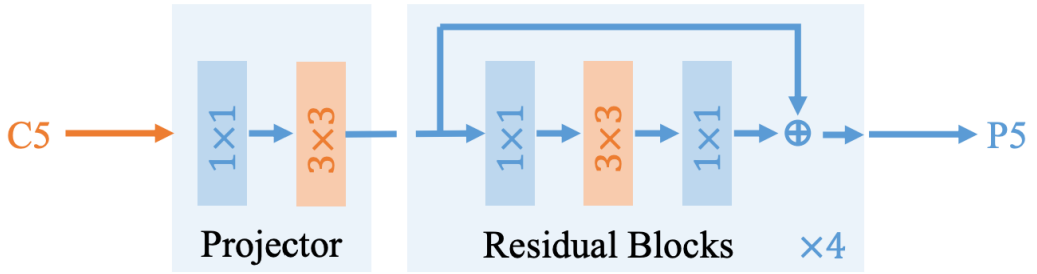
- RetinaNetではstride-2のconvで作成されたreceptive fieldの大きな特徴マップP6, P7を利用している
- 複数の特徴マップを使いたくない病のYOLOFではdilated convolutionでreceptive fieldを拡大することを提案
- Residual構造とすることで小さい物体用も引き続きカバー



C5特徴

C5特徴にdilated conv

C5特徴にdilated conv + skip

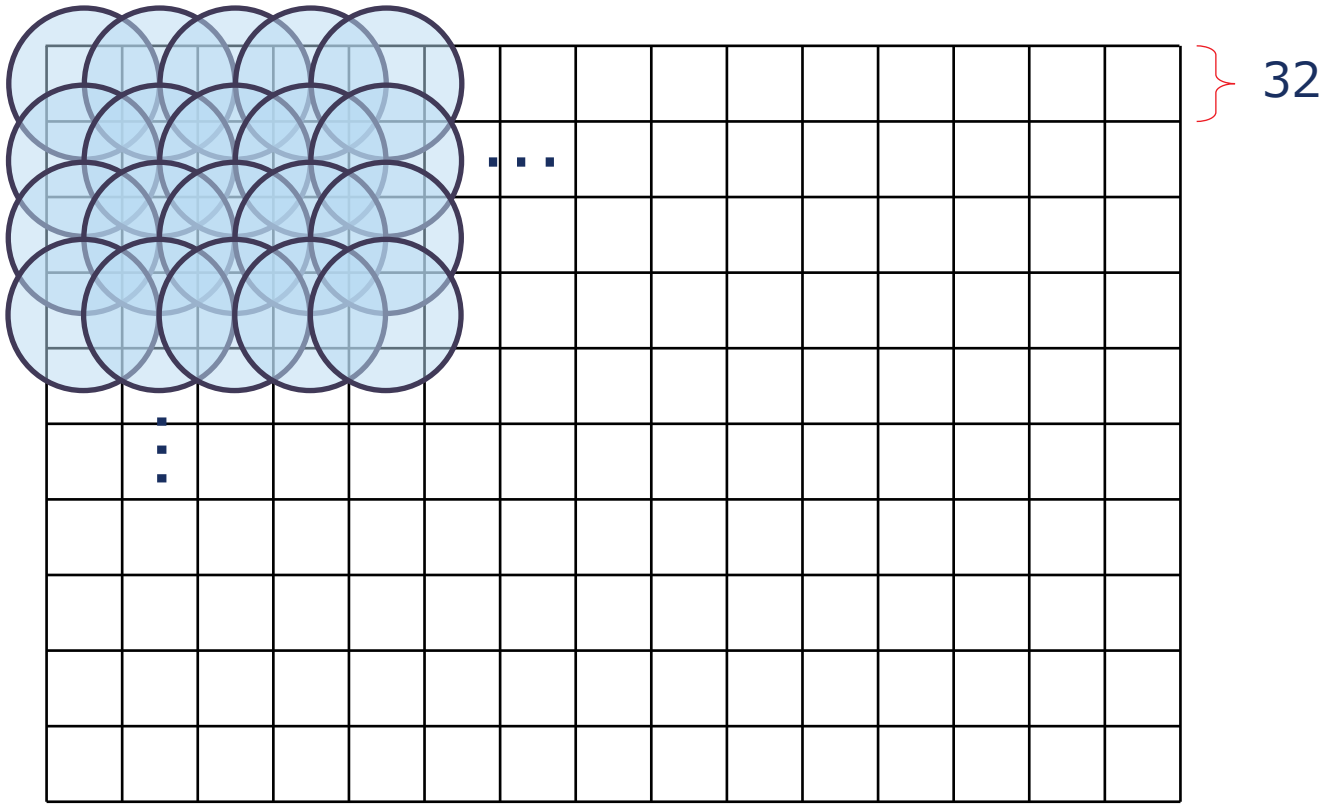


Positive anchorの不均衡問題

- YOLOFではP5特徴に32, 64, 128, 256, 512サイズのアンカーが存在（アスペクト比は固定）

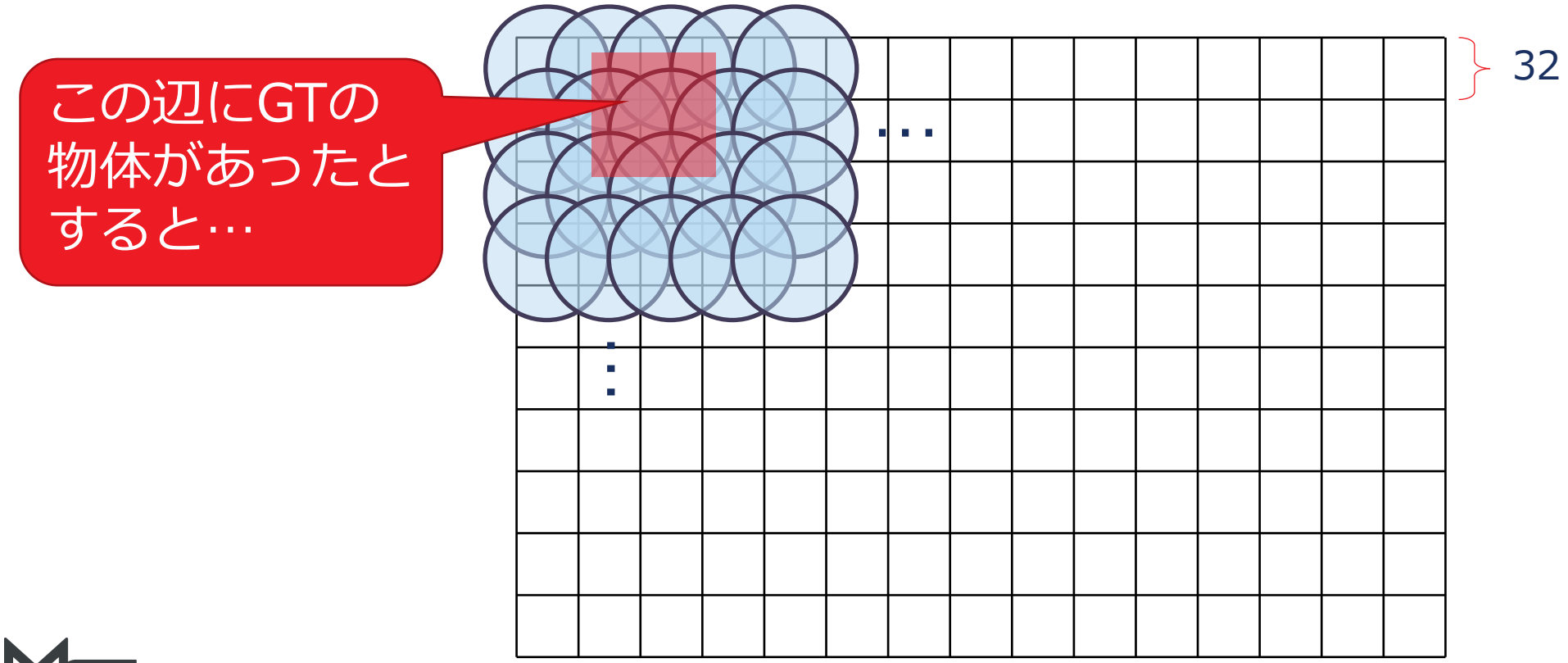
Positive anchorの不均衡問題

- YOLOFではP5特徴に32, 64, 128, 256, 512サイズのアンカーが存在（アスペクト比は固定）
- 64サイズのアンカーはこんな感じ



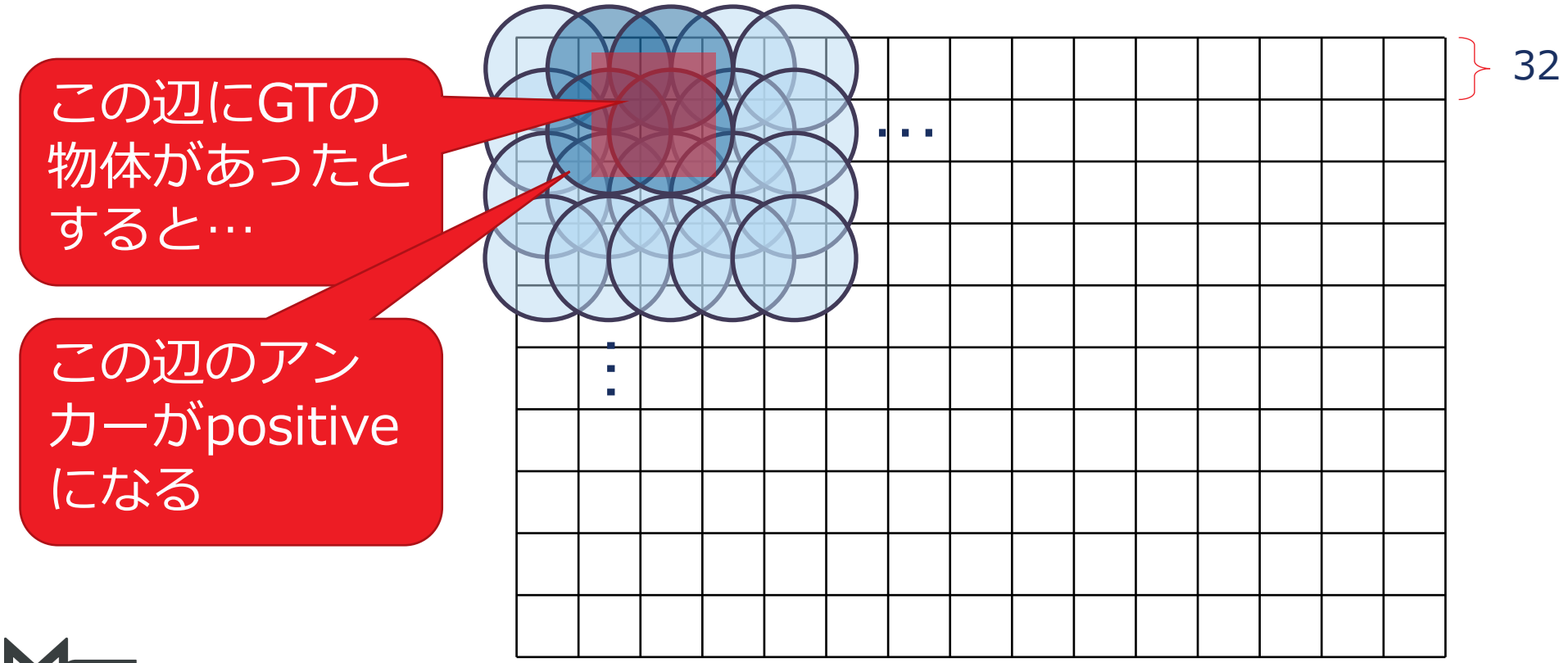
Positive anchorの不均衡問題

- YOLOFではP5特徴に32, 64, 128, 256, 512サイズのアンカーが存在（アスペクト比は固定）
- 64サイズのアンカーはこんな感じ



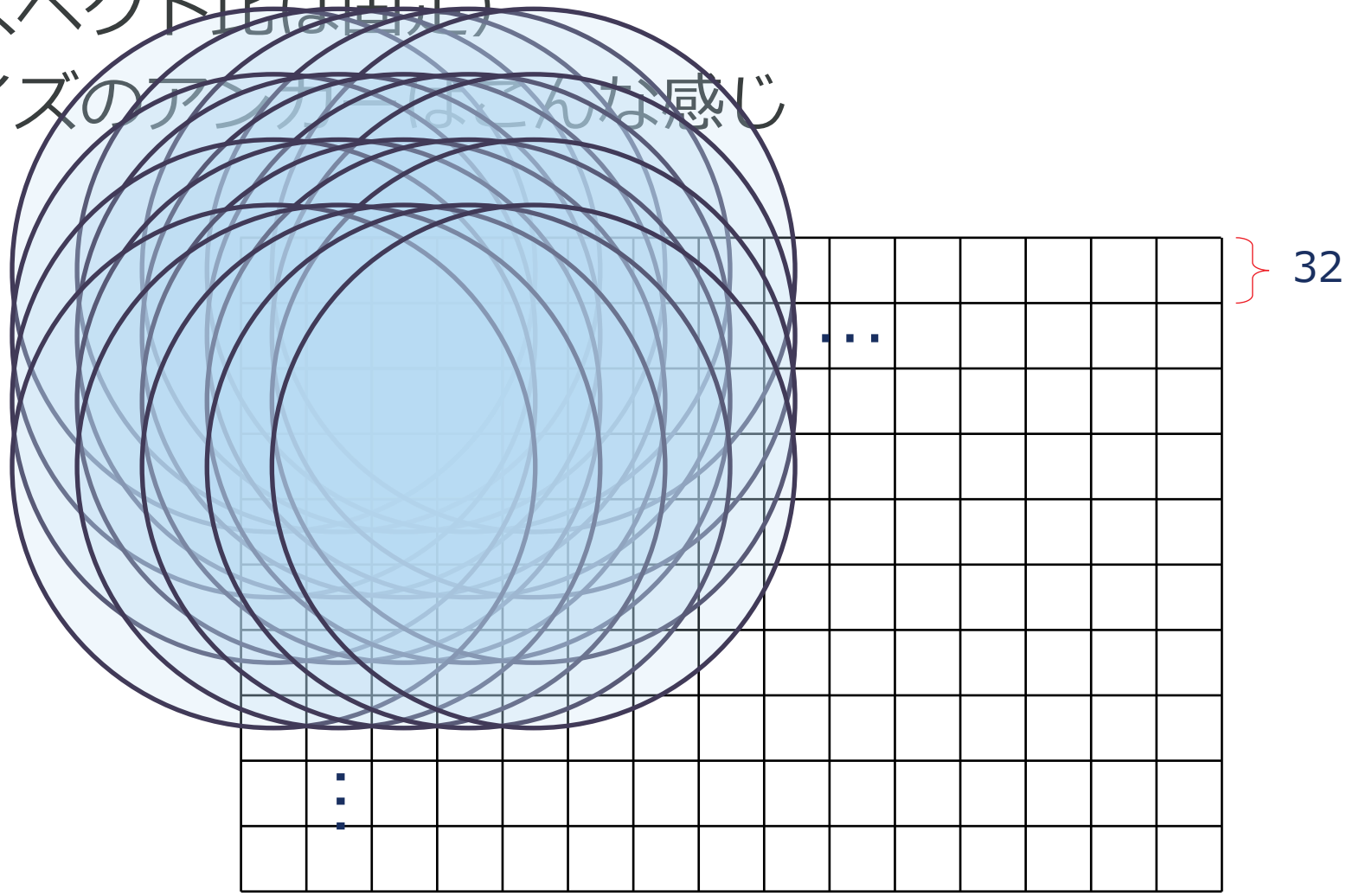
Positive anchorの不均衡問題

- YOLOFではP5特徴に32, 64, 128, 256, 512サイズのアンカーが存在（アスペクト比は固定）
- 64サイズのアンカーはこんな感じ



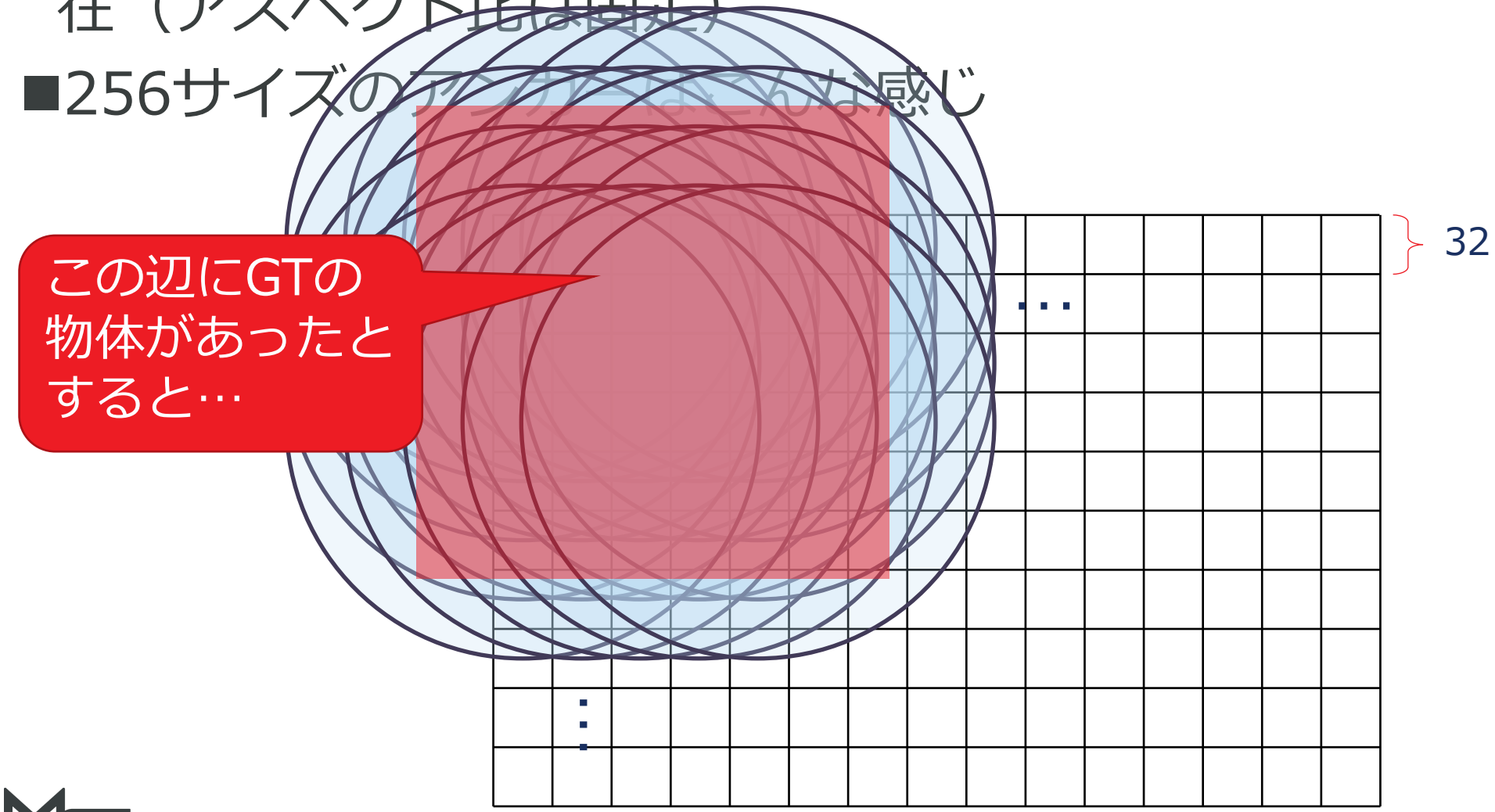
Positive anchorの不均衡問題

- YOLOFではP5特徴に32, 64, 128, 256, 512サイズのアンカーが存在（アスペクト比は固定）
- 256サイズのアンカーはこんな感じ



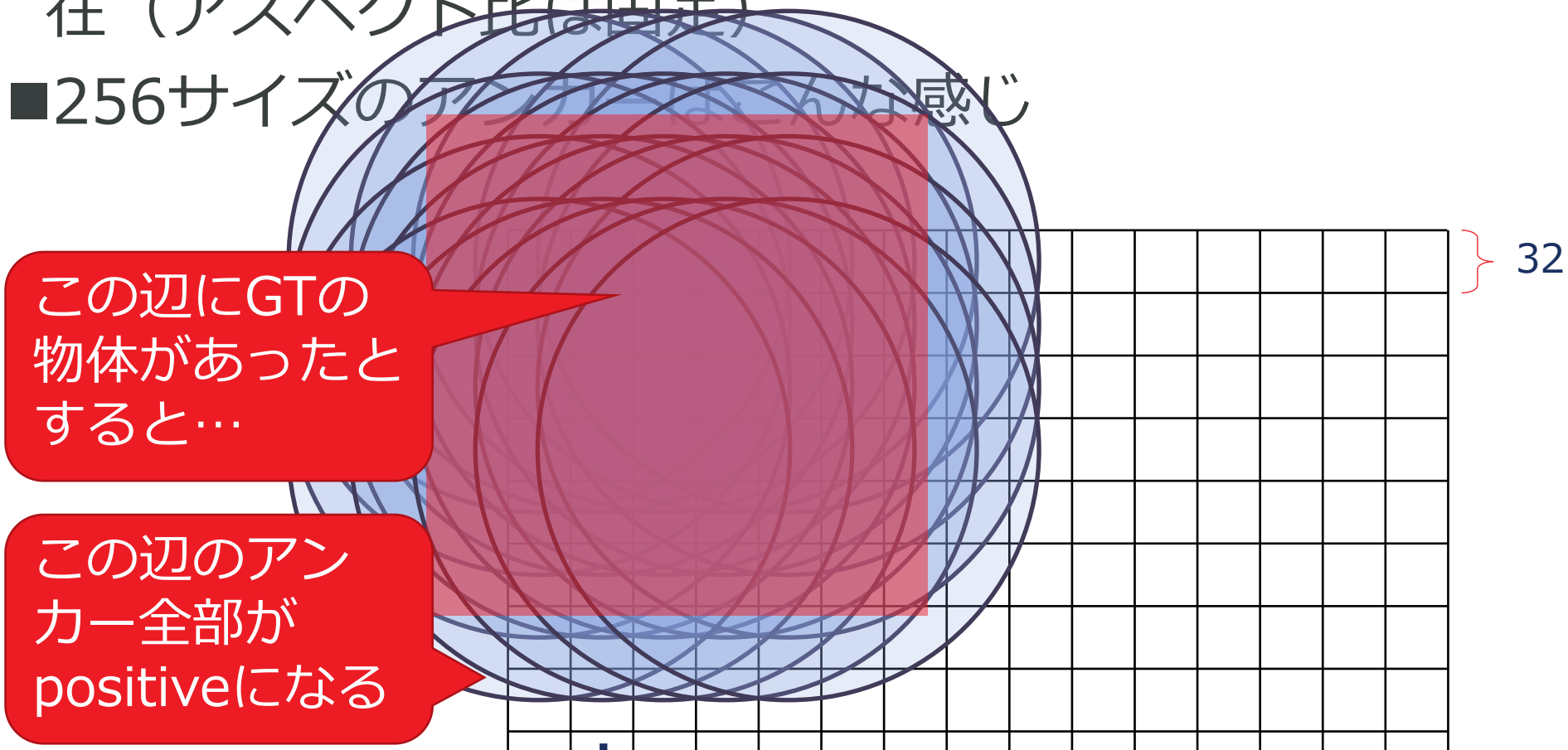
Positive anchorの不均衡問題

- YOLOFではP5特徴に32, 64, 128, 256, 512サイズのアンカーが存在（アスペクト比は固定）
- 256サイズのアンカーはこんな感じ



Positive anchorの不均衡問題

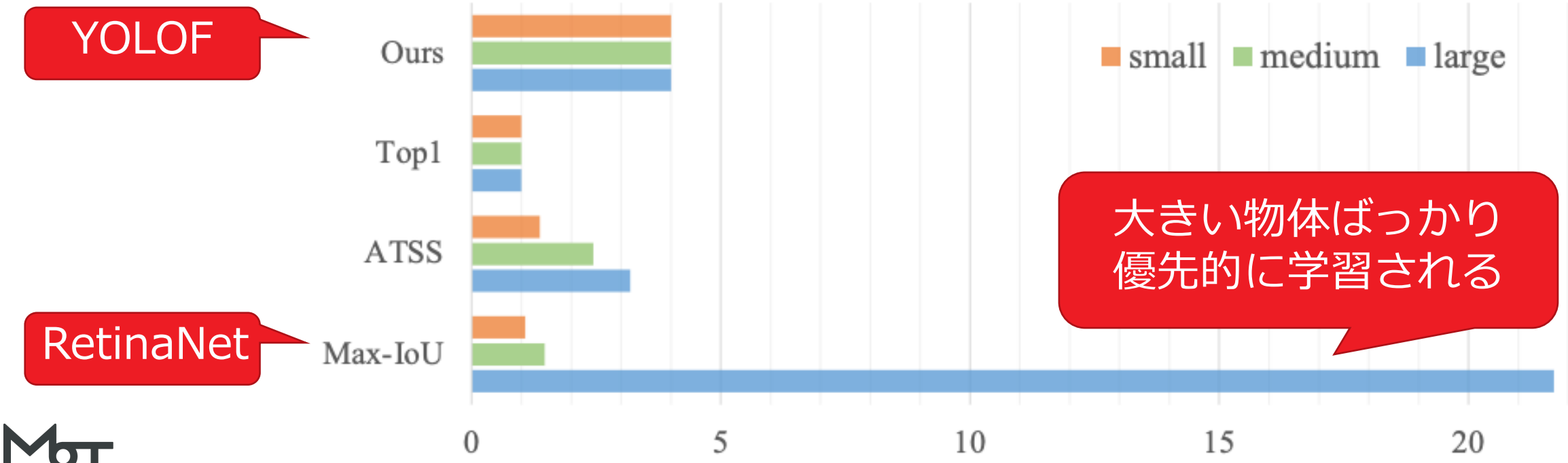
- YOLOFではP5特徴に32, 64, 128, 256, 512サイズのアンカーが存在（アスペクト比は固定）
- 256サイズのアンカーはこんな感じ



Positive anchorの不均衡問題

■ということが言いたいのが論文のこの図
- 横軸が1GTあたりの異なる物体サイズ毎のpositive Anchor数
- 縦軸で異なるmatching手法を比較している

■この問題に対応するためYOLOFでは固定のtop-k (k=4) をpositiveとすることを提案



ちなみに

■“we set IoU thresholds in Uniform Matching to ignore large IoU (>0.7) negative anchors and small IoU (<0.15) positive anchors.”

■特に大きなアンカーはIoUが大きくなるアンカーが大量に出る

- これらに対してnegativeなlossをかけるのはよろしくない

■RetinaNet等、複数スケールの特徴を利用する場合、大きな物体を担当する特徴マップは低解像度で、アンカーは前述のように細かく配置されていないためこの問題は顕著ではない

関連手法

■Adaptive Training Sample Selection (ATSS)

- Anchor-basedな手法とAnchor-freeな手法のパフォーマンスの差は（色々な細かい改善手法と）positive, negative Anchorを定義するmatchingアルゴリズムの差であることを指摘
- 各GT毎に、近傍アンカーとのIoUとの統計量を基に適応的にpositive, negativeへアサインするためのしきい値を決定する手法を提案

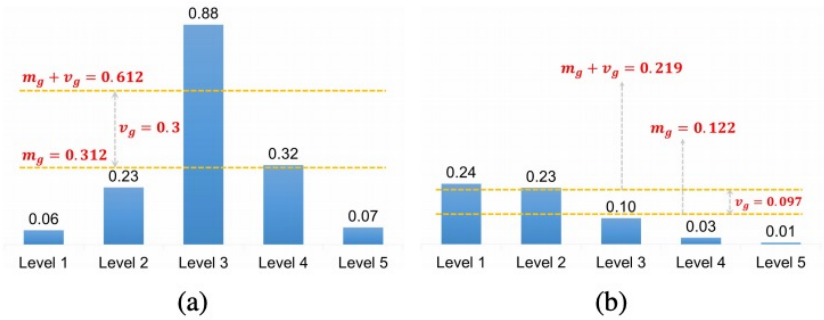


Figure 3: Illustration of ATSS. Each level has one candidate with its IoU. (a) A ground-truth with a high m_g and a high v_g. (b) A ground-truth with a low m_g and a low v_g.

S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Li, "Bridging the Gap Between Anchor-based and Anchor-free Detection via Adaptive Training Sample Selection," in Proc. of CVPR, 2020.

関連手法

■YOLO, YOLOv2はpositiveはbest matchの1件のみ

■DETRはハンガリアンアルゴリズムでglobalかつ暗黙的なAnchorとのmatchingを最適化している

■YOLOFはpositiveの個数に着目してバランスすることを目的にしている（というexcuse

■他にもCVPR’21で、GTとAnchorのassignを最適化する手法が出ている

J. Wang, L. Song, Z. Li, H. Sun, J. Sun, and N. Zheng, "End-to-End Object Detection with Fully Convolutional Network," in Proc. of CVPR, 2021.
Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, Jian Sun, "OTA: Optimal Transport Assignment for Object Detection," in Proc. of CVPR, 2021.

YOLOFまとめ

- NeckとしてC5特徴入力し、P5特徴を出力するdilated convモジュールを利用
- 1つの特徴マップに全スケールのアンカーを押し込んだRetina Headを利用
- GTからtop-k (k=4) のAnchorをpositiveAnchorとする

結果: ベースであるRetinaNetとの比較

■RetinaNetより同等以上の精度で早い

Model	schedule	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	#params	GFLOPs	FPS
RetinaNet [23]	1x	35.9	55.7	38.5	19.4	39.5	48.2	38M	201	13
RetinaNet-R101 [23]	1x	38.3	58.5	41.3	21.7	42.5	51.2	57M	266	11
RetinaNet+	1x	37.7	58.1	40.2	22.2	41.7	49.9	38M	201	13
RetinaNet-R101+	1x	40.0	60.4	42.7	23.2	44.1	53.3	57M	266	10
YOLOF	1x	37.7	56.9	40.6	19.1	42.5	53.2	44M	86	32
YOLOF-R101	1x	39.8	59.4	42.9	20.5	44.5	54.9	63M	151	21
YOLOF-X101	1x	42.2	62.1	45.7	23.2	47.0	57.7	102M	289	10
YOLOF-X101†	3x	44.7	64.1	48.6	25.1	49.2	**60.9**	102M	289	10
YOLOF-X101†‡	3x	**47.1**	**66.4**	**51.2**	**31.8**	**50.9**	60.6	102M	-	-

RetinaNet+: YOLOF実装に合わせたRetinaNet

Mobility Technologies

結果: 同じC5特徴だけを利用するDETRとの比較

■7倍早く収束する！

Model	Epochs	#params	GFLOPS/FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR [4]	500	41M	86/24*	42.0	62.4	44.2	20.5	45.8	61.1
DETR-R101 [4]	500	60M	152/17*	43.5	**63.8**	46.4	21.9	48.0	**61.8**
YOLOF	72	44M	86/32	41.6	60.5	45.0	22.4	46.2	57.6
YOLOF-R101	72	63M	151/21	**43.7**	62.7	**47.4**	**24.3**	**48.3**	58.9

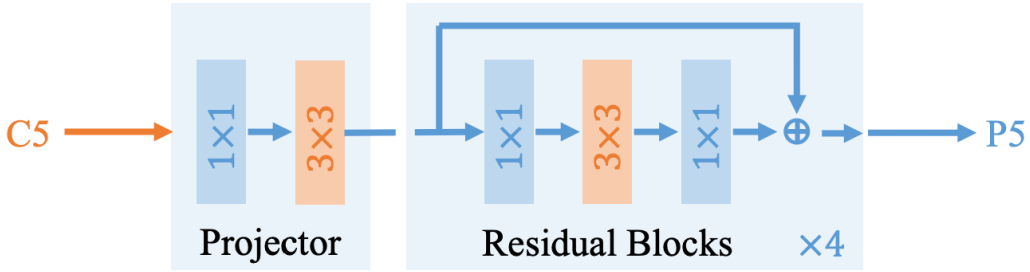
■大きな物体はDETRのほうが得意

結果: Single shot detectorと言えば…のYOLO系と比較

■ちょっと早くてちょっと精度が良い

Model	Epochs	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4 [1]	273	53*	43.5	**65.7**	47.3	**26.7**	47.6	53.3
YOLOF-DC5	184	**60**†	**44.3**	62.9	**47.5**	24.0	**48.5**	**60.4**

Ablation Study: Neck構造



N	AP	AP_s	AP_m	AP_l
0	33.8	17.7	40.9	43.8
2	34.9	17.8	41.3	46.8
4	**35.5**	**17.6**	**41.4**	**48.4**
6	36.0	17.7	41.9	49.5
8	36.6	18.5	42.0	50.7
10	36.9	18.3	42.4	50.4

(a) **Number of ResBlocks** (ResNet-50): More residual blocks bring more gains. N represent the number of ResBlocks. To keep YOLOF simple and neat, we add 4 blocks in the encoder by default.

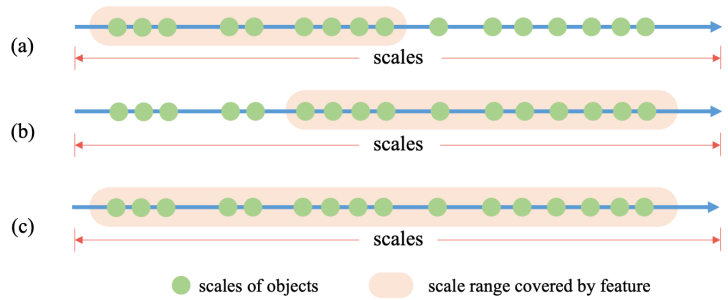
Dilations	AP	AP_s	AP_m	AP_l
1,1,1,1	35.5	17.6	41.4	48.4
2,2,2,2	36.4	18.1	41.8	50.2
3,3,3,3	36.9	18.4	42.1	51.0
1,2,3,4	37.4	18.6	42.6	51.8
2,4,6,8	**37.7**	**19.1**	**42.5**	**53.2**
3,6,9,12	37.3	18.7	42.1	52.6

(b) **Different dilations** (ResNet-50-N4): 'N4' means we add 4 ResBlocks in the encoder. Dilation in the residual block gives large gains on large objects and slightly improve the performance of small and medium objects.

Dilations & Shortcut	AP	AP_s	AP_m	AP_l
2,4,6,8 ✓	**37.7**	**19.1**	**42.5**	**53.2**
2,4,6,8 -	34.1	16.2	38.4	47.5
1,1,1,1 ✓	35.5	17.6	41.4	48.4
1,1,1,1 -	32.6	15.0	38.4	44.2

(c) **Add shortcut or not** (ResNet-50): YOLOF results with shortcuts or not on various dilation settings. Shortcut brings considerable gains on all object scales and becomes more important when the dilations are adopted (+3.6 AP with dilations 2,4,6,8 vs. +2.9 AP when dilations are all ones).

- ■ResBlockは多いほうが良い（が4つで勘弁してやる）
- ■Dilationは2,4,6,8
  - 1,1,1,1の精度が悪いので単に深くするだけでは駄目
- ■Residual機構はあった方が良い
  - 全スケール良くなっているので元々の複数スケールカバーする云々は…↑



Ablation Study: matching部分

topk	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
top1	35.9	55.6	38.4	17.5	40.3	50.2
top2	37.2	56.7	39.9	18.9	41.6	52.0
top3	37.5	**57.1**	40.2	18.6	41.9	52.5
top4	**37.7**	56.9	**40.6**	**19.1**	**42.5**	**53.2**
top5	37.5	56.7	40.3	18.1	42	**53.2**

(d) **Number of positives** (ResNet-50-N4): Number of positive anchors in *Uniform Matching*. Increase the positive anchor for each ground-truth box can improve the performance while it saturates when too many positive anchors. We choose the top4 anchors in YOLOF which achieves best results.

Matching Methods	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Max-IoU Matching [23]	29.1	45.9	29.6	9.5	32.2	50.6
ATSS(topk=9) [48]	34.6	54.3	37.1	17.7	40.6	46.9
ATSS(topk=15) [48]*	36.5	55.9	38.6	18.1	41.4	50.8
Hungarian Matching [4]	35.8	55.5	38.3	18.2	39.9	50.2
Uniform Matching	**37.7**	**56.9**	**40.6**	**19.1**	**42.5**	**53.2**

(e) **Uniform matching *vs.* other matchings** (ResNet-50-N4): Comparison with other matching methods. Uniform Matching achieve balance in positive anchors and get the best results among other matching methods, which is consistent with the comparison in Figure 6. Note that '*' represents that we get the best result for ATSS [48] when setting topk as 15. More details can be found in the Appendix.

■Uniformにtop-4が良い

所感

- C5特徴でも小さい物体を検出できる部分は面白い
- 精度速度のトレードオフを追い求めるという観点では、複数スケールの特徴を使って重くなるのはstride=8, 16のところなので、そこだけ使わずにP5-7は使うでよいのではないか
  - 複数スケール使いたくない病なら仕方がない
- YOLOv3~はstride=8の特徴マップを使っているがweight sharedの重いHeadを使っていないので問題ない

- Anchorの定義、Anchor matching, NMSあたりはまだまだ綺麗な手法があるのでは？
  - 高解像度特徴まで効率的に見るようなDETRが全てを解決する？

Acknowledgement

■資料作成に当たり色々議論してくれた
同僚の @hirotomusiker に感謝！