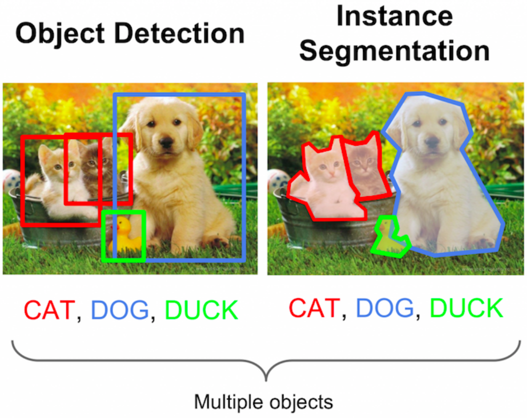
# TensorMask: A Foundation for Dense Object Segmentation

Xinlei Chen, Ross Girschick, Kaiming He, Piotr Dollar
presented by Yonk Shi

# Recap: Object detection and Instance Segmentation

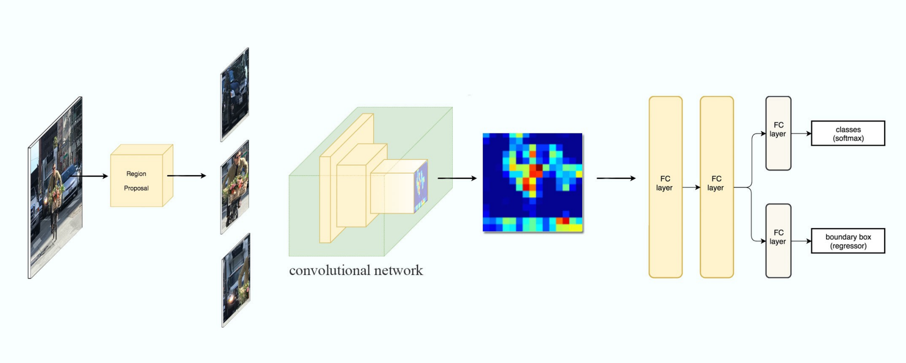
Object detection and instance segmentation are closely related tasks. Predominantly two approaches in object detection & instance segmentation:

- Region Proposal based method (e.g. RCNN)
- Sliding Windows based method (e.g. SSD, RetinaNet)



# Recap: Region Proposal Based Object Detection

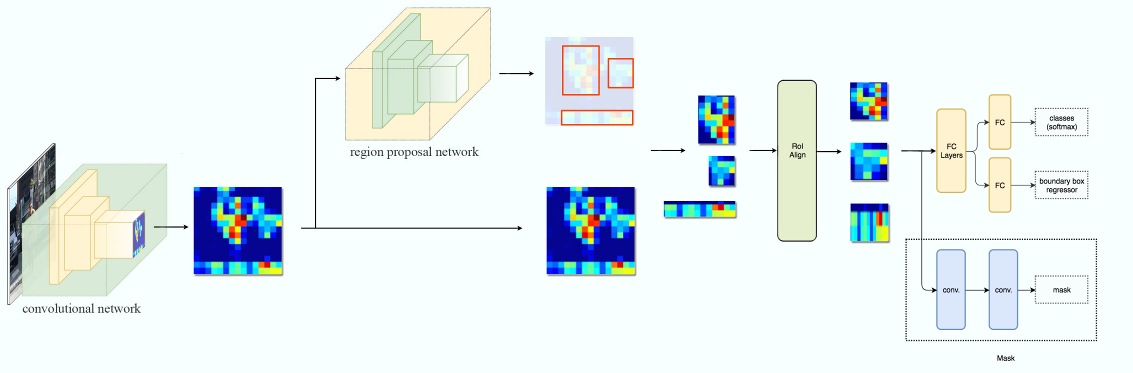
Region Proposal methods work by cropping & warping a set of sub-regions from the image, also known as Regions of Interest (RoI) then perform object detection.
E.g. RCNN:




[1] Courtesy:https://medium.com/@jonathan_hui/what-do-we-learn-from-region-based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377

# Recap: Region Proposal Based Object Detection

RCNN was the first, then came Fast RCNN, followed by Faster RCNN, and current state of the art Mask RCNN.

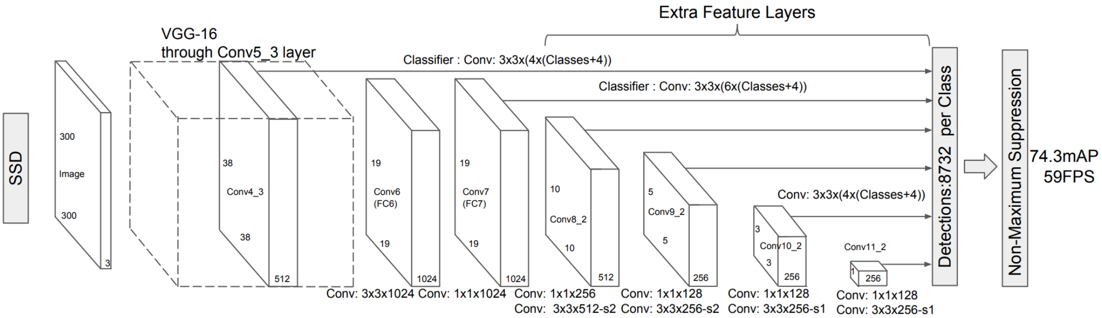



**Very complicated!**

[1] Courtesy:https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272

# Recap: Sliding Window based object detetion

No region proposal, directly extracting features from the backbone network and perform classification & box regression tasks.
E.g. Single Shot multibox Detector (SSD)



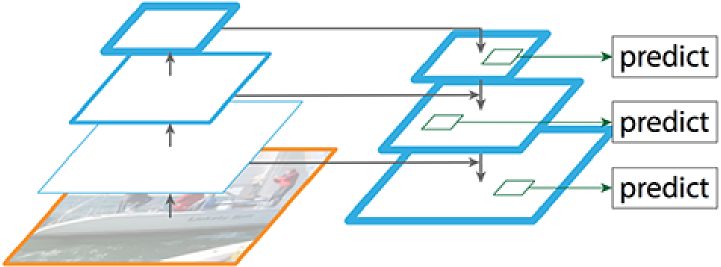

[1] Courtesy:https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11

# Recap: Sliding Window vs Region Proposal

- Sliding window methods require less engineering
- Sliding window methods are usually faster
- Region proposal based methods currently have higher accuracy

# Recap: Feature Pyramid Network

- Feature Pyramid Network (FPN) is a useful backbone feature extraction network
- Information flows upwards and downwards
- Ability to extract feature at all levels of the feature




[1] Courtesy: https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

# TensorMask Overview

- Sliding window based architecture similar to SSD & RetinaNet
- Structured 4-D tensor
- Dense Pixel Labelling
- Detection & Mask representations are intrinsic to the network's features
- Instance segmentation is independent of object detection

# The 4D Feature tensor

Features are represented by a structured 4D-tensor, no channels

$$(V, U, H, W)$$

$(H, W)$ representing the object position
$(V, U)$ representing the relative mask position

# Unit of Length

For each axis of tensor, there is a **unit of length** denoted by $\sigma$. The unit corresponds to the ratio the raw image. For example,
E.g. $\sigma_{HW} = 2$ means a single pixel in $(H, W)$ tensor corresponds to two pixels in image.
Similarly, $\sigma_{VW} = 2$ means a single pixel in $(V, W)$ tensor maps to two pixels in image mask.
The **ratio of units** is denoted as

$$\alpha = \frac{\sigma_{VU}}{\sigma_{HW}}$$

($\alpha$ must be a positive integer)

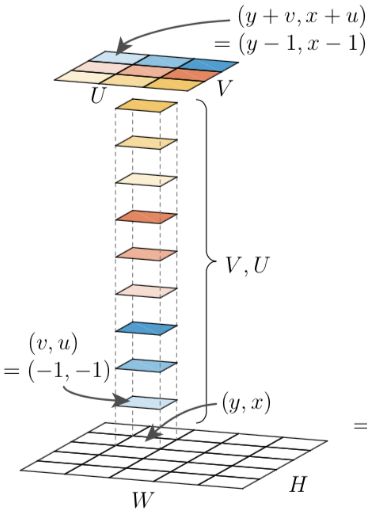
# Two representations of Tensor

There are two ways to represent the tensor. Both are utilized by TensorMask.

- Natural Representation
- Aligned Representation

# Natural Representation

For a 4D tensor of shape $(V, U, H, W)$, a single element at coordinate $(v, u, h, w)$ represents the mask value at $(y + \alpha v, x + \alpha u)$.
Where $(v, u, h, w) \in [-\frac{V}{2}, \frac{V}{2}) \times [-\frac{U}{2}, \frac{U}{2}) \times [0, H) \times [0, W)$



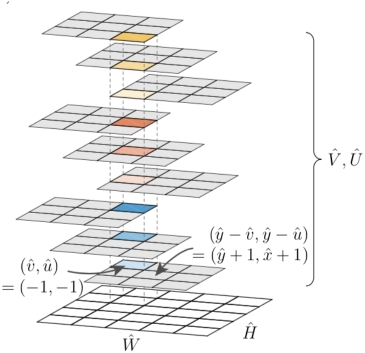

$=$

# Natural Representation

- A 2-d mask $(V, U)$ at a point $(x, y)$ represents a mask for neighbouring pixels of size $(V, U)$ centered at that point.
- Representation for the output layer

# Aligned Representation

Similar to the natural representation, the 4D tensor in aligned representation is denoted as $(\hat{V}, \hat{U}, \hat{H}, \hat{W})$, for a coordinate $(\hat{v}, \hat{u}, \hat{y}, \hat{x})$ it represents the mask at $(\hat{y} - \alpha\hat{v}, \hat{x} - \alpha\hat{u})$



# Aligned Representation

- Preserving the pixel-to-pixel alignment
- A vector of size $(UV)$ represents all the masks that affect this pixel
- Can be used in intermediate layers
- Computationally simpler, easier to apply convolution filters

# Transformations between Aligned and Natural representation

Natural and Aligned representations have identical shape, and it's easily to transform from one representation to the other.

- **align2nat**: $\mathcal{F}(v, u, y, x) = \hat{\mathcal{F}}(u, v, y + \alpha v, x + \alpha u)$
- **nat2align**: $\hat{\mathcal{F}}(\hat{v}, \hat{u}, \hat{y}, \hat{x}) = \mathcal{F}(\hat{u}, \hat{v}, \hat{y} + \alpha\hat{v}, \hat{x} + \alpha\hat{u})$

# Tensor Bipyramid

Like many popular modern networks, TensorMask also contains a pyramid structure. It has a Bipyramid structure defined by:

$$(2^k V, 2^k U, \frac{1}{2^k} H, \frac{1}{2^k} W)$$

where $k \geq 0$ indexes scale (layer)

- $(H, W)$ decreases by layer, geometrically
- $(V, U)$ increases by layer, inversely proportional to $(H, W)$
  *$\lambda$ is a general scaling factor for tensor, when using bipyramid, $\lambda = 2^k$

# Tensor Bipyramid

Tensor bipyramid created with two operations

*up_align2nat*

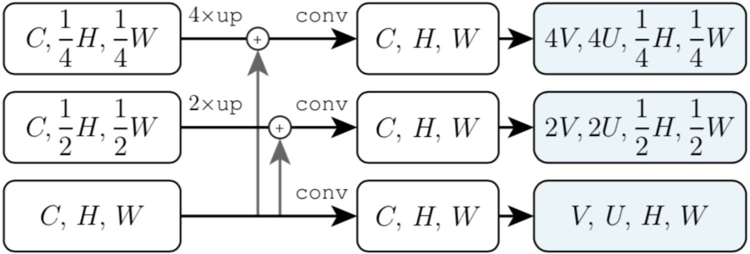
$$\overset{\color{red}{\hat{\sigma}_{VU}=\lambda s}\quad \hat{\sigma}_{HW}=s}{\boxed{\hat{V},\hat{U},\hat{H},\hat{W}}} \xrightarrow[\texttt{up\_bilinear}]{} \overset{\hat{\sigma}_{VU}=s\quad \hat{\sigma}_{HW}=s}{\boxed{\lambda\hat{V},\lambda\hat{U},\hat{H},\hat{W}}} \xrightarrow[\texttt{align2nat}]{} \overset{\sigma_{VU}=s\quad \sigma_{HW}=s}{\boxed{\lambda V,\lambda U,H,W}}$$

*swap_align2nat* (contains *up_align2nat*)

$$\overset{\color{red}{\hat{\sigma}_{VU}=\lambda s}\quad \hat{\sigma}_{HW}=s}{\boxed{\hat{V},\hat{U},\hat{H},\hat{W}}} \xrightarrow[\texttt{up\_align2nat}]{} \overset{\sigma_{VU}=s\quad \sigma_{HW}=s}{\boxed{\lambda V,\lambda U,H,W}} \xrightarrow[\texttt{subsample}]{} \overset{\sigma_{VU}=s\quad \color{red}{\sigma_{HW}=\lambda s}}{\boxed{\lambda V,\lambda U,\frac{1}{\lambda}H,\frac{1}{\lambda}W}}$$

# Adaptation to FPN

The TensorMask's bipyramid structure utilize feature pyramid from FPN, simplifying the contrustion process.



# TensorMask Architecture

- FPN backbone
- Output is in natural representation
- intermediate layers can use any representation.
- Three heads:
  - Object classification
  - Bounding box regression
  - Instance segmentation

# Training - Loss

- **Bounding box regressor** L1-loss
- **Classification** Focal loss
- **Mask** Per pixel binary cross entropy
  - Individual loss for each window, averaged across all pixels in that window
  - Negative windows (no classification) do not contribute to the mask loss

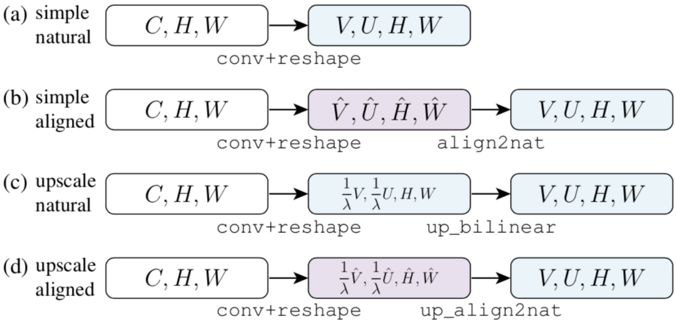
# Training - Label assignment

Since there are far more bounding boxes than ground truth, a mechanism is needed to assign the appropriate label to the corresponding bounding box:

- **Containment** The window fully contains mask $m$ and reasonably tight
- **Centrality** Bounding box must be centered within a unit $(\sigma_{VU}$ of window center
- **Uniqueness** Only one mask $m$ can the above two conditions

# Experiments

- Simple Heads
- Upscaling Head
- Interpolation Head
- Tensor Bipyramid
- Multiple Window Sizes



## Experiments - Simple heads

| head | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| natural | 28.5 | 52.2 | 28.6 | 14.4 | 30.2 | 40.1 |
| aligned | 28.9 | 52.5 | 29.3 | 14.6 | 30.8 | 40.7 |

Table 1. **Simple heads**: natural *vs.* aligned (Fig. 6a *vs.* 6b) with $V \times U = 15 \times 15$ perform comparably if upscaling is not used.

Both natural and aligned representations perform similarly

# Experiments - Upscaling Heads

| head | $\lambda$ | AP | $AP_{50}$ | $AP_{75}$ | $\Delta$ aligned - natural | | |
|---|---|---|---|---|---|---|---|
| natural | 1.5 | 28.0 | 51.7 | 27.8 | *+0.9* | *+0.7* | *+1.5* |
| aligned | | **28.9** | **52.4** | **29.3** | | | |
| natural | 3 | 24.7 | 48.4 | 22.7 | *+4.1* | *+3.9* | *+6.4* |
| aligned | | **28.8** | **52.3** | **29.1** | | | |
| natural | 5 | 19.2 | 42.1 | 15.6 | *+9.2* | *+9.7* | *+13.0* |
| aligned | | **28.4** | **51.8** | **28.6** | | | |

(a) **Upscaling heads**: natural *vs.* aligned heads (Fig. 6c *vs.* 6d). The $V\times U$=15×15 output is upscaled by $\lambda\times$: `conv+reshape` uses $\frac{1}{\lambda^2}VU$ output channels as input. *The aligned representation has a large gain over its natural counterpart when $\lambda$ is large.*


Aligned representation gives significant improve (+48%) when scaling factor $\lambda = 5$

# Experiments - Upscaling: Bilinear vs Nearest Neighbour

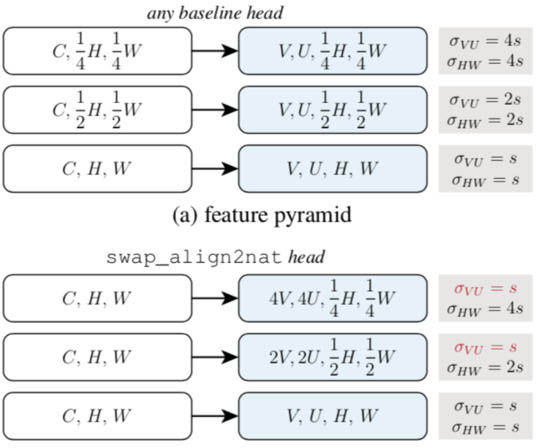
| head | $\lambda$ | AP | AP$_{50}$ | AP$_{75}$ | $\Delta$ bilinear - nearest | | |
|---|---|---|---|---|---|---|---|
| nearest | 1.5 | 28.6 | 52.1 | 29.0 | *+0.3* | *+0.3* | *+0.3* |
| bilinear | | **28.9** | **52.4** | **29.3** | | | |
| nearest | 3 | 27.8 | 51.0 | 28.0 | *+1.0* | *+1.3* | *+1.1* |
| bilinear | | **28.8** | **52.3** | **29.1** | | | |
| nearest | 5 | 25.3 | 47.6 | 25.0 | *+3.1* | *+4.2* | *+3.6* |
| bilinear | | **28.4** | **51.8** | **28.6** | | | |

Bilinear upscaling appears to perform much better than nearest neighbour.

# Experiments - Tensor Bipyramid

| head | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| feature pyramid, best | 28.9 | 52.5 | 29.3 | 14.6 | 30.8 | 40.7 |
| tensor bipyramid | **34.0** | **55.2** | **35.8** | **15.3** | **36.3** | **48.4** |
| $\Delta$ | *+5.1* | *+2.7* | *+6.5* | *+0.7* | *+5.5* | *+7.7* |

Big jump in performance when the FPN + Bipyramid is used, upscaling is key to make tensor bipyramid possible.




(a) feature pyramid

(b) tensor bipyramid

# Experiments - Window Sizes

| $V \times U$ | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| $15\times15$ | 34.0 | 55.2 | 35.8 | 15.3 | 36.3 | 48.4 |
| $15\times15, 11\times11$ | **35.2** | **56.4** | **37.0** | **17.4** | **37.4** | **49.7** |
| $\Delta$ | *+1.2* | *+1.2* | *+1.2* | *+2.1* | *+1.1* | *+1.3* |

Analogous to Anchors boxes in RCNN methods, TensorMask can have multiple window sizes. Having multiple window slightly boosts performance. (Though potentially big impact on speed)

# Experiments - COCO Average Precision

TensorMask achieves comparable results to MaskRCNN without much optimization.

| method | backbone | aug | epochs | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|---|
| Mask R-CNN [13] | R-50-FPN | | 24 | 34.9 | 57.2 | 36.9 | 15.4 | 36.6 | 50.8 |
| Mask R-CNN, *ours* | R-50-FPN | | 24 | 34.9 | 56.8 | 36.8 | 15.1 | 36.7 | 50.6 |
| Mask R-CNN, *ours* | R-50-FPN | ✓ | 72 | **36.8** | **59.2** | **39.3** | **17.1** | **38.7** | **52.1** |
| TensorMask | R-50-FPN | ✓ | 72 | 35.4 | 57.2 | 37.3 | 16.3 | 36.8 | 49.3 |
| Mask R-CNN, *ours* | R-101-FPN | ✓ | 72 | **38.3** | **61.2** | **40.8** | **18.2** | **40.6** | **54.1** |
| TensorMask | R-101-FPN | ✓ | 72 | 37.1 | 59.3 | 39.4 | 17.4 | 39.1 | 51.6 |

However, the speed is much slower. 0.38s/im (TensorMask) vs 0.09s/im (Mask-RCNN)

# Pros and Cons of TensorMask

- Simpler structure than RCNN based methods
- Dense pixel labelling/representation
- $(V, U)$ acts as an implicit anchor box as well as mask
- Classless mask windows
- Masks are independent from boxes
- Slower than Mask RCNN (But lots of room for improvement!)
- More rigid structure, pyramid shape must be $\frac{1}{2^k}$
- Rigid number of bounding boxes, and possibly much much larger

# Implementation

An official implementation of TensorMask is scheduled to be released within a few weeks along with Detectron2
https://github.com/facebookresearch/detectron2

# Extra: Focal Loss

$$FL(p_t) = -\alpha(1 - p_t)^\gamma \log(p_t)$$

where $\alpha$ and $\gamma$ are hyperparameters and $p_t$ is the prediction likelihood.

Thanks!