TensorMask: A Foundation for Dense Object Segmentation

Xinlei Chen, Ross Girshick, 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:
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!

¹Courtesy: https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272
Recap: Sliding Window based object detection

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

\[\text{Diagram of SSD with feature extraction and classification layers.}\]

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-45b227b8100c
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
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
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 $(v, u, h, w) \in \left[ -\frac{V}{2}, \frac{V}{2} \right] \times \left[ -\frac{U}{2}, \frac{U}{2} \right] \times [0, H) \times [0, W)$
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
Natural and Aligned representations have identical shape, and it’s easily to transform from one representation to the other.

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

- **nat2align**: \( \hat{F}(\hat{v}, \hat{u}, \hat{y}, \hat{x}) = F(\hat{u}, \hat{v}, \hat{y} + \alpha \hat{v}, \hat{x} + \alpha \hat{u}) \)
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
\textit{up\_align2nat}

\[
\hat{V}, \hat{U}, \hat{H}, \hat{W} \xrightarrow{\text{up\_bilinear}} \lambda \hat{V}, \lambda \hat{U}, \hat{H}, \hat{W} \xrightarrow{\text{align2nat}} \lambda V, \lambda U, H, W
\]

\textit{swap\_align2nat} (contains \textit{up\_align2nat})

\[
\hat{V}, \hat{U}, \hat{H}, \hat{W} \xrightarrow{\text{up\_align2nat}} \lambda V, \lambda U, H, W \xrightarrow{\text{subsample}} \lambda V, \lambda U, \frac{1}{\lambda} H, \frac{1}{\lambda} W
\]
Adaptation to FPN

The TensorMask’s bipyramid structure utilize feature pyramid from FPN, simplifying the construction 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
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 (\( \sigma_{\text{VU}} \) 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

<table>
<thead>
<tr>
<th>head</th>
<th>AP</th>
<th>AP\textsubscript{50}</th>
<th>AP\textsubscript{75}</th>
<th>AP\textsubscript{S}</th>
<th>AP\textsubscript{M}</th>
<th>AP\textsubscript{L}</th>
</tr>
</thead>
<tbody>
<tr>
<td>natural</td>
<td>28.5</td>
<td>52.2</td>
<td>28.6</td>
<td>14.4</td>
<td>30.2</td>
<td>40.1</td>
</tr>
<tr>
<td>aligned</td>
<td>28.9</td>
<td>52.5</td>
<td>29.3</td>
<td>14.6</td>
<td>30.8</td>
<td>40.7</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>head</th>
<th>λ</th>
<th>AP</th>
<th>AP₅₀</th>
<th>AP₇₅</th>
<th>Δ aligned - natural</th>
</tr>
</thead>
<tbody>
<tr>
<td>natural aligned</td>
<td>1.5</td>
<td>28.0</td>
<td>51.7</td>
<td>27.8</td>
<td>+0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28.9</td>
<td>52.4</td>
<td>29.3</td>
<td>+0.7 +1.5</td>
</tr>
<tr>
<td>natural aligned</td>
<td>3</td>
<td>24.7</td>
<td>48.4</td>
<td>22.7</td>
<td>+4.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28.8</td>
<td>52.3</td>
<td>29.1</td>
<td>+3.9 +6.4</td>
</tr>
<tr>
<td>natural aligned</td>
<td>5</td>
<td>19.2</td>
<td>42.1</td>
<td>15.6</td>
<td>+9.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28.4</td>
<td>51.8</td>
<td>28.6</td>
<td>+9.7 +13.0</td>
</tr>
</tbody>
</table>

(a) Upscaling heads: natural vs. aligned heads (Fig. 6c vs. 6d). The $V \times U = 15 \times 15$ output is upscaled by $\lambda \times \text{conv} + \text{reshape}$ 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

Bilinear upscaling appears to perform much better than nearest neighbour.

<table>
<thead>
<tr>
<th>head</th>
<th>λ</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>Δ bilinear - nearest</th>
</tr>
</thead>
<tbody>
<tr>
<td>nearest</td>
<td>1.5</td>
<td>28.6</td>
<td>52.1</td>
<td>29.0</td>
<td>+0.3</td>
</tr>
<tr>
<td>bilinear</td>
<td></td>
<td>28.9</td>
<td>52.4</td>
<td>29.3</td>
<td>+0.3</td>
</tr>
<tr>
<td>nearest</td>
<td>3</td>
<td>27.8</td>
<td>51.0</td>
<td>28.0</td>
<td>+1.0</td>
</tr>
<tr>
<td>bilinear</td>
<td></td>
<td>28.8</td>
<td>52.3</td>
<td>29.1</td>
<td>+1.3</td>
</tr>
<tr>
<td>nearest</td>
<td>5</td>
<td>25.3</td>
<td>47.6</td>
<td>25.0</td>
<td>+3.1</td>
</tr>
<tr>
<td>bilinear</td>
<td></td>
<td>28.4</td>
<td>51.8</td>
<td>28.6</td>
<td>+4.2</td>
</tr>
</tbody>
</table>
Big jump in performance when the FPN + Bipyramid is used, upscaling is key to make tensor bipyramid possible.
Experiments - Window Sizes

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)

<table>
<thead>
<tr>
<th>$V \times U$</th>
<th>AP</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_{S}$</th>
<th>AP$_{M}$</th>
<th>AP$_{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>15$\times$15</td>
<td>34.0</td>
<td>55.2</td>
<td>35.8</td>
<td>15.3</td>
<td>36.3</td>
<td>48.4</td>
</tr>
<tr>
<td>15$\times$15, 11$\times$11</td>
<td><strong>35.2</strong></td>
<td><strong>56.4</strong></td>
<td><strong>37.0</strong></td>
<td><strong>17.4</strong></td>
<td><strong>37.4</strong></td>
<td><strong>49.7</strong></td>
</tr>
<tr>
<td>$\Delta$</td>
<td>+1.2</td>
<td>+1.2</td>
<td>+1.2</td>
<td>+2.1</td>
<td>+1.1</td>
<td>+1.3</td>
</tr>
</tbody>
</table>
TensorMask achieves comparable results to MaskRCNN without much optimization.

<table>
<thead>
<tr>
<th>method</th>
<th>backbone</th>
<th>aug</th>
<th>epochs</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>APs</th>
<th>AMP</th>
<th>APL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask R-CNN [13]</td>
<td>R-50-FPN</td>
<td></td>
<td>24</td>
<td>34.9</td>
<td>57.2</td>
<td>36.9</td>
<td>15.4</td>
<td>36.6</td>
<td>50.8</td>
</tr>
<tr>
<td>Mask R-CNN, ours</td>
<td>R-50-FPN</td>
<td></td>
<td>24</td>
<td>34.9</td>
<td>56.8</td>
<td>36.8</td>
<td>15.1</td>
<td>36.7</td>
<td>50.6</td>
</tr>
<tr>
<td>Mask R-CNN, ours</td>
<td>R-50-FPN</td>
<td>✓</td>
<td>72</td>
<td>36.8</td>
<td>59.2</td>
<td>39.3</td>
<td>17.1</td>
<td>38.7</td>
<td>52.1</td>
</tr>
<tr>
<td>TensorMask</td>
<td>R-50-FPN</td>
<td>✓</td>
<td>72</td>
<td>35.4</td>
<td>57.2</td>
<td>37.3</td>
<td>16.3</td>
<td>36.8</td>
<td>49.3</td>
</tr>
<tr>
<td>Mask R-CNN, ours</td>
<td>R-101-FPN</td>
<td>✓</td>
<td>72</td>
<td>38.3</td>
<td>61.2</td>
<td>40.8</td>
<td>18.2</td>
<td>40.6</td>
<td>54.1</td>
</tr>
<tr>
<td>TensorMask</td>
<td>R-101-FPN</td>
<td>✓</td>
<td>72</td>
<td>37.1</td>
<td>59.3</td>
<td>39.4</td>
<td>17.4</td>
<td>39.1</td>
<td>51.6</td>
</tr>
</tbody>
</table>

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!