输入

img → VGG16 网络

原始图片(放缩后) img 1,3,600,901

Bbox bbox 1,2,4

标签 label 1,2

放缩 scale 1,

img_size=(600,901)

VGG16 网络：

13个conv层：3x3, pad=1, s=1. 则 $H_{new} = \frac{H_{old}-3+2\times1}{1}+1$

13个relu层 经过conv层，长宽不变

4个pooling层：2x2, pad=0, s=2. 则 $H_{new} = \frac{H_{old}-2}{2}+1 = \frac{H_{old}}{2}$

特征图 features 1,512,37,56

512,512,3,1,1

Conv2D

relu

新特征图 1,512,37,56

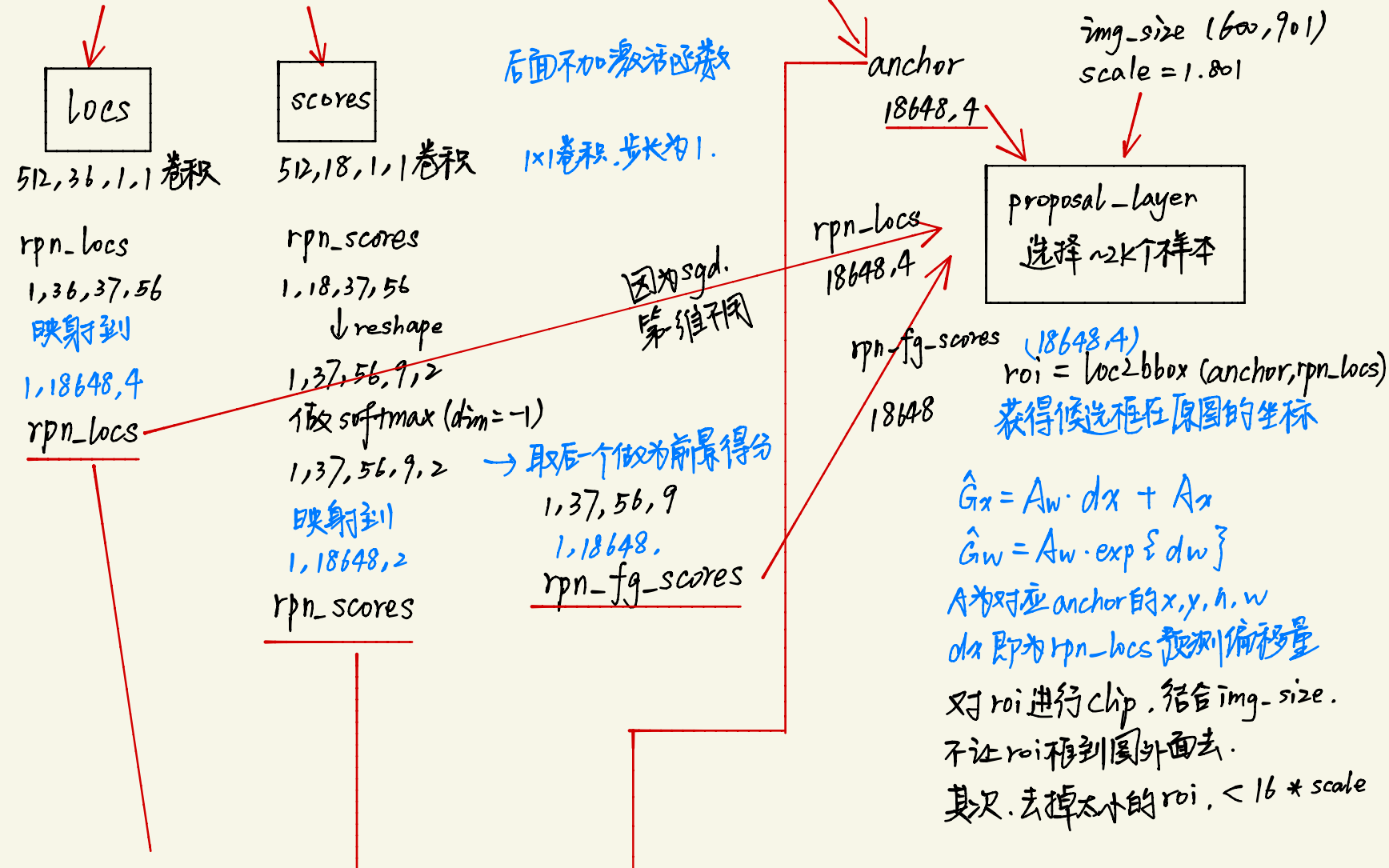
_enumerate_shifted_anchor(anchor_base, feat_stride, f_h, f_w)

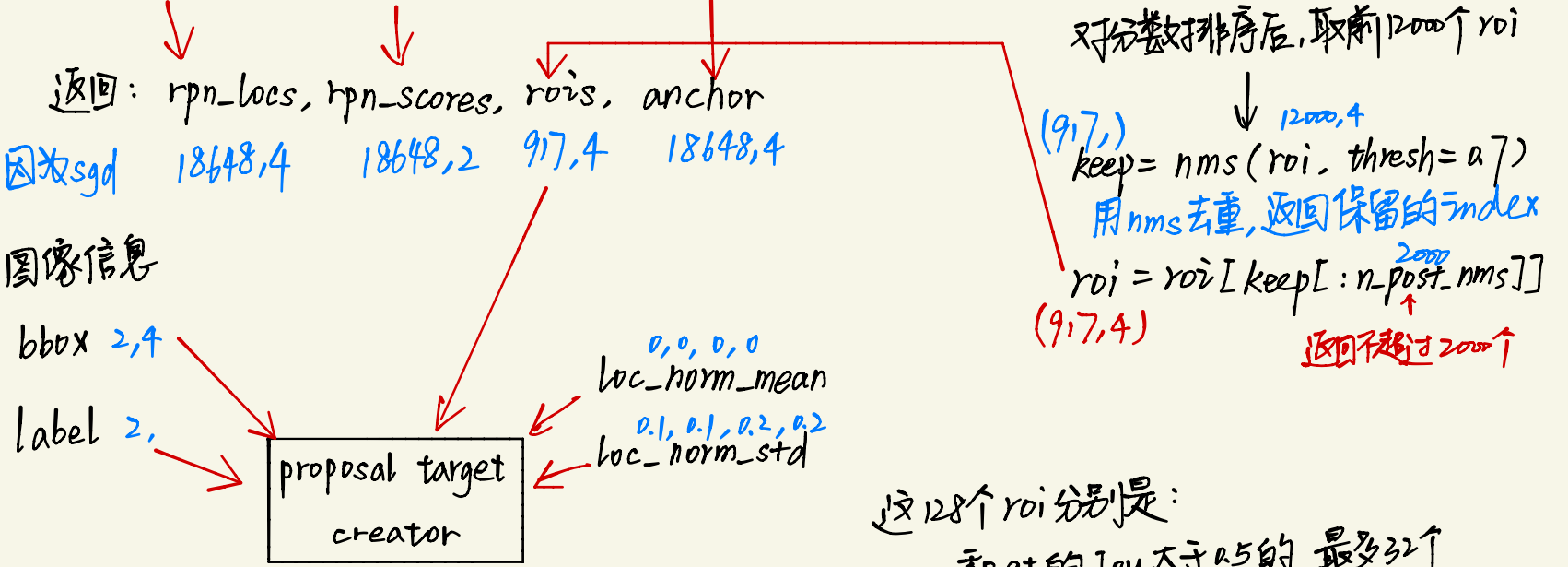
16 37 56

生成 anchor 18648,4

= 37x56x9

原图上每16个像素有个anchor中心


locs
512,36,1,1 卷积
rpn_locs
1,36,37,56
映射到
1,18648,4
rpn_locs
scores
512,18,1,1 卷积
rpn_scores
1,18,37,56
↓reshape
1,37,56,9,2
做 softmax (dim=-1)
1,37,56,9,2
映射到
1,18648,2
rpn_scores
后面不加激活函数
1×1卷积，步长为1.
→ 取后一个做为前景得分
1,37,56,9
1,18648,
rpn_fg_scores
因为sgd.
第一维不同
anchor
18648,4
img_size (600,901)
scale = 1.801
rpn_locs
18648,4
proposal_layer
选择~2K个样本
rpn_fg_scores
18648
(18648,4)
roi = loc2bbox (anchor,rpn_locs)
获得候选框在原图的坐标
$\hat{G}_x = A_w \cdot d_x + A_x$
$\hat{G}_w = A_w \cdot \exp\{d_w\}$
A为对应anchor的x,y,h,w
dx 即为 rpn_locs 预测偏移量
对 roi 进行 clip，结合 img_size.
不让 roi 框到图外面去.
其次，去掉太小的 roi，< 16 * scale



从前面917个roi中，选择128个进行训练.

这128个roi分别是：

和gt的Iou大于0.5的 最多32个

和gt的Iou在0.1~0.5的，补齐128个

917 = 917+2. 把真实框也算成roi

具体操作：

```
gt_roi_label = gt_label[keep_idx]  正样本label
gt_roi_label[32:] = 0.  负样本label为0.
sample_roi = roi[keep_idx]
```

(128,) (128,4)

```
iou = bbox_iou(roi, bbox)
```

(917,2) 每个roi和bbox们的iou值

```
gt_idx = iou.argmax(axis=1)
```

(917,) 每个roi和图中第n个bbox的iou大

```
max_iou = iou.max(axis=1)
```

(917,) 每个roi和图中任一bbox的iou最大值.

gt_roi_loc = bbox2loc(sample_roi, bbox[gt_idx[keep_idx]])

128,4

选取出128个roi的真实偏移量

$G_x = \hat{G}_x + \hat{G}_w \cdot dx$

$G_w = \hat{G}_w \cdot \exp\{dw\}$ 求出 dx, dw...

919, gt_label = label[gt_idx] + 1

0-19换成1-20. 0作为背景

pos_idx = max_iou > 0.5

从中抽32个. 如果iou>0.5的更多

neg_idx = max_iou < 0.5 & max_iou > 0.1

取满128个

keep_idx = np.append(pos_idx, neg_idx)

128,4 128,4 128,

sample_roi, gt_roi_loc, gt_roi_label

作roipooling.

前面的features

1,512,37,56

roi_head

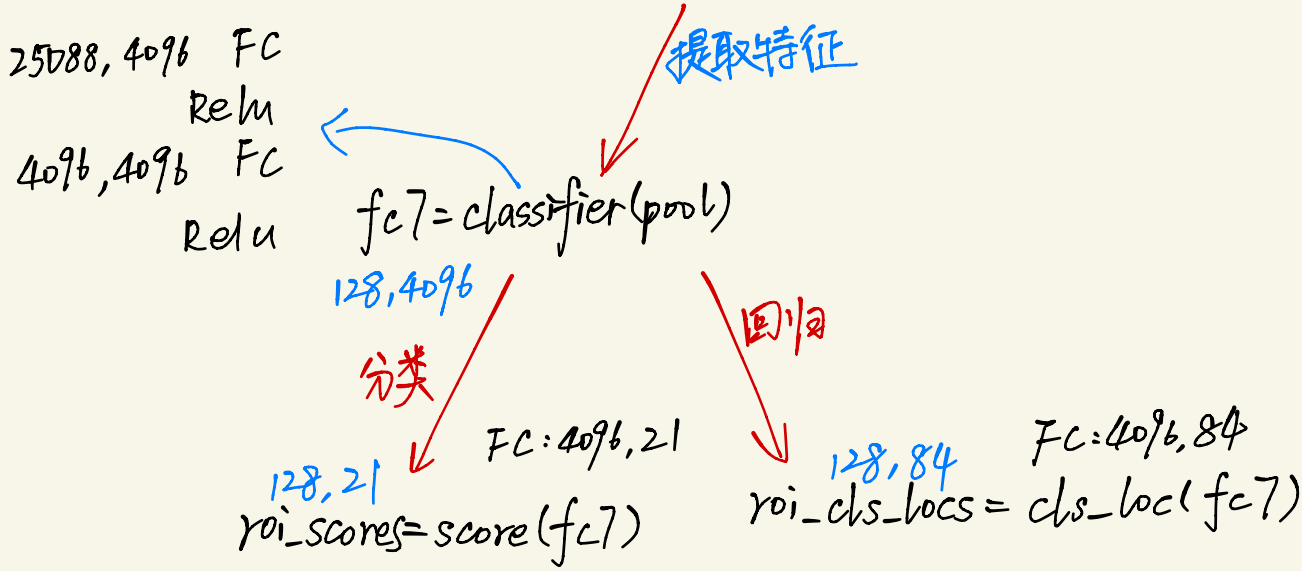
pool = roi(features, rois)

128,512,7,7

reshape成

128, 512x7x7 → 128,25088

pool



anchor target creator 为所有的anchor分配gt

输入：bbox 2,4
img_size (600,901)
anchor 18648,4

输出：gt_rpn_label 18648,

全部anchor 18648

↓

不越界的anchor 6928

找出这6928个anchor和哪个bbox的iou最大

loc = bbox2loc(anchor_in, bbox[对应大的])

6928,4 6928,4

gt_rpn_loc (18648,4)

这6928个anchor转换到bbox的dx.dw..

$G_x = A_x + A_w\,dx$ 还要 loc − mean / std

$G_w = A_w \cdot \exp(dw)$

然后把刚才不全在图中的anchor补回来

label为-1, loc全为0.

↓不参与损失计算.

计算rpn Loss:

rpn_loc_loss = loc_loss (rpn_locs, gt_rpn_loc, gt_rpn_label, rpn_sigma)
(18648,4) (18648,4) (18648) (3)

rpn_cls_loss = cross_entropy (rpn_score, gt_rpn_label, ignore_idx=-1)
(18648,2) (18648)

计算roi Loss:

roi_cls_loc (128,84) 变形为 128,21,4 · 用 gt_roi_label 选出那一类的 loc → roi_loc (128,4)

则 roi_loc_loss = loc_loss (roi_loc, gt_roi_loc, gt_roi_label, roi_sigma)
(128,4) (128,4) (128)

roi_cls_loss = cross_entropy(roi_score, gt_roi_label)
(128,21) (128,)
↓负样本label为0.

在evaluate模式时：(返回更多roi)

计算得到 roi_cls_loc, roi_scores, rois
(300,84) (300,21) (300,4)

对 roi_cls_loc * std + mean. 再转换到 300,21,4

rois 转换成 300,21,4

cls_box = loc2bbox(rois_, roi_cls_loc_)
(300×21,4) (300×21,4)

↓reshape

cls_bbox (300,84). 对坐标 clip. 不能超出图片边界.

对 roi_scores 作 softmax → prob (300,21)

(n,4) (n,) (n,)

bbox, label, score = suppress(cls_bbox, prob) 对每类作 nms (除了背景类) 最终结果！