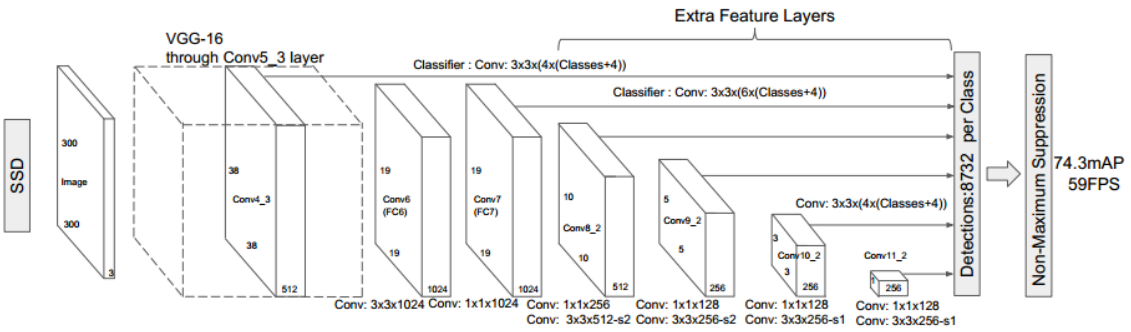
## Related works:

Object detection methods can be separated into two groups:

1. **Region proposal–based** methods:
   Some of the famous methods are R-CNN [1], Fast R-CNN [2], Faster R-CNN [3], and Mask R-CNN [4].
2. **Regression**-based methods:
   Some of the famous methods are OverFeat [5], YOLO [6] [7] [8] [9] [10], SSD [11].

- **R-CNN** [1]: consists of three modules:
  1. Region proposal.
     Despite the fact that R-CNN is agnostic to the region proposal method, in the paper they chose to use Selective search.
  2. CNN for feature extraction.
     4096-dimensional feature vector from each region.
  3. Class-specific SVMs.
- **Fast R-CNN** [2]:
  Training is single-stage, using multi-task loss, Whereas R-CNN use multi-stage training.
  In addition, it is faster and has better detection quality and more space-efficient.
- **Faster R-CNN** [3]:
  Consists of two modules:
  1. Deep CNN that proposes regions.
  2. The Fast R-CNN detector.

  Using attention RPN tells Fast R-CNN where to look.
  It uses the idea of anchor boxes.
- **OverFeat** [5]:
  It uses one network for classification, localization, and detection, which learn to predict object boundaries.
  Bounding boxes are then accumulated rather than suppressed.
- **SSD** [11]:
  It uses anchor boxes with different scales and aspect ratio, and also combines predictions from multiple feature maps.



- **Problems that arise in aerial object detection:**

  - **Orientation:**

    Traditional object detection tasks deal with horizontally aligned bounding boxes (HBBs), however, in aerial images objects can appear in arbitrary orientation, which yields HBBS inaccurate. That leads to the creation of datasets annotated using oriented bounding boxes (OBBs).

    In OBBs we need a new representation of the boxes which introduces new problems discussed in the next section.

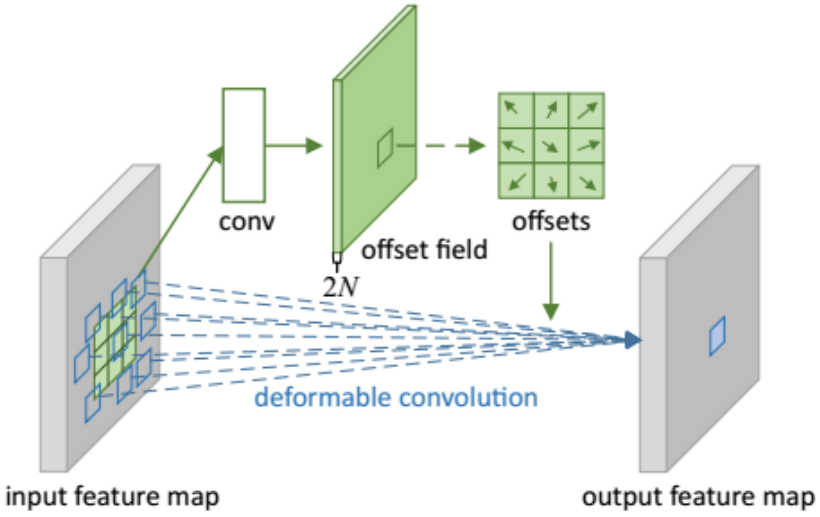
    ☒ One of the solutions is **Deformable ConvNet** [12], which addressed the limitation of CNN in modeling transformations, introducing deformable convolution and deformable RoI pooling.

    The 2D convolution consists of two steps:

    1. Sampling using a regular grid over the input feature map.
    2. Summation of sampled values weighted by ***w***.

    The grid defines the receptive field size and dilation.

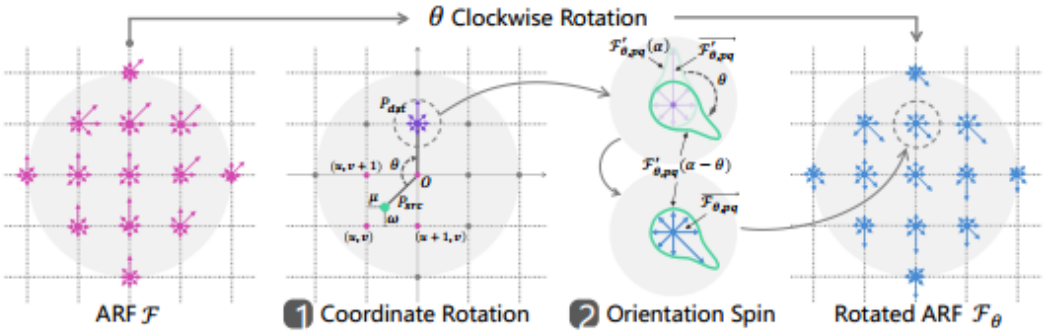
    In deformable convolution, the regular grid is augmented with offsets, which is obtained by applying a convolutional layer over the same input feature map.




    ☒ **Data augmentation (DA)**:

    Despite the effectiveness of DA, the main drawback lies in that learning all the possible transformations of augmented data usually requires more network parameters.

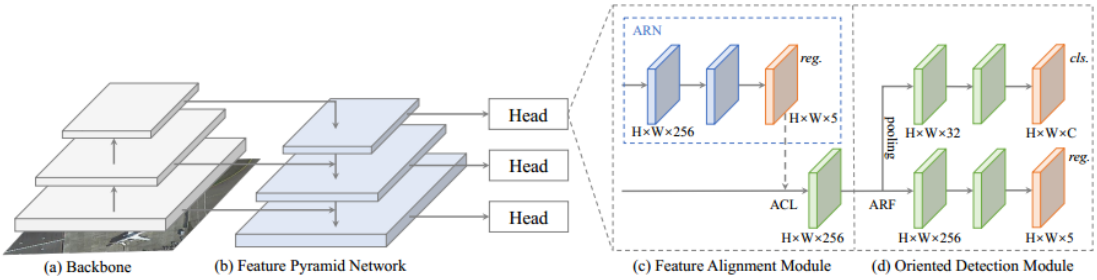
☒ In addition, to solve the problem of limited ability to handle significant local and global rotations in CNNs, **Active Rotation Filters (ARFs)** [13] were proposed.
ARF is a filter that actively rotates during convolution to produce a feature map with multiple orientation channels.
It must be noted that ARFs do not utilize the OBB annotation.

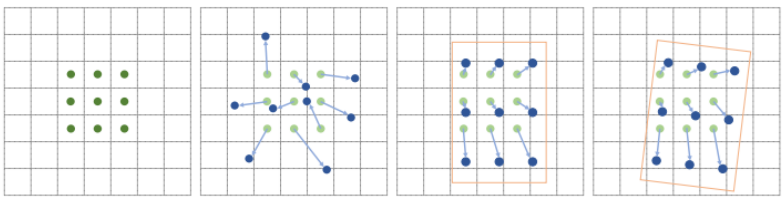



An ARF $\mathcal{F}$ is clockwise rotated by $\theta$ to yield its rotated variant $\mathcal{F}_\theta$ in two steps: coordinate rotation and orientation spin.

☒ Apart from the ability of CNN to capture orientation information, we have the problem of relying on heuristically defined anchors which usually yields methods that suffer from severe misalignment between anchor boxes and axis-aligned convolutional features.
To address this issue [14] proposed a **Single-shot Alignment Network (S$^2$ A-Net)** consisting of two modules:

1. A Feature Alignment Module (FAM) that can generate high-quality anchors with an Anchor Refinement Network and adaptively align the convolutional features according to the anchor boxes with a novel alignment convolution.
2. Oriented Detection Model (ODM) that first adopts ARFs to encode the orientation information and then produce orientation-sensitive and orientation-invariant features to alleviate the inconsistency between classification score and localization accuracy.



❖ **Representation:**

There are two main approaches to represent OBBs:

1. **Regression-based**:

   There are three common ways to represent OBBs using this approach:

   a. $\theta - based$ :

   $$\{(cx,\ cy,\ h,\ w,\ \theta)\}$$

   $cx,\ cy$ are the center coordinate of OBB.

   $h,\ w,\ \theta$ are the height, width, and angle respectively.

   b. $point - based$ :

   $$\{(x_i,\ y_i)|i = 1,\ 2,\ 3,\ 4\}$$

   $(x_i,\ y_i)$ is the $i - th$ vertex.

   c. $h - based$ :

   $$\{(x_1,\ y_1,\ x_2,\ y_2,\ h)\}$$

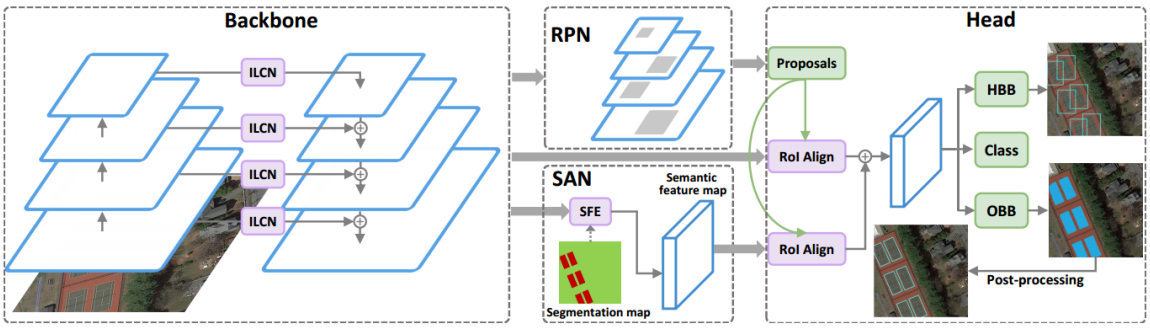
   $(x_1, y_1)$, $(x_2, y_2)$ is the 1st and 2nd vertex.

   $h$ is the height.

   Although these representations ensure the uniqueness in the OBBs definition with some rules, they still allow extreme conditions.
   In these conditions, a tiny change in OBB angle would result in a large change in OBB representation, we denote angle values in these conditions as discontinuity points. For oriented object detectors, similar features extracted by the detector with close positions are supposed to generate similar position representations. However, OBB representations of these similar features would differ greatly near discontinuity points. This would force the detector to learn totally different position representations for similar features. It would impede the detector training process and deteriorate the detector's performance obviously.

2. **Segmentation-based**:

   **Mask-OBB** [15] is an example that handles ambiguity by representing the oriented object as a binary segmentation map which ensures the uniqueness naturally, and the problem of detecting OBB can be treated as a pixel-level classification for each proposal.

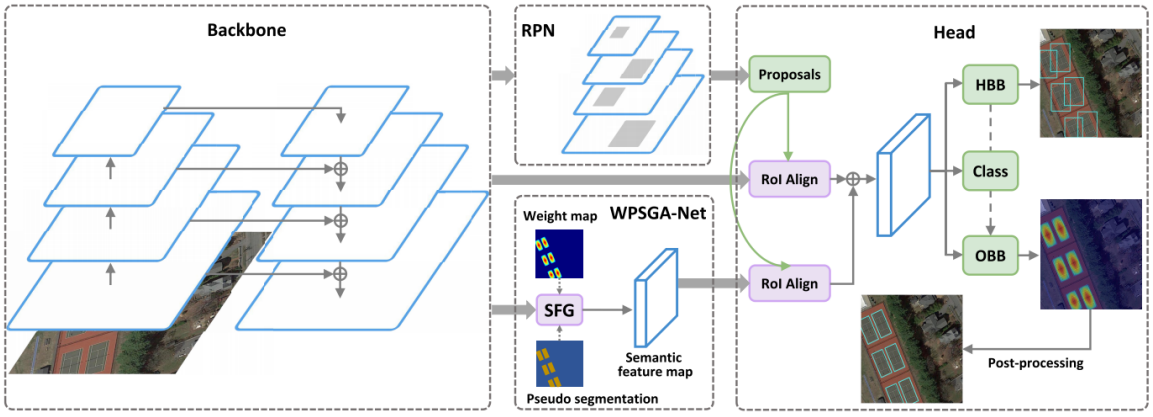


Then the OBBs are generated from the predicted masks by post-processing, and we call this kind of OBB representation “mask-oriented bounding box representation” (Mask OBB). Under this representation, there is no discontinuity point and ambiguity problem.

In the Mask-OBB pixels inside the OBB are labeled as positive and pixels outside are labeled as negative, but there are still lots of background pixels surrounding the real object in the Mask OBB when the real object is not a strict quadrilateral, such as a plane, harbor, and helicopter in the aerial images.

In these objects, foreground pixels are concentrated in the center of the OBB, while background pixels are concentrated on the boundary of the OBB. However, these background pixels might affect the convergence of the segmentation network, resulting in low-quality segmentation results. We call this the ambiguity problem of the background pixels.

To handle the ambiguity problem of the background pixels, [23] propose a **CenterMap OBB** for better representing the oriented objects in the aerial images. Unlike Mask OBB which labels each pixel discretely, the CenterMap OBB encodes OBB with a CenterMap, where the values for the border/center pixels are 0/1 and the values of other pixels gradually decay from center to border.



❖ **Scale:**

In aerial images, the objects appear on different scales, from small cars to harbors and football pitches.

This huge variance in scale results in a problem we cannot ignore.

Specifically, the receptive field increase as we go deeper in the network which is essential in order to see the “big picture” and encode the global context and that gives us the higher-level features which contain more powerful semantic information than shallower layers, but that constitutes a problem for small objects since the network is more likely to focus on the big objects.

In contrast, the spatial information lost its precision as we move deeper.

To summarize, deeper layers give us a much larger receptive field and therefore encode global context and contain semantic information, whereas shallow layers contain spatial information.

In order to solve this problem a wide range of solutions arose.

We will present some of them here.

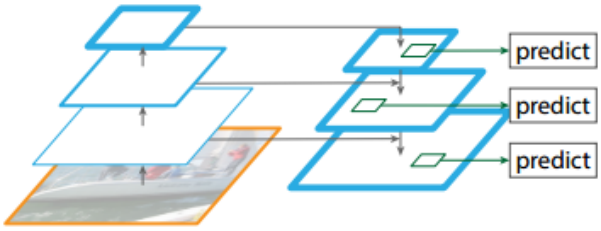
1. **SSD** [11]:

   It tries to solve the problem as follows:

   a. They use a base network they truncated before any classification layer.

   b. They add convolutional feature layers to the end of the base network.
   These layers decrease in size progressively and allow predictions of detections at multiple scales.

2. **Feature Pyramid Network** [16]:




   It includes two pathways:

   a. Bottom-up pathway:
   It is just a feedforward CNN, which computes a features 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 and we say these layers are in the same network stage. For our feature pyramid, we define one pyramid level for each stage. We choose the output of the last layer of each stage as our reference set of feature maps, which we will enrich to create our pyramid. This choice is natural since the deepest layer of each stage should have the strongest features.

b. Top-down pathway:
The top-down pathway hallucinates higher resolution features by upsampling spatially coarser, but 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. The bottom-up feature map is of lower-level semantics, but its activations are more accurately localized as it was subsampled fewer times.

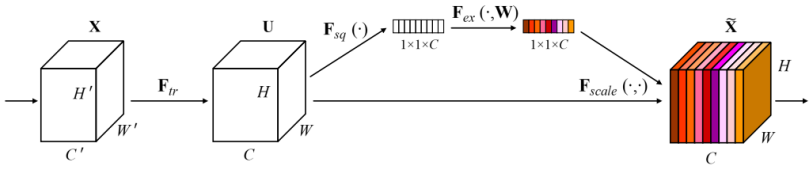
3. **YOLO v3** [8]:
YOLOv3 predicts boxes at 3 different scales. Their system extracts features from those scales using a similar concept to feature pyramid networks.
From our base feature extractor, they add several convolutional layers.
The last of these predict a 3-d tensor encoding bounding box, objectness, and class predictions.
Next, they take the feature map from 2 layers previous and upsample it by 2×. they also take a feature map from earlier in the network and merge it with our upsampled features using concatenation. This method allows us to get more meaningful semantic information from the upsampled features and finer-grained information from the earlier feature map.
Then, they add a few more convolutional layers to process this combined feature map, and eventually predict a similar tensor, although now twice the size.
They perform the same design one more time to predict boxes for the final scale.
Thus, our predictions for the 3$^{rd}$ scale benefit from all the prior computations as well as fine-grained features from early on in the network.

4. **Attention**:
The attention mechanism can be used here also.
Since we will focus on using attention to enhance the detection, we preserve a special section for it.

- ## Attention:

  Attention can be used to guide the network. Specifically, we can use it to tell the network where to look, thus focusing on useful information.
  In addition to “Where” we can use it in the “What” by focusing on the useful properties of objects.
  Attention in CNNs can be applied in different ways. We will list some of them here.

  1. **Squeeze-and-Excitation (SE) Networks** [17]:
     The goal of this paper is to improve the representational power of a network by explicitly modeling the interdependencies between the channels of its convolutional features.
     To achieve this they propose a mechanism that allows the network to perform features recalibration, through which it can learn to use global information to selectively emphasize informative features and suppress less useful ones.




     The SE block can be constructed for any given transformation:

$$F_{tr}: X \rightarrow U,$$

$$X \in R^{H' \times W' \times C'},\ U \in R^{H \times W \times C}$$

     And perform:

     a. Squeeze:
        Each learned filters operates with a local receptive field and consequently each unit of the transformation output $U$ is unable to exploit contextual information outside of this region.
        This is an issue that becomes more severe in the lower layers of the network whose receptive field sizes are small.
        To mitigate this problem, they propose to squeeze global spatial information into a channel descriptor. This is achieved by using global average pooling to generate channel-wise statistics.
        More sophisticated aggregation strategies could be employed here as well.

b. Excitation:
   To make use of the information aggregated in the squeeze operation, they follow it with a second operation which aims to fully capture channel-wise dependencies. To fulfill this objective, the function must meet two criteria:
   i. First, it must be flexible (in particular it must be capable of learning a non-linear interaction between channels).
   ii. Second, it must learn a non-mutually-exclusive relationship since we would like to ensure that multiple channels are allowed to be emphasized as opposed to one-hot activation.

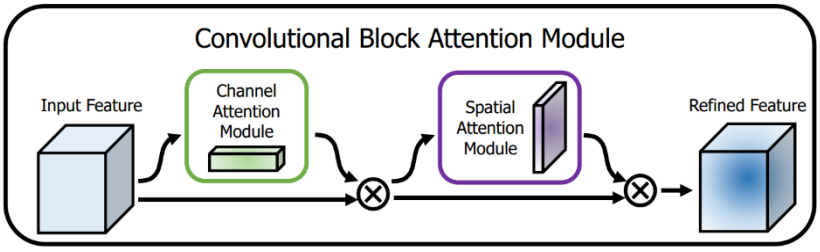
Finally, they rescale it.

2. **Gather-Excite (GE)** [18]:
   GE is similar to SE but instead of taking the global context they introduce a parameter they called 'e' and defined the *gather* operator as

$$\xi_G: R^{H\times W\times C} \rightarrow R^{H'\times W'\times C}$$

$$H' = \lceil \frac{H}{e} \rceil, W' = \lceil \frac{W}{e} \rceil$$

3. **Convolutional Block Attention Module (CBAM)** [19]:




SE used global average-pooled features to compute channel-wise attention. However, [19] show that those are suboptimal features in order to infer fine channel attention, and they suggested using max-pooled features as well.
Also, SE misses spatial attention, which plays an important role in deciding "Where" to focus.

Given an intermediate features map $F \in R^{H\times W\times C}$ as input, CBAM sequentially infers a 1D channel attention map $M_c \in R^{1\times 1\times C}$ and a 2d spatial attention map $M_s \in R^{H\times W\times 1}$.

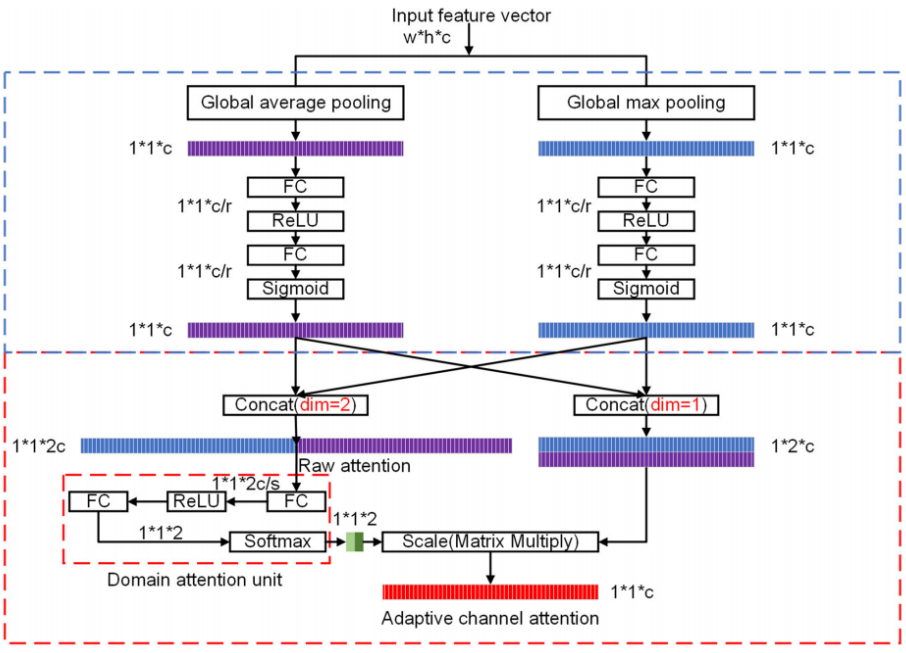
The overall attention process can be summarized as:

$$F' = M_c(F) \otimes F$$

$$F'' = M_s(F') \otimes F'$$

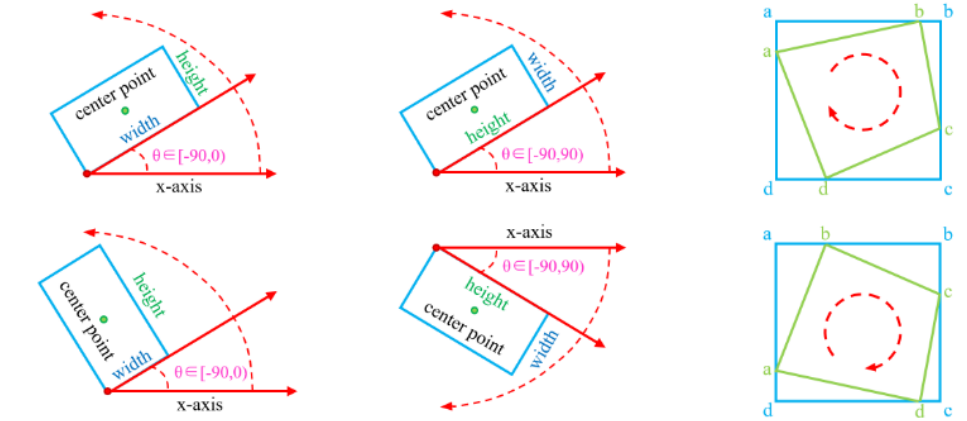
Where ⊗ denotes element-wise multiplication with broadcast.
They generated a channel attention map by combining two versions, one of them uses average pooling, and the other use max pooling.
They argue that max-pooling gathers important clues about distinctive object features.
A spatial attention map was generated using a similar approach.

4. **Adaptive Attention Mechanism** [20]:
   They designed three novel attention units:
   The 1st and the 2nd together as:
   - Adaptive channel-wise attention:
     They have used SE structure and a domain attention unit that acts as a calibrator of the adaptive channel-wise attention units.
     In their use of SE, they consider both average pooling and max pooling.
     The basic intuition behind this is that given an input feature map, global average pooling tends to identify the object extent. On the other hand, the global max point identified by global max pooling indicates that the position contains the feature of an object that can be used for the detection task.
     Global max pooling is more useful when the object is small and when the scale of the feature map shrinks considerably with respect to the spatial dimension during forward propagation.
     They addressed the sub-optimality of weighting both kinds equally.
     They use domain attention to weigh the two kinds simply by using fully connected la



- Spatial attention:
  Different from other methods that had used global max pooling and global average pooling to generate spatial-wise attention, in this paper they generate it through fully convolutional layers in a learning manner.

- **More on the discontinuity problem [21]:**




(a) Five-parameter method with 90° angular range. (b) Five-parameter method with 180° angular range. (c) Ordered quadrilateral representation.

Although the parametric regression-based rotation detection method has achieved competitive performance in different vision tasks, and has been a building block for a number of excellent detection methods, these methods essentially suffer the discontinuous boundaries problem. Boundary discontinuity problems are often caused by angular periodicity in the five-parameter method and corner ordering in the eight-parameter method, but there exist more fundamental root causes regardless of the representation choices of the bounding box.
The boundary discontinuity problem often makes the model's loss value suddenly increase at the boundary situation. Thus methods have to resort to particular and often complex tricks to mitigate this issue. Therefore, these detection methods are often inaccurate in boundary conditions. We describe the boundary problem in three typical categories of regression-based methods according to their different representation forms (the first two refer to the five-parameter methods):

1. 90°-regression-based method, as sketched below, Figure (a). It shows an ideal form of regression (the blue box rotates counterclockwise to the red box), but the loss of this situation is very large due to the periodicity of angular (PoA) and exchangeability of edges (EoE), see the example in the below Figure (a). Therefore, the model has to be regressed in other complex forms (such as the blue box rotating clockwise to the gray box while scaling w and h), increasing the difficulty of regression.

2. $180°$-regression-based method, as illustrated in the below Figure (b). Similarly, this method also has a problem with the sharp increase of loss caused by the PoA at the boundary. The model will eventually choose to rotate the proposal at a large angle clockwise to get the final predicted bounding box.

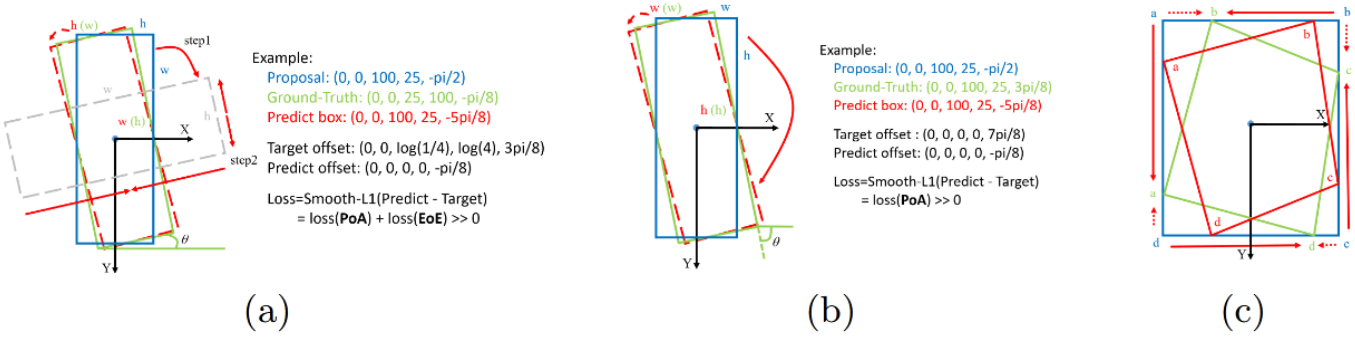
3. Point-based method, as shown below, Figure (c). Through further analysis, the boundary discontinuity problem still exists in the eight-parameter regression method due to the advanced ordering of corner points. Considering the situation of an eight-parameter regression in the boundary case, the ideal regression process should be

$$\{(a \rightarrow b),\ (b \rightarrow c),\ (c \rightarrow d),\ (d \rightarrow a)\}$$

but the actual regression process from the blue reference box to the green ground truth box is

$$\{(a \rightarrow a),\ (b \rightarrow b),\ (c \rightarrow c),\ (d \rightarrow d)\}$$

In fact, this situation also belongs to PoA. By contrast, the actual and ideal regression of the blue to red bounding boxes is consistent.




(a) (b) (c)

## What I will do:

$S^2A - Net$ [14] will be chosen as a starting point. It solves *the problem of orientation* and since it uses **FPN** [16] it also solves *the problem of scale*.

But despite all that, it does not solve *the problem of discontinuity*, so in order to solve that we will use **Dense Label Encoding** [22]. Briefly, it converts the problem of determining the angle from a regression task to a classification task and does that efficiently using a binary-coded representation of the angles.

In order to better guide the network, we will add attention to it, specifically, we will use the previously discussed **Adaptive Attention Mechanism** [20].

# Theoretical notes

## Dense Label Encoding [22]

In the [22], they basically solved the problem of huge computational costs, caused by treating each angle as a single independent class, by representing the angle as a binary number.

To understand that let's take an example:

In case we want to have 256 different angle values, we will need 256 classes in the traditional way of treating the classes, and the labels will be one hot vectors.

But if we use the binary representation, we will decrease that to only 8 classes, but actually, they are not the direct classes we want to have, instead each class here represent a partial amount of the resulting angle value. Also, here we will solve multi-label classification problem.

For example, if we want to represent the 10th (and we will explain why we use the ordinals instead of real angle values) angle, it will have the following representation:

00001010

Therefore we have a huge decrease in space complexity and the number of parameters (and hence better time complexity), the order of the previous decrease is

$O(n) \rightarrow O(log_2(n))$

Where 'n' is the number of classes.

Naturally, we will use Binary Cross Entropy loss with the previous method, but one can observe instantly that:

If we use the previous loss directly, we won't have the concept of angles differences encoded in the loss. Therefore a difference in the first bit will be as same as a difference in the last bit, but the last bit is worth 128 times more than the first one, and that is a huge difference.

To solve the previous problem, they propose a weight function to weigh the loss function as follows:

$$W(\Delta\theta) = \log(|\Delta\theta| + 1) = \log(|\theta_{gt} - \theta_{pred}| + 1)$$
$$\theta_{pred} = \text{Decode}_{dcl}(logits)$$

Obviously, if we use the previous function, we will reintroduce the discontinuity problem as we do not encode the concept of angle periodicity.

Therefore another function is proposed:

$$W_{ADARSW}(\Delta\theta) = |\sin(\alpha(\Delta\theta))| = |\sin(\alpha(\theta_{gt} - \theta_{pred}))|$$

$$\alpha = \begin{cases} 1, & (h_{gt}/w_{gt}) > r \\ 2, & otherwise \end{cases}$$

Here they encoded the concept of angle periodicity by using the 'sin' function which is a periodic function with a period of $2 * \pi$ but when we use the absolute value of it we get a function with a period of $\pi$ which is what we want.

But what the resulting number will represent? this can be handled in various ways, but in the paper the considered angles as offsets from the end of the predefined range of the angles. So we do the following:

1. Convert the output of the network to a binary number by applying the sigmoid function and then rounding its output.
2. Convert from binary to decimal, this number will be in the range [0, 255], since we have 8 bits.
3. multiplying by angle granularity, which represents the length of the range divided by the number of classes of the angle that we want. In our case we will multiply by $\pi/256$ , therefore we will get an angle in the range $[0, \pi)$ which represents the offset from the end of the range.
4. Finally, we subtract that from the end of the range to get the actual angle.

Similar to the previous method, the reverse operation is as follows:

1. Subtract the value from the end of the range to convert real angle to offset.
2. Divide by the angle granularity and round to the nearest integer to get the class number of the angle which will be in the range [0, 255].
3. Convert the previous number to a binary number.

## Dataset

We have used DOTA-v1.5 which contains 16 classes and 1400 images for training.

We trained on 15 classes instead of 16 (excluding the container-crane class) since there are only 237 instances in the whole dataset.

## Results and experiments

We can summarize what we do as follows:

1. Implement the AdaptiveAttention layer which we described earlier.
2. Implement the functionality needed for applying Dense Label Encoding and the corresponding loss function.
3. We found unreasonable results when applying the Dense Label Encoding loss and we proposed a minor change to it.
4. Discuss some probable weaknesses in using binary representation in prediction.
5. Training a lot of models and comparing them.

### Training

Since we don't have the resources that are available to the authors of the papers we try to enhance their methods, we do the following:

1. We don't make any hyperparameters tuning and we have used the same settings for all methods, this can lead to non-optimal performance but the converse is unpractical given the resources we have. But in order to avoid giving our methods any advantage over the original one, we used the settings written in the paper of the original model [14], hence we give them the advantage over our methods.
2. As an exception to what we said above, we will experiment with two values of the parameter that weigh the contribution of the angle classification loss and the loss of regressing other parameters of the boxes. It seems that we can give our model some advantage over the original one, but the original one did that, so we don't do anything that the original model did not do.

### Data preprocessing

Using the whole images in the dataset is computationally expensive for us, we are also forced to make the images smaller in order to be able to use the same mini-batch size as the original model, which is 2.

We can summarize what we did as follows:

1. We have 1400 images for training, we divided them into 1000 for training and the left for testing.
   The whole dataset is balanced and we randomly select the instances for each split, so it is unlikely that we have introduced a big bias.

2. It is recommended to divide the image with a window size 1024x1024 and a step size of 200, but we were forced to use a window size of 720x720 and a step size of 300.

**Experiments**

We use the following settings:

minibatch size = 2

epochs = 12

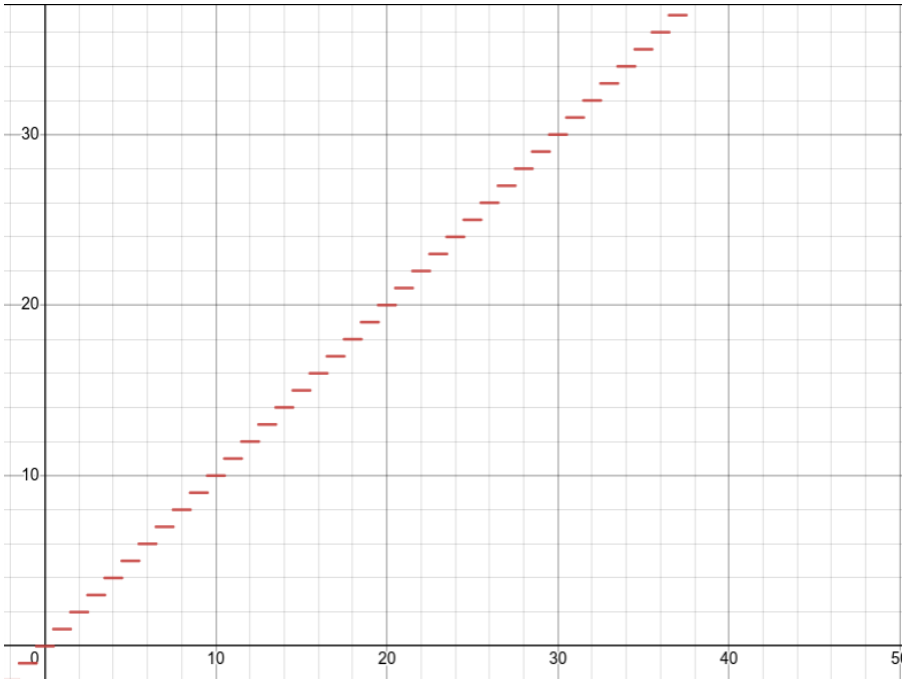
learning rate = 0.025

step decay = 10 after epochs 8, and 11

optimizer = SGD with momentum = 0.9 and weight decay = 1e-5

First, we tried to apply the dense label encoding on the two components of the network, the FAM, and ODM, the main difference between the two is that the results of FAM will go forward in the network and it is necessary to send as real-valued angle, not as a binary number.

Therefore this will introduce a problem caused by the discontinuity of the round function which is discontinuous and hence non-differentiable, but since we pass to it a probability that is always between 0, and 1 we have only one discontinuity point, but the gradient everywhere else will be 0, and hence no information will return to the earlier layers in the network.



we came up with some solutions to this problem:

1. We can keep the first component as it is, i.e let it use regression for angle instead of changing that to classification, and because the second component is responsible for the final predictions this will not affect what we are trying to do.
2. We can use a neural network to convert from binary to real-valued angle, this idea seems to be the best but we did not have the time to try it.

We discussed that the cost function itself does not have any idea about the distance between the angles, and it treats every class (from the eight classes here) the same, and we see that the weight function solved this problem by giving larger weight to larger distance, and use that weight with the cost function.

We observed that this is not enough. Despite the fact that the weight function introduces the idea of the distance between angles, it only works when we want to differentiate between two similar errors but with different classes, for example, when we have an error in only one bit, it can differentiate between an error in the last bit and an error in the first bit.

But does it able to differentiate in the general case?

We suspect that this is not true, for example, we must have a large loss value when we have an error in the last bit, which corresponds to a 90 degrees difference in the angle, which is the largest error that we can have.

But this does not happen since the loss function takes the average of values.

To solve this problem we propose using an additional weighting function that for each class takes the value of that class, i.e 2 to the power of the number of the class, divided by the max value. Then we apply the square root to increase the value of the small difference.

$$w_i = \sqrt{\frac{\theta_i}{\max_i \{\theta_i\}}}$$

As a comparison between our method, which we will call cls_weighting, and the original method we can see the values of errors for some examples in the following table:

| the value of cls_weighting loss | the value of the original loss | ground truth | prediction |
|---|---|---|---|
| 0.51 | 0.2 | 1, 0, … ,0 | 0.4, 0.1, …., 0.1 |
| 0.46 | 0.26 | 1, 0, … ,0 | 0.9, 0.9, 0.1, …,0.1 |
| 1.2 | 0.38 | 1, 0, … ,0 | 0.1, …, 0.1 |

There is another solution to this problem, which is by using the focal loss, we tried this too, but with no good results.

## Results

By looking at the results of the original model + attention, we can observe that the results for some objects enhanced a lot, however, it degraded for other objects. We hypothesize that this is maybe because of using the sigmoid in the attention which blocks the flow of the gradient, Hence, adding residual connections may improve the results even better.

Also, we think that the results of the original paper are better than ours because of the bigger minibatch size and better splitting of the images, i.e using 1024x1024 with a step size of 200.

The results we get from applying the cls_weighting are worse than what we think they will be, we think this may happen because the second component deals with small angles, since the first component should give good anchors, and hence there are not too many improvements needed.

We tried to see if the reversing of the weight will affect the results, and we can see from the results table that it gives results similar to the regular one, we don't have a good explanation for that, but we think that this may prove that the big angles are not needed in the second component, since the results are the same regardless of the weight.

We can conclude that the best result is when we added attention to the original model, and we observe that it enhances the objects with a few features the most.

| method | PL | BD | BR | GTF | SV | LV | SH | TC | BC | ST | SBF | RA | HA | SP | HC | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base model | **90.7** | 81.8 | 12.5 | 43.8 | **71.4** | **67.4** | **69.4** | **81.7** | 59.8 | 77.4 | 50.4 | 54.4 | 10.9 | 43.9 | **23.9** | 55.96 |
| RC-0.5 | **90.5** | **82.5** | 12.8 | 44.6 | 67.4 | 62.8 | 65.9 | 81.2 | 58.8 | **78** | 56.9 | 55.8 | **12.9** | 45.3 | 12.3 | 55.18 |
| RC-0.1 | **90.6** | 79 | 12.5 | **46.7** | 65.2 | 57 | 61.6 | 79.5 | 58 | 77.1 | **61.9** | 57.4 | 7.8 | 41.5 | 22.4 | 54.55 |
| RC-0.5 + cls_weighting | **90.7** | 73.7 | 12.5 | 39.3 | 67.7 | 58.5 | 62.8 | 80.3 | **62.1** | 79 | 56.9 | 63.7 | 9.2 | 43.9 | 18.5 | 54.59 |
| RC-0.5 + focal loss | **90.6** | 80 | 10.7 | 41.7 | 59.2 | 52 | 55 | 78 | 55.3 | 77.6 | 52 | 60.7 | 7 | 41.4 | 22.7 | 52.26 |
| Base model + attention | **90.6** | 76.8 | **13.8** | 43.2 | **71.5** | 64.7 | **69.2** | **81.6** | **62.1** | 76.2 | 55.1 | **65.7** | 10 | **46.2** | 21 | **56.51** |
| RC-0.5 + attention with res links | **90.6** | 79.8 | 13 | 42.2 | 66.3 | 59.1 | 64 | 80.4 | 57.9 | **78.1** | 54.4 | 61.3 | 6.6 | **48.4** | 11 | 54.20 |
| RC-0.5 + cls_weighting reversed | **90.5** | 80.4 | 11.5 | **46.8** | 66.5 | 57.8 | 63.7 | 80.9 | 58.9 | 77.6 | 55.5 | 58.1 | 8 | 43.7 | 19.4 | 54.62 |

RC-0.1 represents the proposed modification by converting the prediction of angles in the second component to classification instead of regression, and by weighting the corresponding loss by 0.1.
RC-0.5 is like the previous one.
We tried using attention with residual connections with the modified model, but with no success.

## References


[1] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, arXiv:1311.2524v5.

[2] Ross Girshick. Fast R-CNN, arXiv:1504.08083v2.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, arXiv:1506.01497v3.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. Mask R-CNN, arXiv:1703.06870v3.

[5] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, arXiv:1312.6229v4.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection, arXiv:1506.02640v5.

[7] Joseph Redmon, Ali Farhadi. YOLO9000: Better, Faster, Stronger, arXiv:1612.08242v1.

[8] Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement, arXiv:1804.02767v1.

[9] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection, arXiv:2004.10934v1.

[10] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, Jian Sun. YOLOX: Exceeding YOLO Series in 2021, arXiv:2107.08430v2.

[11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector, arXiv:1512.02325v5.

[12] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, Yichen Wei. Deformable Convolutional Networks, arXiv:1703.06211v3.

[13] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, Jianbin Jiao. Oriented Response Networks, arXiv:1701.01833v2.

[14] Jiaming Han, Jian Ding, Jie Li, Gui-Song Xia. Align Deep Features for Oriented Object Detection, arXiv:2008.09397v3.

[15] Wang, J.; Ding, J.; Guo, H.; Cheng, W.; Pan, T.; Yang, W. Mask OBB: A Semantic Attention-Based Mask Oriented Bounding Box Representation for Multi-Category Object Detection in Aerial Images. *Remote Sens.* **2019**, *11*, 2930. https://doi.org/10.3390/rs11242930.

[16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie. Feature Pyramid Networks for Object Detection, arXiv:1612.03144v2.
[17] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu. Squeeze-and-Excitation Networks, arXiv:1709.01507v4.
[18] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Andrea Vedaldi. Gather-Excite: Exploiting Feature Context in Convolutional Neural Networks, arXiv:1810.12348v3.
[19] Sanghyun Woo, Jongchan Park, Joon-Young Lee, In So Kweon. CBAM: Convolutional Block Attention Module, arXiv:1807.06521v2.
[20] Li, W., Liu, K., Zhang, L. et al. Object detection based on an adaptive attention mechanism. Sci Rep 10, 11307 (2020). https://doi.org/10.1038/s41598-020-67529-x.
[21] Xue Yang and Junchi Yan. Arbitrary-oriented object detection with circular smooth label. In Proceedings of the European Conference on Computer Vision, pages 677–694. Springer, 2020.
[22] Xue Yang, Liping Hou, Yue Zhou, Wentao Wang, Junchi Yan. Dense Label Encoding for Boundary Discontinuity Free Rotation Detection, arXiv:2011.09670v4.
[23] J. Wang, W. Yang, H. -C. Li, H. Zhang and G. -S. Xia, "Learning Center Probability Map for Detecting Objects in Aerial Images," in IEEE Transactions on Geoscience and Remote Sensing, vol. 59, no. 5, pp. 4307-4323, May 2021, doi: 10.1109/TGRS.2020.3010051.