**Implementation Results:**

**Step1:**

**Global Wheat Head detection(DWHD)**

**Implementation process :**

1. **DATASET:**

[Global Wheat Head Detection (GWHD) Dataset](https://r.search.yahoo.com/_ylt=AwrwXxRX4_9fpH4AbB3nHgx.;_ylu=Y29sbwMEcG9zAzEEdnRpZAMEc2VjA3Ny/RV=2/RE=1610634200/RO=10/RU=https%3a%2f%2fspj.sciencemag.org%2fjournals%2fplantphenomics%2f2020%2f3521852%2f/RK=2/RS=j7FfP7wpfqxzo2PYgZGH1MNnAug-" \t "_blank) is download from below link.

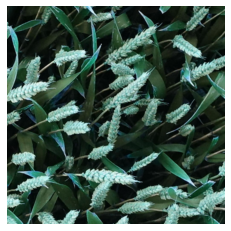
<http://www.global-wheat.com/data-description/>

1. **Pr-processing:**

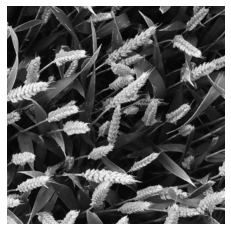
Sequence of image pr-processing algorithms used for lens distortion, White

balance, Geometric transformations using filtering algorithm.

1. Input image



1. Grayscale conversion



1. Geometric transformation

Geometric operations is applied for correcting the rotation of wheat image.



1. Gaussian Filtering

For lens distortion correction Gaussian filter is used.



Image enhancement using Graph theory is implemented for contrast adjustment.



3:Albumentation :

Albumentations is a Python library for fast and flexible [image augmentations](https://albumentations.ai/" \o "). Albumentations efficiently implements a rich variety of image transform operations that are optimized for performance, and does so while providing a concise, yet powerful image augmentation interface for different computer vision tasks, including object classification, segmentation, and detection.

**Existing System:**

In the existing system, we use RCNN, FCN fully Convolutional Network as approaches for Object Detection or in this case Wheat Detection.

RCNN: R-CNNs ( Region-based Convolutional Neural Networks) a family of machine learning models Specially designed for object detection, the original goal of any R-CNN is to detect objects in any input image.

An input image given to the R-CNN model goes through a mechanism called selective search to extract information about the region of interest. Region of interest can be represented by the rectangle boundaries. Depending on the scenario there can be over 2000 regions of interest. This region of interest goes through CNN to produce output features. These output features then go through the [SVM](https://analyticsindiamag.com/understanding-the-basics-of-svm-with-example-and-python-implementation/)(support vector machine) classifier to classify the objects presented under a region of interest.

## Problems with R-CNN

1.It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.

2.It cannot be implemented real time as it takes around 47 seconds for each test image.

3.The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

Disadvantages:

1.CNN **do not encode the position and orientation of object**. Lack of ability to be spatially invariant to the input data. Lots of training data is required

**FCN:**

* ****Image Classification****: Classify the object (Recognize the ****object class****) within an image.
* ****Object Detection****: Classify and detect the object(s) within an image with bounding box(es) bounded the object(s). That means we also need to know the ****class, position and size of each object****.
* ****Semantic Segmentation****: Classify the ****object class for each pixel****within an image. That means there is ****a label for each pixel****.

**Drawbacks:**

1. . **FCN often smooths detailed structures and ignores small objects** and
2. .CRF is employed as a standalone post-processing step disconnected from the FCN.

**Proposed System:**

1. **Net Architecture**

UNet, evolved from the traditional convolutional neural network,As a general convolutional neural network focuses its task on image classification.

UNet is dedicated to solving this problem. **The reason it is able to localise and distinguish borders is by doing classification on every pixel, so the input and output share the same size.**

Now let’s get to the detail implementation of UNet. I will:

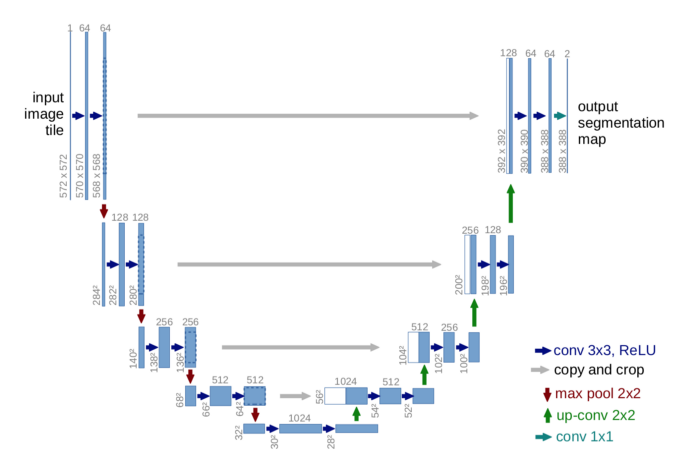
1. Show the overview of UNet
2. Breakdown the implementation line by line and further explain it

# **Overview**

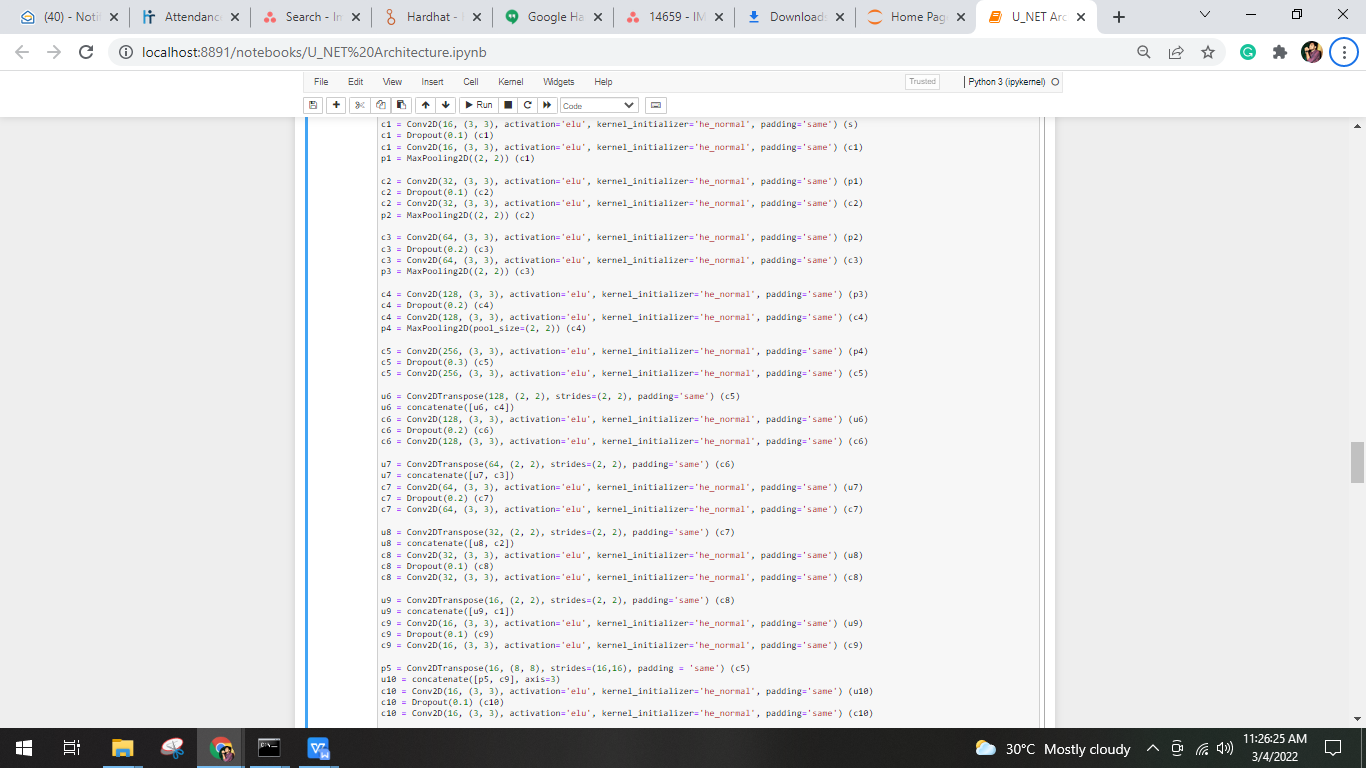
**Network Architecture:**

The network consists of a contracting path and an expansive path, which gives it the u-shaped architecture. The contracting path is a typical convolutional network that consists of repeated application of [convolutions](https://en.wikipedia.org/wiki/Convolutions" \o "Convolutions), each followed by a [rectified linear unit](https://en.wikipedia.org/wiki/Rectified_linear_unit" \o "Rectified linear unit) (ReLU) and a [max pooling](https://en.wikipedia.org/wiki/Max_pooling" \o "Max pooling) operation. During the contraction, the spatial information is reduced while feature information is increased. The expansive pathway combines the feature and spatial information through a sequence of up-convolutions and concatenations with high-resolution features from the contracting path.

The network has basic foundation looks like:



First sight, it has a “U” shape. The architecture is symmetric and consists of two major parts — the left part is called contracting path, which is constituted by the general convolutional process; the right part is expansive path, which is constituted by transposed 2d convolutional layers(you can think it as an upsampling technic for now).



Now let’s break down the implementation line by line and maps to the corresponding parts on the image of UNet architecture.

# **Line by Line Explanation**

## Contracting Path :

The contracting path follows the formula:

**conv\_layer1 -> conv\_layer2 -> max\_pooling -> dropout(optional)**

So the first part of our code is:

**inputs = Input((IMG\_HEIGHT, IMG\_WIDTH, 3))**

**s = Lambda(lambda x: x / 255) (inputs) # rescale inputs**

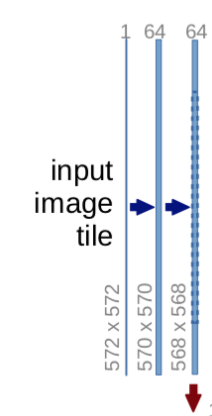
**c1 = Conv2D(16, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (s)**

**c1 = Dropout(0.1) (c1)**

**c1 = Conv2D(16, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c1)**

**p1 = MaxPooling2D((2, 2)) (c1)**

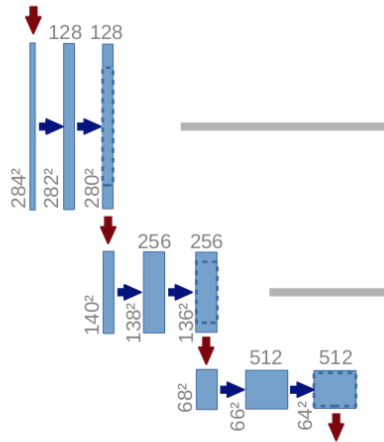
which matches to:



****Notice that each process constitutes two convolutional layers****, and the number of channel changes from 1 → 64, as convolution process will increase the depth of the image. The red arrow pointing down is the max pooling process which halves down size of image(the size reduced from 572x572 → 568x568 is due to padding issues, but the implementation here uses padding= “same”).

The process is repeated 3 more times:

with code:



**c2 = Conv2D(32, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (p1)**

**c2 = Dropout(0.1) (c2)**

**c2 = Conv2D(32, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c2)**

**p2 = MaxPooling2D((2, 2)) (c2)**

**c3 = Conv2D(64, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (p2)**

**c3 = Dropout(0.2) (c3)**

**c3 = Conv2D(64, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c3)**

**p3 = MaxPooling2D((2, 2)) (c3)**

**c4 = Conv2D(128, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (p3)**

**c4 = Dropout(0.2) (c4)**

**c4 = Conv2D(128, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c4)**

**p4 = MaxPooling2D(pool\_size=(2, 2)) (c4)**

and now we reaches at the bottom most:

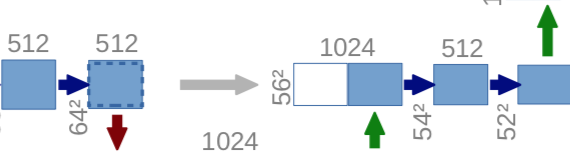
IMG_256

still****2 convolutional layers are built, but with no max pooling****:

The image at this moment has been resized to 28x28x1024. Now let’s get to the expansive path.

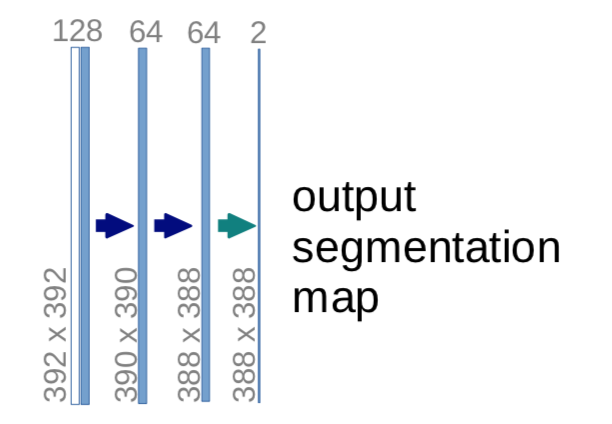
In the expansive path, the image is going to be upsized to its original size. The formula follows:

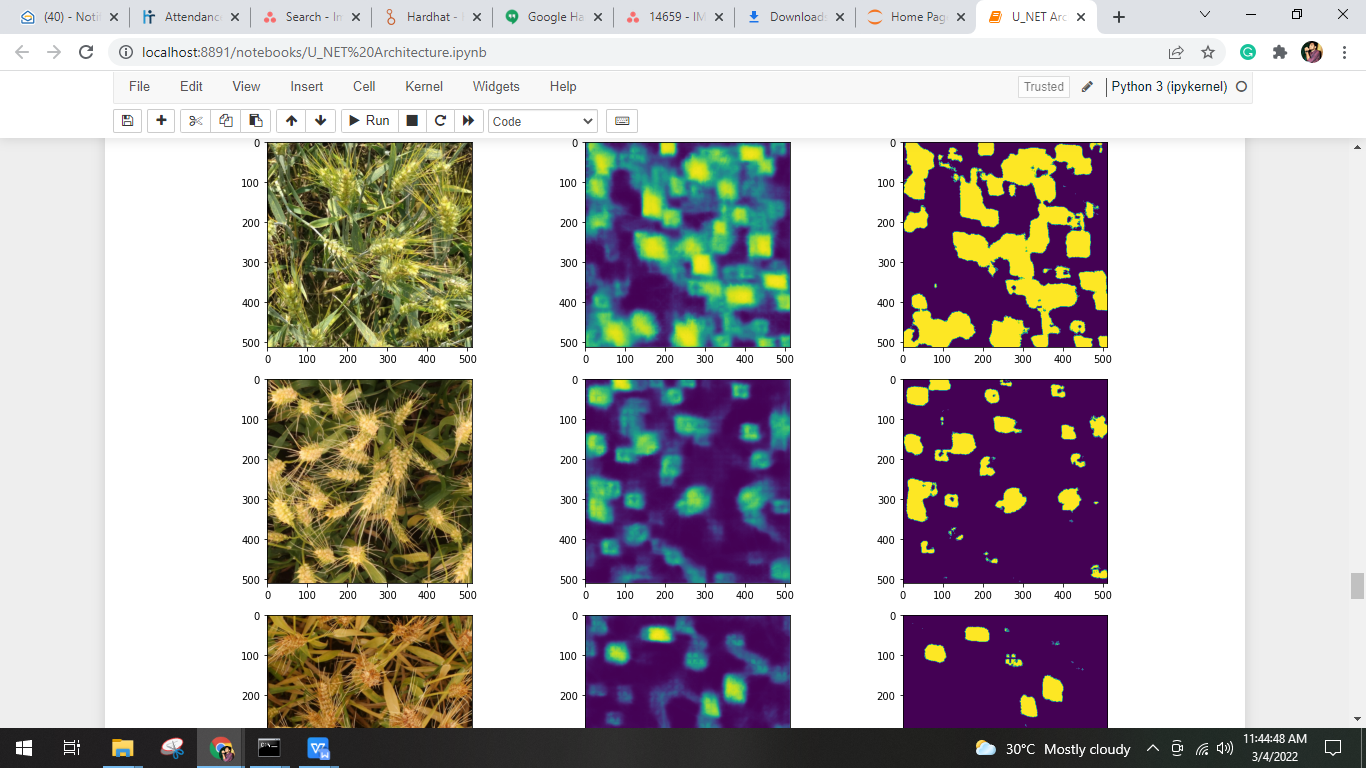
**conv\_2d\_transpose -> concatenate -> conv\_layer1 -> conv\_layer2**

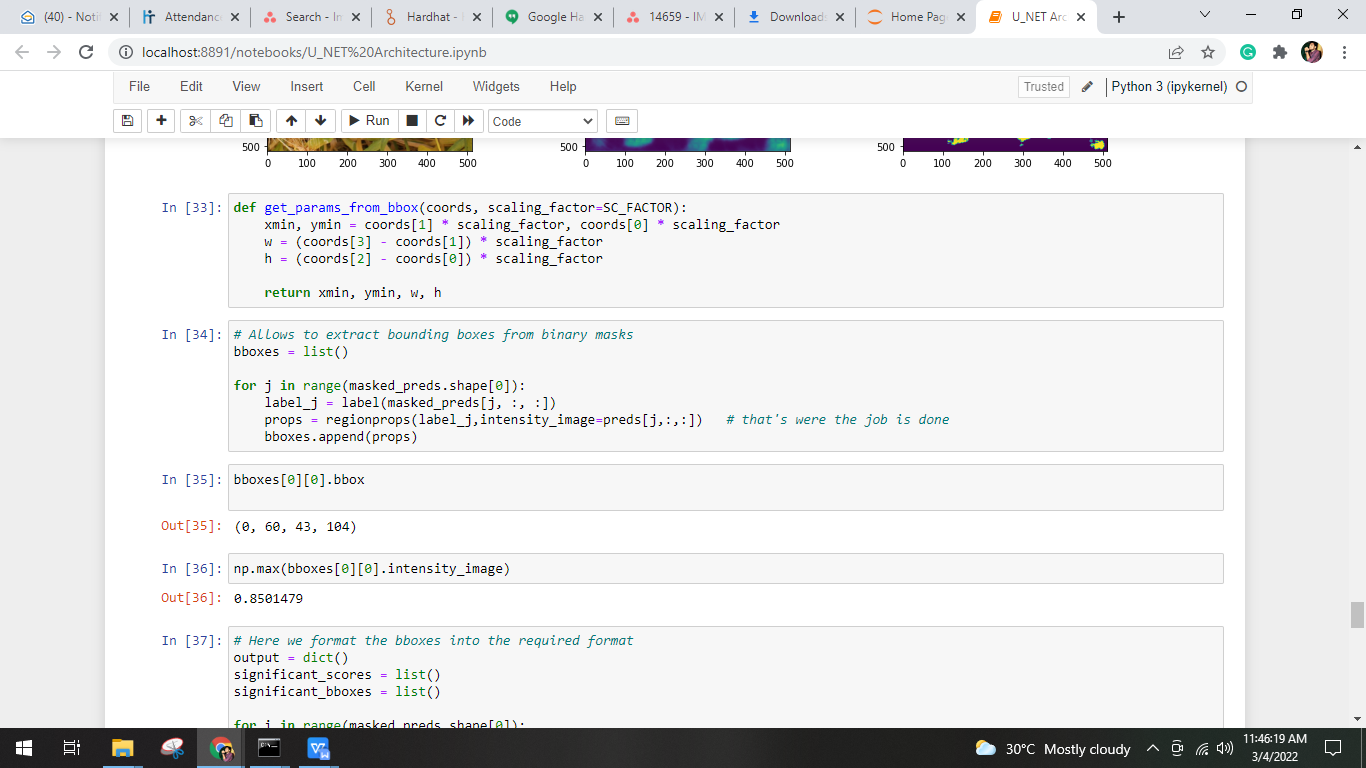


After the transposed convolution, the image is upsized from 28x28x1024 → 56x56x512, and then, this image is concatenated with the corresponding image from the contracting path and together makes an image of size 56x56x1024. The reason here is to combine the information from the previous layers in order to get a more precise prediction.

Now we’ve reached the uppermost of the architecture, the last step is to reshape the image to satisfy our prediction requirements.





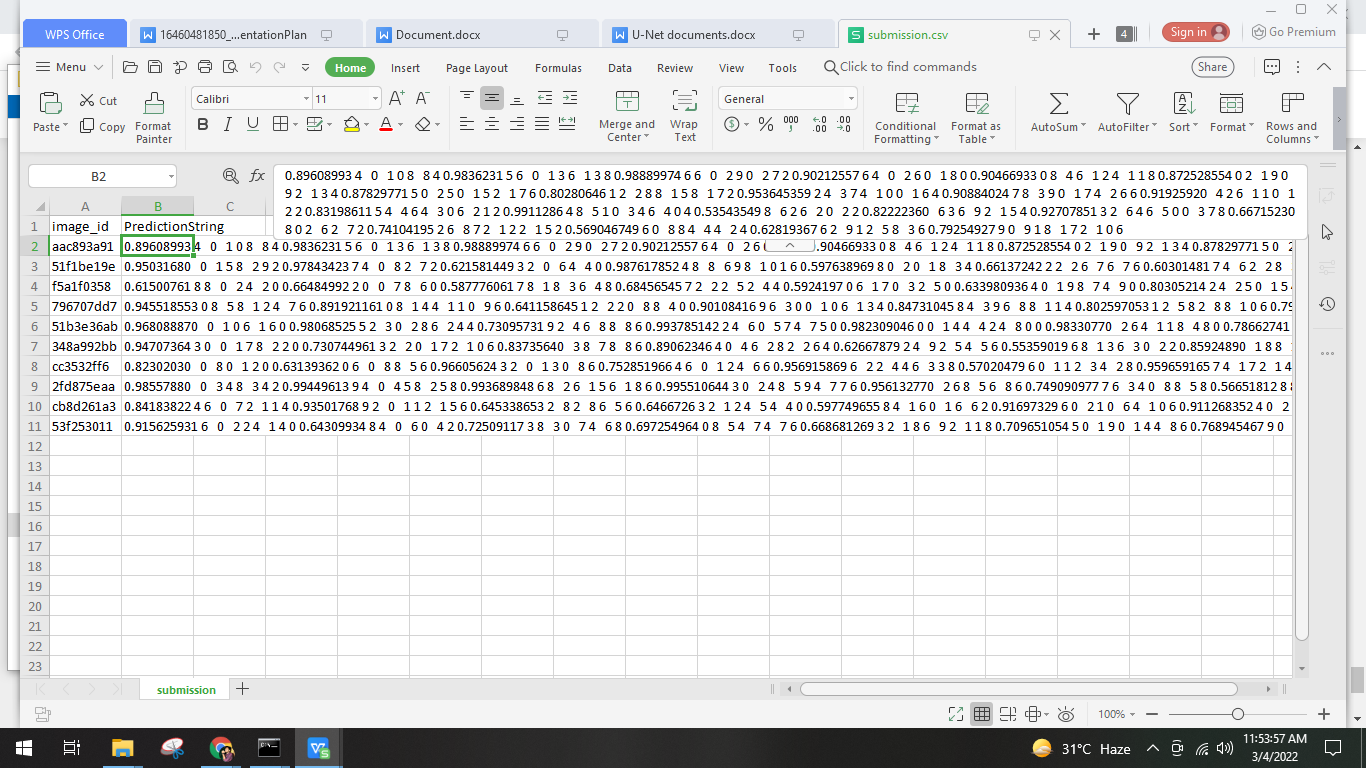


# 

# 

**Output:**

This Model Output is located in **Submission.csv**



# **Conclusion:**

There are many applications of image segmentation using UNet and it also occurs in lots of competitions.

**Applications:**

1. Pixel-wise regression using U-Net and its application on pansharpening
2. U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation
3. TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation.
4. Image-to-image translation to estimate fluorescent stains
5. In binding site prediction of protein structure

**Reference link for IOU:**

<https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>