Vietnam National University Ho Chi Minh City
University of Information Technology
Faculty of Computer Science

Introduction to Computer Vision

Body Segmentation on TikTok images

Le Doan Phuc Minh - 20520243

Nguyen Duy Dat - 20520435

Phan Doan Thai Binh - 20520043

Lecturer: PhD. Mai Tien Dung

Table of Contents

Introduction and Problem Information 4

Introduction 4

Dataset Information 4

Method 5

Classic methods for Image Segmentation 5

K-Means 5

Introduction to Clustering Algorithm 5

K-Means clustering algorithm 6

Problems 6

Meanshift 7

Meanshift clustering algorithm 7

Problems 8

Grabcut 8

Introduction to Grabcut 8

Problems 8

U-Net – Semantic Segmentation 10

Introduction 10

Architecture 10

Model implementation 11

Model 11

Loss function 11

Metric 12

Experiments 13

Data split 13

Training 14

Result 14

Conclusion 15

Mask R-CNN – Instance Segmentation 16
Introduction 16
Faster R-CNN 17
Mask R-CNN 18
Model implementation 21
References 23

Introduction and Problem Information

Introduction

Image Segmentation is a broad part of Machine Vision, in image segmentation we classify every pixel of the image into one of the classes. Image segmentation has 2 main forms: Semantic and Instance segmentation.

Fig 1. Target of image segmentation

In this report we will apply some approaches on Tiktok dancing video for body segmentation. Tiktok is a short-form video hosting, it hosts a variety of short-form user videos, from genres like pranks, stunts, tricks, jokes, dance, and entertainment with durations from 15 seconds to ten minutes. Body segmentation could be very useful for filters on Tiktok which are fascinated by users. Our input has one image, image segmentation's target is a mask of input image. For semantic segmentation, its result returns people (or group) and background. More advanced in instance segmentation, its output is people (distinguish people) or detail object.

Dataset Information

Dataset consists of 2615 images with their masks. Image can have one or two people dancing in diverse places. We will consider each image as an individual one. Our mission is solving problem with input is an image, we hope in future we will be able to improve it to handle with videos.

Method

Before implementing with the state of art methods such as Unet and Mask RCNN, we will evaluate classic Kmeans, Meanshift and Grabcut methods that if they could lead to expected output.

Classic methods for Image Segmentation

K-Means

Introduction to Clustering Algorithm

To understand about Clustering Algorithm, we need to understand what Clustering is first. Take an example when a person listening to music, to have a best listening experience, they will be trying to select music maybe based on genre, or maybe based on period, or even popularity. And then they will group music tracks based on the criteria they have chosen and pick out which group fit them most. Although each person will have many ways of grouping music tracks, what they all have learned is to find similarity between music to put them into a desired group.

In Machine Learning, we often group examples as a first step to understand a subject (dataset) in a machine learning system. Grouping unlabeled examples is called clustering. A Clustering Algorithm is an unsupervised learning algorithm that do the clustering task on the given data by clustering or partitioning those data into K-clusters or parts based on the K-centroids.

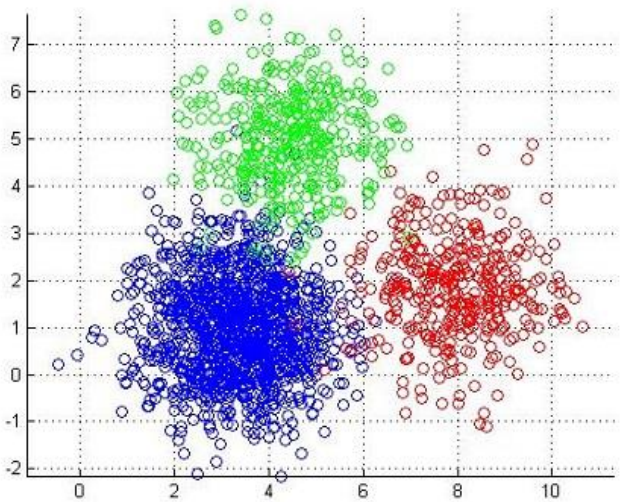

Fig 2. An example of Clustering

K-Means clustering algorithm

K-Means clustering algorithm is an unsupervised algorithm and it is used to segment the interest area from the background. It clusters or partitions the given data into K-clusters or parts based on the K-centroids. The algorithm is used when you have unlabeled data (data without defined categories or groups).

In Image Segmentation, these input data are pixels from a given image. The goal of this algorithm is to cluster the image into K clusters (K is user- defined). The value of each pixel will be replaced by their centroids to segment the image.

Fig 3. Before and after of Image Segmentation using K-Means (8 clusters)

Problems

As metioned earlier, since the K-value must be user-defined, this means the result of clustering will be depended on the number of clusters that user have chosen.

Secondly, this algorithm is computationally expensive because while training, we need to calculate the distance from centroids to all pixels in the image which can be problematic with large images.

And although this algorithm captures pixel similarity, it doesn't capture continuity of contours, captures near/far relationships only weakly and can merge far away object together.

Fig 4. How number of clusters affects output result (6, 8, 10 clusters respectively)

Meanshift

Meanshift clustering algorithm

Meanshift clustering algorithm is a centroid-based algorithm that helps in various use cases of unsupervised learning. It works by shifting data points towards centroids to be the mean of other points in the region. This is also known as the mode seeking algorithm. Unlike K-Means, it assigns clusters to the data without automatically defining the number of clusters based on defined bandwidth.

Fig 5. Before and after of Image Segmentation using Meanshift

Problems

Although this algorithm is flexible in number and shape of regions and robust to outliers, Meanshift algorithm does not work well in case of high-dimension features per pixel (color, gradients, texture, etc.), where number of clusters changes abruptly.

Another problems with this kind of clustering algorithm which also apply to K-Means algorithm is it does not truly differentiate person with background, rather they only clustering by color similiarities.

Grabcut

Introduction to Grabcut

GrabCut is an image segmentation method based on graph cuts. Starting with a user-specified bounding box around the object to be segmented, the algorithm estimates the color distribution of the target object and that of the background using a Gaussian mixture model. This is used to construct a Markov random field over the pixel labels, with an energy function that prefers connected regions having the same label, and running a graph cut based optimization to infer their values. As this estimate is likely to be more accurate than the original, taken from the bounding box, this two-step procedure is repeated until convergence.

Fig 6. Before and after using Grabcut to get instance (Source: docs.opencv.org)

Problems

As clearly seen, because of the fact Grabcut require user to choose bounding box, this means the result of Grabcut will be affected based on the selected bounding box. If the bounding box doesn't have a good coverage, it may result the instance

isn’t perfectly segmented or sometimes can be incorrectly segmented (No segmented instance, wrong target instance….).

Secondly, some situations require user to manually load the mask to create better result. The example on top required us to create a mask layer (aka the layer with the unknown black and white brush) so that this method can avoid the masked detail. Without the masked layer, some unwanted instance may go into the result. Because of this, it creates another inconvenience to this method.

Fig 7. Before and after using Grabcut (without mask layer)

And even with the mask layer, this method is still struggle against some situation such as camourflaged objects.

U-Net – Semantic Segmentation

Introduction

U-Net is a convolutional neural network originally developed for segmenting biomedical images. When visualized its architecture looks like the letter U and hence the name U-Net.

Architecture

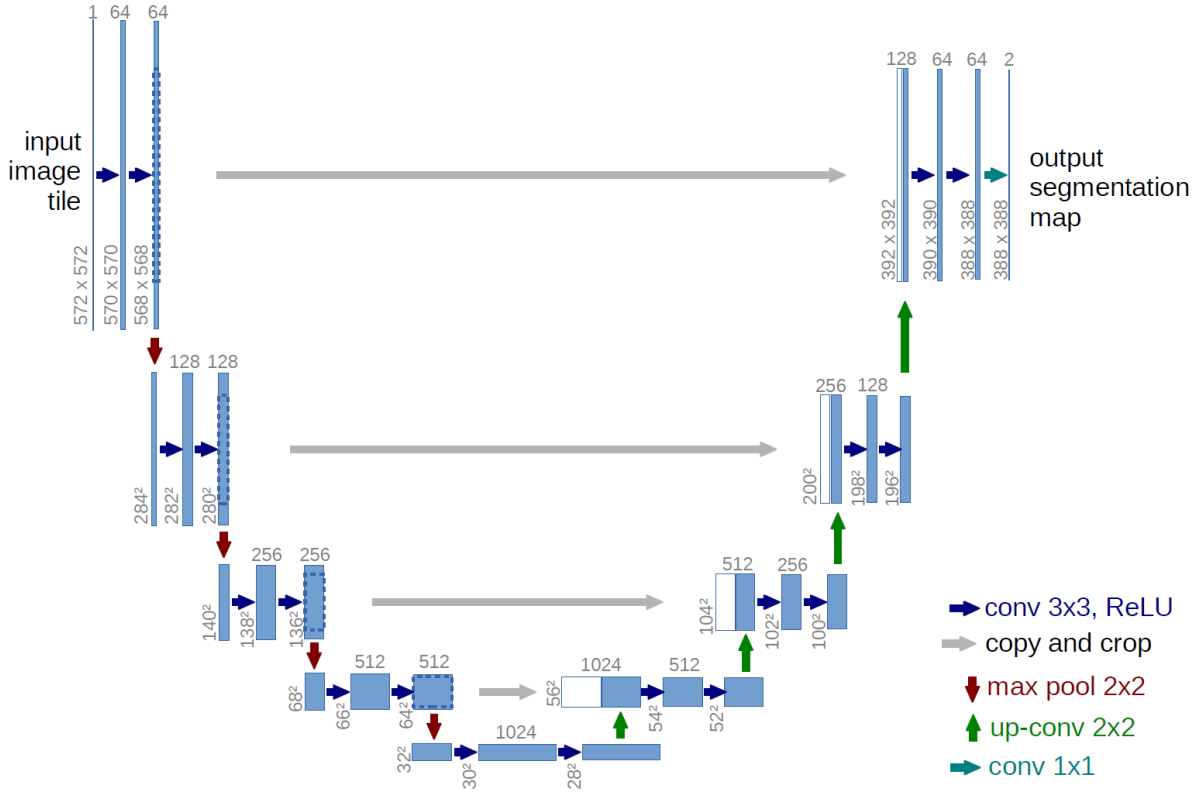

Fig 8. Architecture of U-Net

Its architecture is made up of two parts, the left part – the contracting path and the right part – the expansive path. The purpose of the contracting path is to capture context while the role of the expansive path is to aid in precise localization.

First, let's talk about CNN. CNN learns features from the images and compressed image into a feature vector, which we can use in image classification and other things.

Now, talk about Unet- In Segmentation, we need to reconstruct the image from the feature vector created by CNN. So, here we convert the feature map into a vector and reconstruct an image from this vector.

We use U-Net because it can reconstruct the image. We will feed some images as features and their respected mask images as labels to the model. Because of the reconstructive features of U-net, the U-Net will be able to generate images as output also. Here we are using a supervised learning approach.

Model implementation

Model

After implementing the model of U-Net, we wil be able to see that the number of parameters is super huge (more than 30 million parameters). This model is not really suitable for us to train, so we need to try a difference approach, and that is a similar model with number of filters is just one quarter of the original U-Net. The model is introduced by Dr. Sreenivas Bhattiprolu. The final parameter number is 1,941,105.

Loss function

Semantic segmentation models usually use a simple binary-cross entropy loss function during training.

Cross-entropy is used to measure the difference between two probability distributions. It is used as a similarity metric to tell how close one distribution of random events is to another and is used for both classification (in the more general sense) as well as segmentation.

The binary cross-entropy (BCE) loss therefore attempts to measure the differences of information content between the actual and predicted image masks. It is more generally based on the Bernoulli distribution and works best with equal data-distribution amongst classes. In other terms, image masks with very heavy class imbalance may (such as in finding very small, rare tumors from X-ray images) may not be adequately evaluated by BCE.

This is because the BCE treats both positive (1) and negative (0) samples in the image mask equally. Since there may be an unequal distribution of pixels that represent a given object (say, a car from the first image above) and the rest of the image, the BCE loss may not effectively represent the performance of the deep-learning model.

Binary Cross Entropy is defined as:

$$\text{Loss} = -\frac{1}{\substack{\text{output} \\ \text{size}}} \sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Fig 9. Binary Cross Entropy function

If y is our target image-segmentation mask, and $\hat{y}$ is our predicted mask from our deep-learning model, the loss measures the difference between what we want (y) and what the model gave us ($\hat{y}$).

This has been implemented in TensorFlow's *keras.losses* package and as such, can be readily used as-is in your image segmentation models.

Metric

Pixel accuracy is perhaps the easiest to understand conceptually. It is the percent of pixels in your image that are classified correctly.

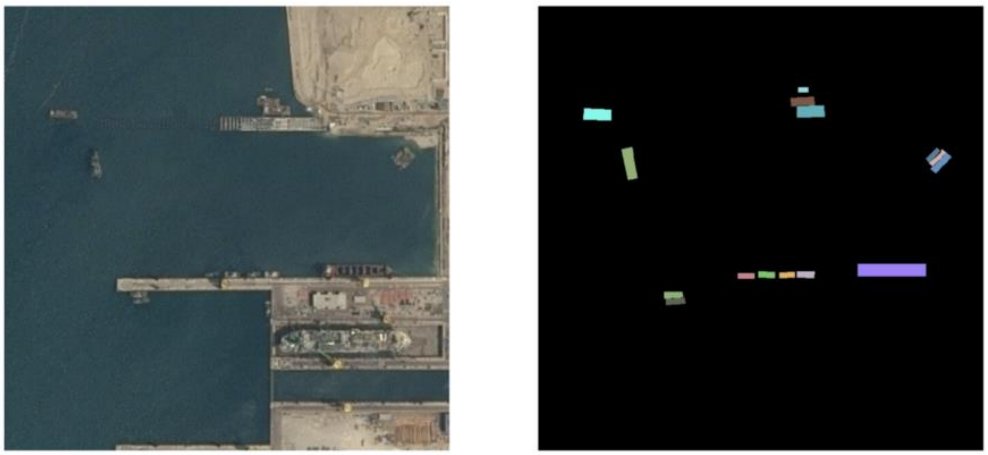

Fig 10. Image from Vlad Shmyhlo in article:

Image Segmentation: Kaggle experience

But the problem is that high pixel accuracy doesn't always imply superior segmentation ability. Like we can get a 95% accuracy for the image above [Fig 10] with a totally black mask.

This issue is called class imbalance. When our classes are extremely imbalanced, it means that a class or some classes dominate the image, while some other classes make up only a small portion of the image. Unfortunately, class imbalance is prevalent in many real-world data sets, so it can't be ignored. Therefore, I use another alternative metric that are better at dealing with this issue: *Intersection-Over-Union*.

The Intersection-Over-Union (IoU), also known as the Jaccard Index, is one of the most commonly used metrics in semantic segmentation… and for good reason. The IoU is a very straightforward metric that's extremely effective.

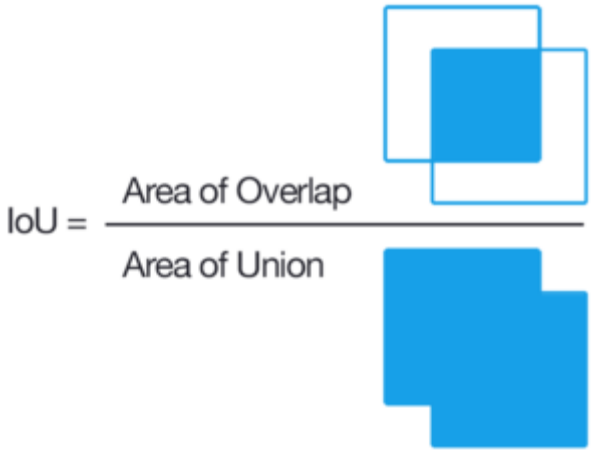

Fig 11. IoU calculation visualized. Source: Wikipedia

Simply put, the IoU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth, as shown on the image to the left. This metric ranges from 0–1 (0–100%) with 0 signifying no overlap and 1 signifying perfectly overlapping segmentation.

For binary (two classes) or multi-class segmentation, the mean IoU of the image is calculated by taking the IoU of each class and averaging them.

If we try to use a totally black mask for the image above [Fig 10] like we did with pixel accuracy, after doing the calculations, we get that the IoU is merely 47.5%! Its better present the accuracy than pixel method.

Experiments

Data split

We split the dataset into 2 sets:

- Train dataset: fit the model – 75%
- Validate dataset: evaluate the fit model – 25%

Training

I trained the model on Google Colab with GPU for 50 epochs (took about 1 hour and 20 minutes) and plot the accuracy/loss.

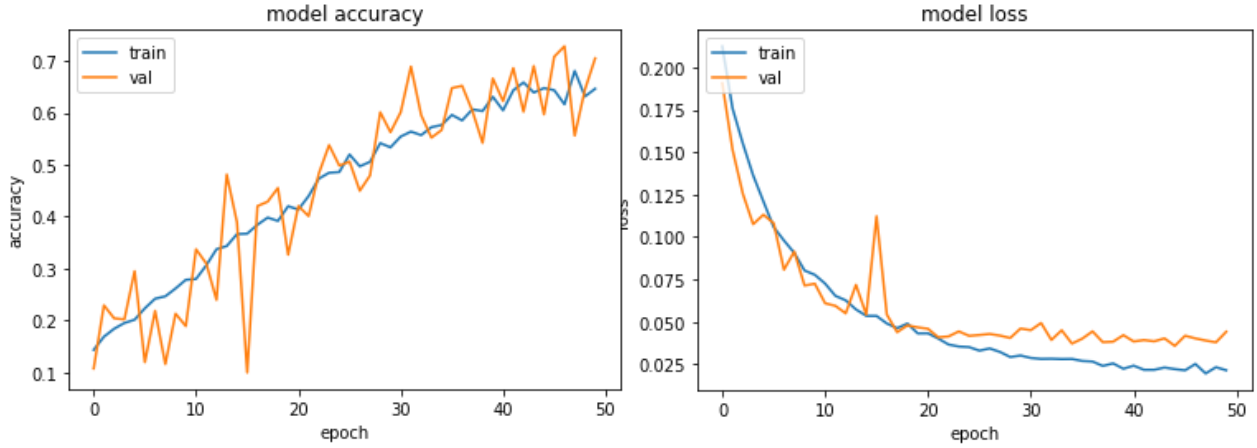

Fig 12. Accuracy and loss after each epoch

As we can see, the accuracy reaches a good level (0.6-0.7), and the model loss is getting not-decrease in epoch 40-50. So, we decided to stop the training and choose the weights we get after epoch 45 (which has the lowest validate loss = 0.035) as final weights for our model.

Result

Average iou accuracy on whole dataset is 0.736.
Average time to predict an image is 0.686 second.

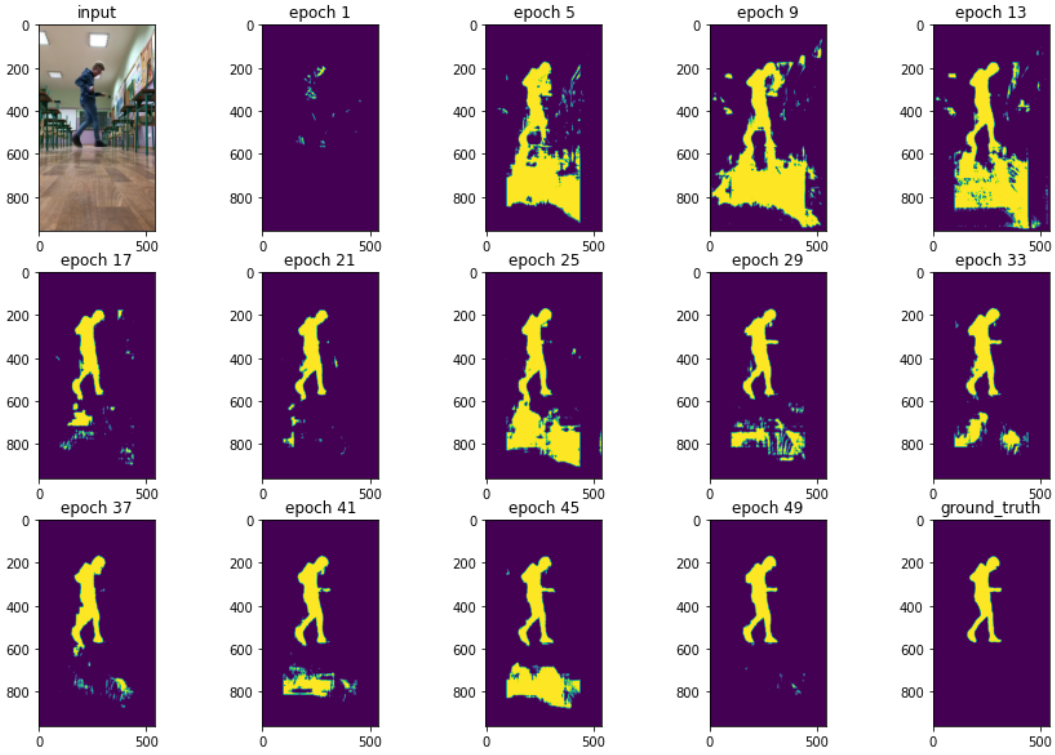

Fig 13. The predictions of an image after each 4 epochs

The model is trying to predict better after epoch, for some image like above, its accuracy could be increase and increase, but for other image the accuracy may be decrease. So, we should think wisely how we decide the final model. In this report, we choose the lowest validate loss one.

Conclusion

Unet prediction is good (overall iou_score = 0.736) when there is no noise in image and brightness is well. However, Unet is quite slow (about 0.686 second / prediction – average on 2615 predictions), so this should only be applied on offline-application.

Mask R-CNN – Instance Segmentation

Introduction

The vision community has rapidly improved object detection and semantic segmentation results over a short period of time. In large part, these advances have been driven by powerful baseline systems, such as the Fast/Faster RCNN and Fully Convolutional Network (FCN) frameworks for object detection and semantic segmentation, respectively. These methods are conceptually intuitive and offer flexibility and robustness, together with fast training and inference time. Our goal in this work is to develop a comparably enabling framework for instance segmentation.

Instance segmentation requires the correct detection of all objects in an image while also precisely segmenting each instance. It therefore combines elements from the classical computer vision tasks of object detection, where the goal is to classify individual objects and localize each using a bounding box, and semantic segmentation, where the goal is to classify each pixel into fixed set of categories without differentiating object instances. Given this, one might expect a complex method is required to achieve good results.

Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI). The Mask branch is a FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.

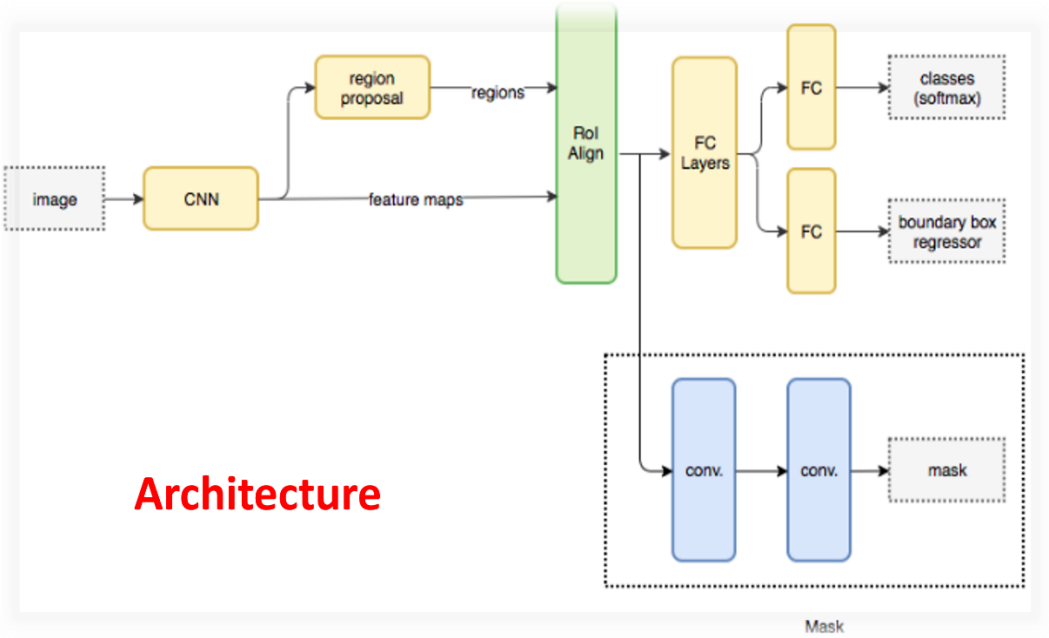

Fig 14. Architecture of Mask-RCNN

The architecture image illustrates us two main Mask R-CNN's components are Faster R-CNN and FCN. Faster R-CNN has two outputs for each candidate object, a class label and a bounding-box offset; to this we add a third branch that outputs the object mask.

Faster R-CNN

Faster R-CNN consists of two stages. The first stage, called a Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage, which is in essence Fast R-CNN, extracts features using RoIPool from each candidate box and performs classification and bounding-box regression. The features used by both stages can be shared for faster inference. We refer readers to for latest, comprehensive comparisons between Faster R-CNN and other frameworks.

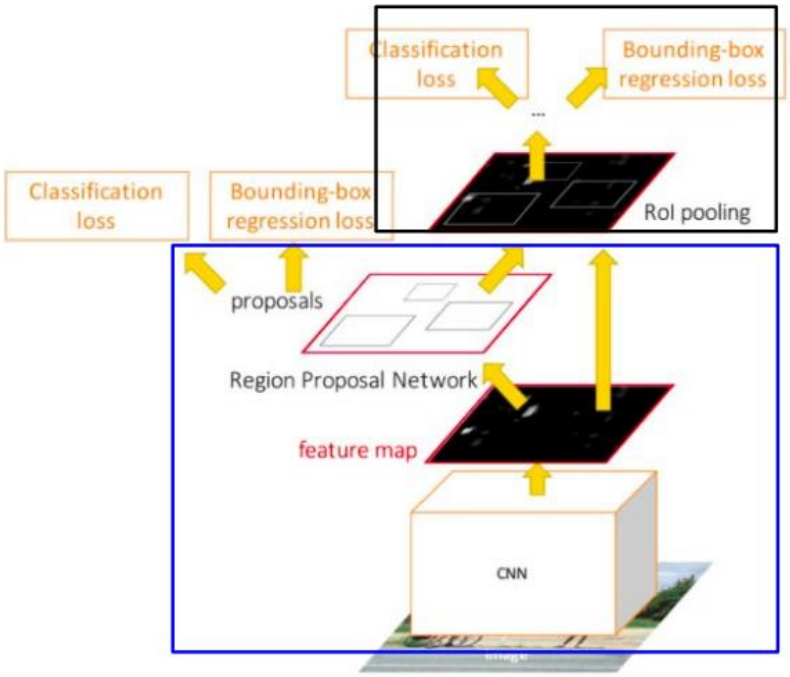

Fig 15. Architecture of Faster-RCNN

There are four main loss functions: classification loss, bounding box for proposals and classification loss, bounding box final prediction. Reason why feature maps transform from proposals to RoI pooling is resizing feature maps because they have other sizes.

Mask R-CNN

Mask R-CNN adopts the same two-stage procedure, with an identical first stage (which is RPN). In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI. This contrasts with most recent systems, where classification depends on mask predictions. Our approach follows the spirit of Fast R-CNN that applies bounding-box classification and regression in parallel (which turned out to largely simplify the multi-stage pipeline of original R-CNN.

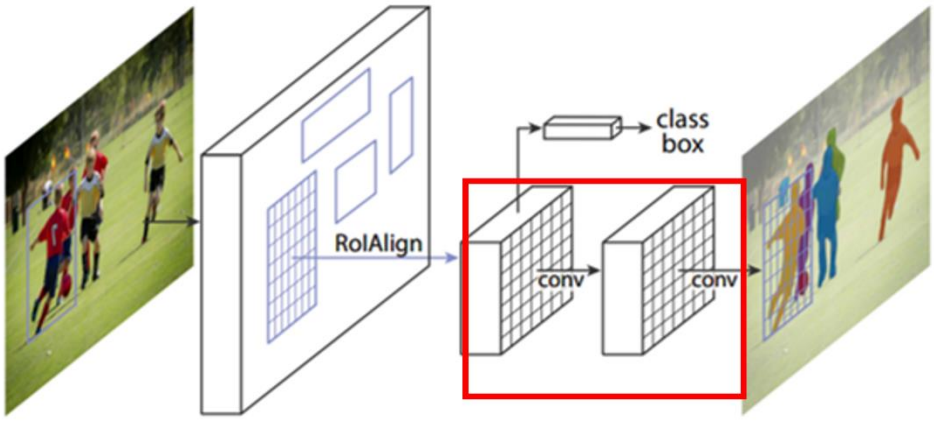

Fig 16. The main branch of mask-RCNN

Important point in Mask R-CNN is predicting a mask from each RoI using an FCN and maintaining the explicit object spatial layout without collapsing it into a vector representation that lacks spatial dimensions.

This pixel-to-pixel behavior requires our RoI features, which themselves are small feature maps, to be well aligned to faithfully preserve the explicit per-pixel spatial correspondence. This motivated us to develop the following RoIAlign layer that plays a key role in mask prediction.

Then, we will compare RoI Align and RoI Pooling:

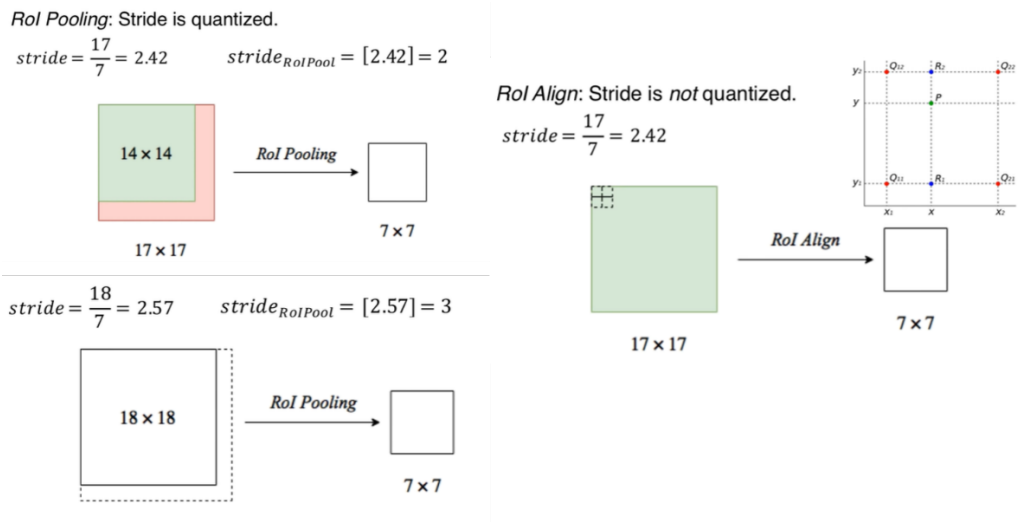

Fig 16. The difference between RoI Pooling and RoI Align

In term of RoI Pooling, it is quantized meaning stride is rounded (2.42 to 2, 2.57 to 3 respectively image). For RoI Align, it is not quantized meaning stride is not changed, keep decimal part (2.42 to 2.42). Thus, RoI Align have higher precision than RoI Pooling. RoI Align applies bilinear interpolation to calculate the weights for each point. This is a large improvement for predicting pixel-to-pixel level.

Bilinear interpolation extends linear interpolation. The interpolation process uses the nearest 4 pixels to calculate the value of the new pixel. The value of the interpolated point is averaged average of the four nearest points, the weights for each point are calculated based on the distance of that point with the point to be interpolated.

Mathematical function of bilinear interpolation method:

$$h(x) = \begin{cases} 1 - |x| & |x| \leq 1 \\ 0 & 1 \leq |x| \end{cases}$$

where x is the distance between the interpolation point and the grid point. To calculate the value at the new pixel P(x,y) know the four nearest points are Q11 = (x1, y1), Q12 = (x1, y2), Q21 = (x2, y1), and Q22 = (x2, y2).

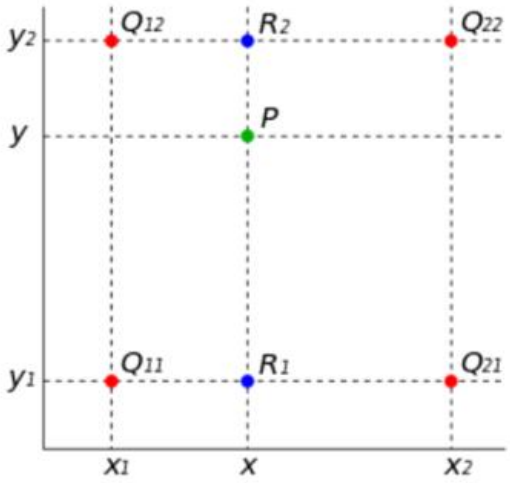

Fig 17. Bilinear interpolation for point P(x,y)

Step 1: Linear interpolation for the values at points R1 and R2

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \Big| \quad R_1 = (x, y_1)$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \Big| \quad R_2 = (x, y_2)$$

Step 2: From two points R1 and R2, linear interpolation for the value at point P

$$f(P) \approx \frac{y_2 - y}{x_2 - x_1} f(R_1) + \frac{y - y_1}{y_2 - x y_1} f(R_2)$$

➔ Bilinear interpolation method reduces image distortion when zoomed in, blurring image contours. This interpolation has high execution time and complexity. However, bilinear method gives good visual effect.

Loss function: Formally, during training, apart from two loss functions mentioned above, main additional loss function is loss mask function. The mask branch has a Km^2 - dimensional output for each RoI, which encodes K binary masks of resolution m × m, one for each of the K classes. To this we apply a per-pixel sigmoid and define L_{mask} as the average binary cross-entropy loss. For an RoI associated with ground-truth class k, L_{mask} is only defined on the k-th mask (other mask outputs do not contribute to the loss).

$$\mathcal{L}_{\text{mask}} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij} \log \hat{y}_{ij}^k + (1 - y_{ij}) \log(1 - \hat{y}_{ij}^k) \right]$$

Fig 18. Loss mask function

where y_{ij} is label of cell (i, j) and $\hat{y}_{ij}$ is predicted value of pixel(i,j) for per class.

Model implementation

Because our resource is limited, we only implement with google colab by using pre-trained Mask RCNN models (MS-COCO) to predict mask.

Model
mask_rcnn_resnet18_v1b_coco
mask_rcnn_fpn_resnet18_v1b_coco
mask_rcnn_resnet50_v1b_coco
mask_rcnn_fpn_resnet50_v1b_coco
mask_rcnn_resnet101_v1d_coco
mask_rcnn_fpn_resnet101_v1d_coco

Fig 19. The list of model for implementation

At bottom the first demo, overall models detect person mask very precisely. resnet50_v1b is quite noisy while fpn_resnet50_v1b, fpn_resnet101_v1d, resnet18_v1b are good.

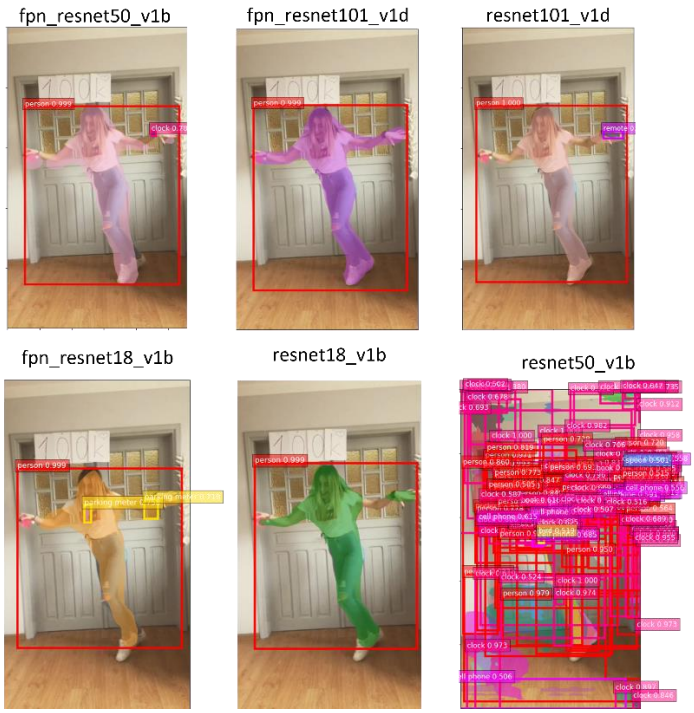

Fig 20. The result for the first demo

At bottom the second demo, overall models detect person (2 people) and car masks very exactly. Fpn_resnet_18_v1b detects frisbee that could not be important.

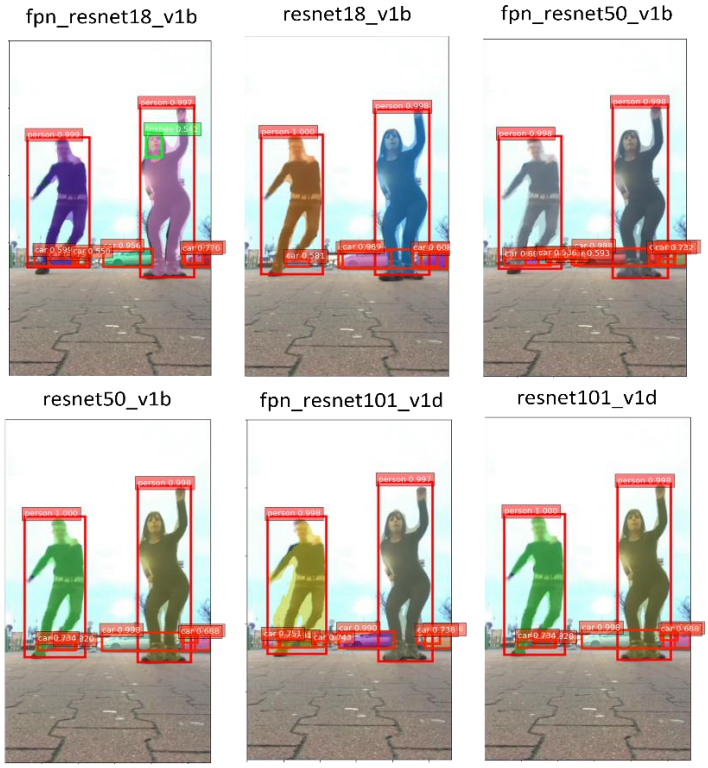

Fig 21. The result for the second demo

References

a. Squeeze U-Net: A Memory and Energy Efficient Image Segmentation Network
b. U-Net: Convolutional Networks for Biomedical Image Segmentation
c. Humans Image Segmentation with Unet using Tensorflow Keras
d. simple_multi_unet_model by Dr. Sreenivas Bhattiprolu
e. Demo Mask RCNN
f. He_Mask_R-CNN_ICCV_2017_paper.pdf
g. NGHIÊN CỨU VÀ ĐÁNH GIÁ CÁC PHƯƠNG PHÁP NỘI SUY ẢNH VIỄN THÁM CHO BÀI TOÁN PHÂN LOẠI LỚP PHỦ ĐÔ THỊ TẠI VIỆT NAM
h. Mask Region based Convolution Neural Networks - EXPLAINED!
i. TÌM HIỂU VỀ THUẬT TOÁN R-CNN, FAST R-CNN, FASTER R-CNN và MASK R-CNN
j. https://en.wikipedia.org/wiki/GrabCut
k. https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html
l. https://www.ipb.uni-bonn.de/html/teaching/photo12-2021/2021-pho1-12-segmentation.pptx.pdf
m. https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_clustering_algorithms_mean_shift.htm
n. https://developers.google.com/machine-learning/clustering/overview
o. https://www.kdnuggets.com/2019/08/introduction-image-segmentation-k-means-clustering.html