Locate Identity Card On Photo Using Semantic Segmentation with U-Net architecture

Nguyen Phuong Nam
Hanoi University of Science and Technology

Nguyen Hong Dang
Hanoi University of Science and Technology

RubikTALENT program final term project report

Abstract

Citizen Identity Card is the most important card of each citizen in Viet Nam, it contains basic information such as name, birthday, native place, permanent residence , etc. Automatic information extraction from identity card, which is useful for storing, restoring and transforming Government through digitization. Classical image processing approaches often require a complex extraction process, and only produce good results in some concrete cases as the card is set exact direction, and background is solid and contrast to the color of ID card, etc. In this research, we apply recent advances in deep learning to propose effective deep neural networks (CNNs, U-net), that can accurately locate ID card on the photo without manual processing image. Our target is a model which is robust and general enough for real life problem.

I. INTRODUCE

Today, Computer Vision, an AI technology that allows computers to understand and label images, is now used in convenience stores, driverless car testing, daily medical diagnostics, and in monitoring the health of crops and livestock. According to our research, we've found that many of the use cases of computer vision fall into the following clusters: Retail and Retail Security, Automotive, Healthcare, Agriculture, Banking, Industrial. In the past decades, the increase of of generally available computational power provided a helping hand for developing fast learning machines, whereas the Internet supplied an enormous amount of data for training. Among a lot of advanced machine learning techniques that have been developed so far, deep learning is widely considered as one of the most promising techniques to make AI machines approaching human-level intelligence. And one of the most AI fields is Computer Vision, deep learning is also expect techniques to create slam dunk in this one.

In this research, we adopt deep learning techniques combine with image processing. The base idea is to use a semantic segmentation network to predict the mask of ID card on a photo. Hence, background is removed, then apply some image processing technicals to process the mask and get the correct coordinate of 4 corner of ID card, which is much more easier. For the rotation problem, we use a small CNN to rotate the image back to the correct orientation.

For the rest of this paper is organized as follows. In section II, we briefly summarize some related work on locating ID identity card. In section III, we describe how to applying U-Net, image processing and CNN classification model in this problem. Our experiments and evaluation are shown in section IV. The conclusion is in section VI with some discussion for improving model in the future.

II. RELATED WORK

There are many solutions for locating citizen identity card on photos. Classical image processing approaches for this problem are based on the different of colors between ID card and other things in the HSV space, then apply for background subtraction, and bounding box card. This approach is good only if background color is quite solid, and the distinction between color of background and foreground is far enough to segment. Overall, this approach can't apply to a general system in real life.

With the growth of deep learning, the state-of-the-art in many computer vision tasks has been considerably apply. FPT also has a FVI project **[1]**. The structure of FVI comprises 3 basic components: cropper, detector, reader. In the cropper, they treated each corner of the ID card as an object, and detect the 4 corners of ID card. They using SSD (SSD: Single Shot MultiBox Detector), with the descriptive extractor is MobileNet v2 (MobileNetV2: Inverted Residuals và Linear Bottlenecks).

Also, there is an approach, treat the 4 corners as 4 key-points, and using a regression model to predict them.

In this paper, we propose applying semantic segmentation to find ID card on the photo, we use geometric transformations of images to crop the ID card out of the photo, and finally, we use classification models as SVMs, CNNs, etc, to rotate exactly direction. With machine learning, it has K-mean, mean shift, etc, can be used for segmentation, however they produce bad results, so we use U-net architecture for this purpose.

III. OUR PROPOSED APPROACH

Why U-Net?

The two mentioned approaches, as our knowledge, need a large amount of data, which is impossible for us to collect. While working with semantic segmentation problem, we saw that complex problem with many class (>20), any fully convolutional neural networks such as FCN, U-Net,.. work pretty well. So can we use the idea of semantic segmentation for the ID card problem? Additionally, U-Net can work well, also the case of small amount of data.

In this section, we will show how to apply U-net for semantic segmentation and CNNs for classification task. In the U-net, the input image size is resized 864 x 1536 to 16:9 aspect ratio, because this proportion is compatible with most of the smartphone's photo resolutions

3.1. U-net for semantic segmentation phase

3.1.1. Dataset

Since the contents on identity card are personal information, there are no public dataset for this problem. Fortunately, this problem can be solved with a dataset, such as many samples are taken from the same identity card, with different backgrounds and views.
We capture the identity card photos of about 20 people, try to capture with the different kind of backgrounds: simple one color, simple two color, complex with more color, very complex with many colors and many object corners; different kind of light conditions: daylight, lamplight, flashlight. Also, the rotation is randomly different.
The dataset contains only 263 images. We also use **labelme [2]** tool to label our dataset. Many thanks to the authors, the labelling work is more easier.

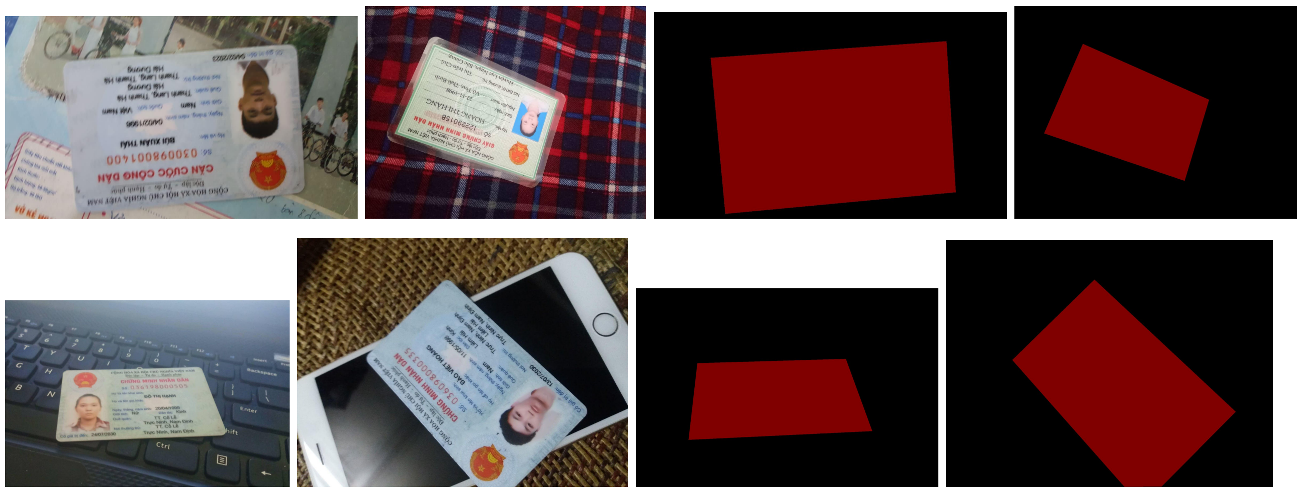


Fig. 1 : Examples of our dataset, images and corresponding labels

3.1.2. U-Net model architecture

The U-Net **[3]** was developed by Olaf Ronneberger et al. for Bio Medical Image Segmentation. The architecture contains two paths. First path is the contraction path (also called as the encoder) which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The second path is the symmetric expanding path (also called as the decoder) which is used to enable precise localization using transposed convolutions. Thus it is an end-to-end fully convolutional network (FCN), i.e. it only contains Convolutional layers and does not contain any Dense layers because of which it can accept image of any size.

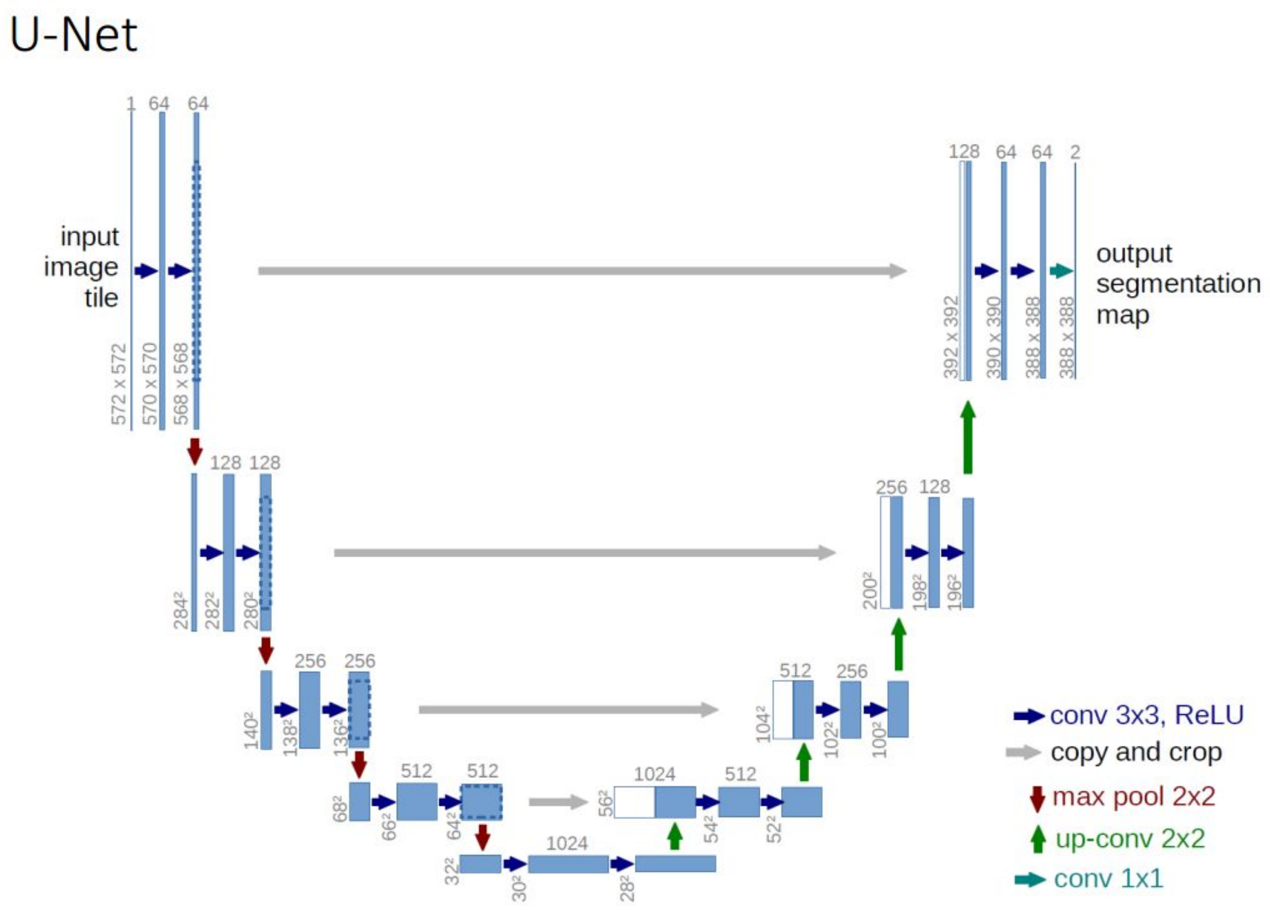


Fig. 2: Original U-Net model architecture

Our U-Net model comes with a little change, as described in Fig 3.

We use fine-tune method on our model, which uses a pretrained **VGG16 model with trained weights on ImageNet [4]** as the encoder portion of a U-Net and thus can benefit from the features already created in the model and only focus on learning the specific decoding features.

VGG16 comes with 5 max pooling layers, the image size is divided by 32 at the last feature extraction layer and become 27 x 48. While, the idea to keep the symmetry between encoder and decoder parts (the U-shape), we use block with 3 convolution layers

3 x 3 instead of 2 in some part of the decoder as the original version. Hence, the model is deeper, the number of skip connections between encoder and decoder layers is increased to 5. The feature pyramid on sample image, which is multi-resolution representation of an image, contribute to the final prediction of our model.

The last layer is convolution 1 x 1, is well known that it takes the same effect as using a Dense layer. But it is more robust this case, when our model become a fully convolutional network (FCN), which can take an image with difference size as input, but the constraint in our model is, width as height must be divided by 32.

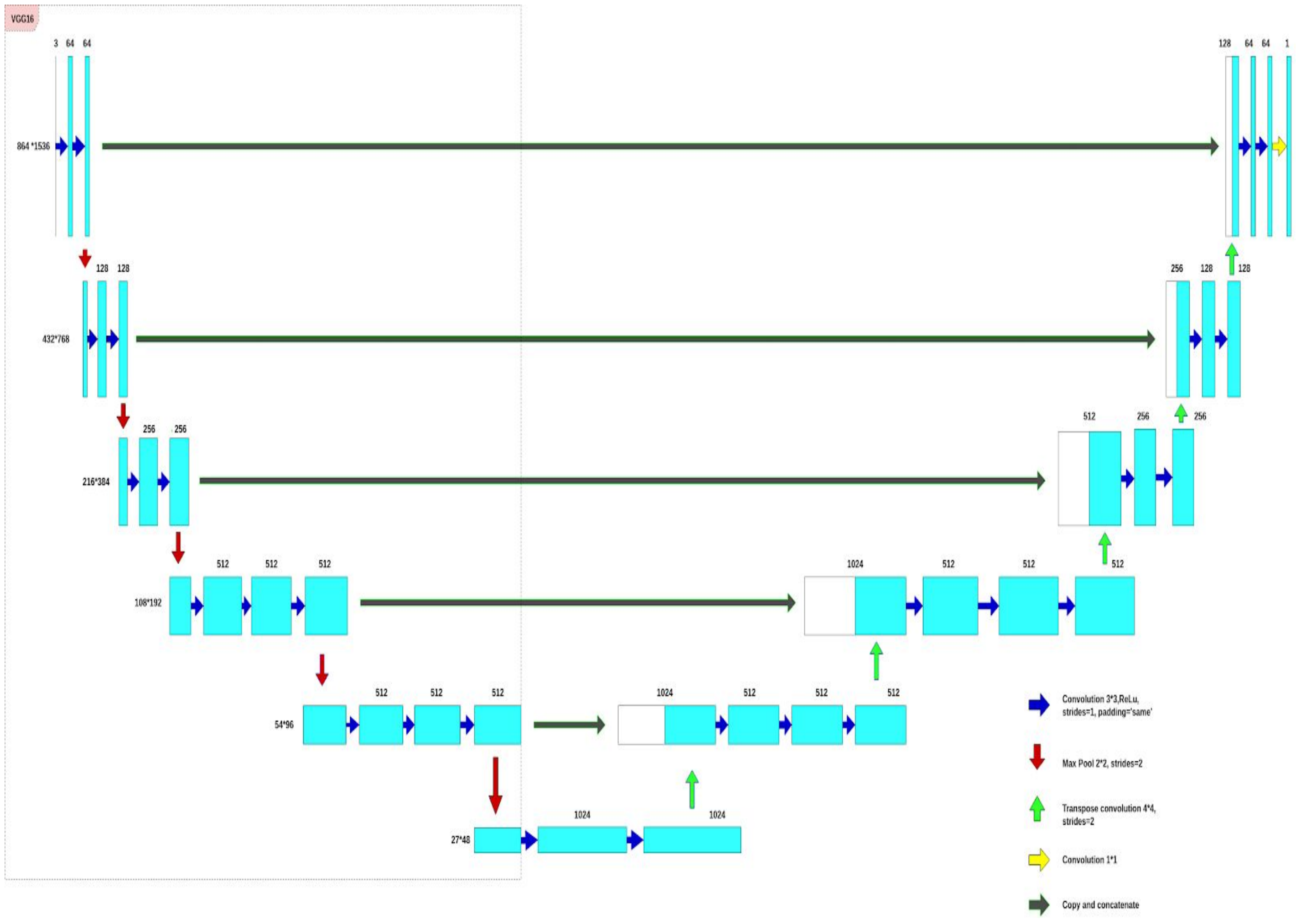


Fig. 3: Our U-Net model architecture

3.1.3. Losses and metrics

Semantic segmentation problem is a hard problem anyway, it is not so easy to evaluate/compare the results between difference approach/model. While working with this problem, not even the choice of the correct quality metric is trivial, let alone the choice of the best cost function.

For this problem, we propose using 2 kinds of loss function: cross entropy loss and Dice-coefficient loss.

The cross entropy loss, which is well known as the standard loss function for semantic segmentation problem:

$$\mathrm{CE}(p, \hat{p}) = -(p \log(\hat{p}) + (1-p)\log(1-\hat{p}))$$

The Dice coefficient is similar to the Jaccard Index (Intersection over Union, IoU):

$$\mathrm{DC} = \frac{2TP}{2TP + FP + FN} = \frac{2|X \cap Y|}{|X| + |Y|}$$

$$\mathrm{IoU} = \frac{TP}{TP + FP + FN} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

And the formula is:

$$\mathrm{DL}(\mathbf{p}, \hat{\mathbf{p}}) = \frac{2\langle \mathbf{p}, \hat{\mathbf{p}} \rangle}{\|\mathbf{p}\|_2^2 + \|\hat{\mathbf{p}}\|_2^2}$$

where $\mathbf{p} \in \{0,1\}^n$ and $\mathbf{0} \le \hat{\mathbf{p}} \le \mathbf{1}$.

One compelling reason for using cross-entropy over dice-coefficient or the similar IoU metric is that the gradients are nicer, while the dice-coefficient gradient sometime makes the training process become more unstable.

However, Dice coefficient performs better at class imbalanced problems by design. But, class imbalance is typically taken care of simply by assigning loss multipliers to each class, such that the network is highly disincentivized to simply ignore a class which appears infrequently.

We saw that in this problem, the classes are not imbalance, so, we use the standard way first, which is using binary cross entropy loss. The experiment shows that the dice-coefficient loss also performs pretty similarly to the cross entropy loss for this problem, as showing later in the next sections.

We also propose a weighted-loss method, as described in the improvement section. Because of the limited time, we can't try it, but it can be a way, or idea to improve the model performance later.

3.1.4. Training progress

We implement this model in Keras, which provide the faster and some auto optimizations for training deep learning model.

Since the size of image are typically difference, we provide some transformations to the original dataset samples to be fetched as input of U-Net:

- Step 1: If the image is vertical orientation, which mean that the height is higher than the width, rotate the image by 90 degrees clockwise to make the resize step below keep the aspect ratio stable.
- Step 2: Resize the image to 864 x 1536 as mentioned above.
- Step 3: Apply image augmentation technical to make model more robust and avoid overfitting.
- Step 4: Apply VGG16 preprocessing technical, provided by keras to get the best performance.

In step 3, we apply some image augmentations. Note that in the semantic segmentation problem, label is a mask image, and so it needs to be augmented too, with the same seed as image. So, fill mode need to be constant with value 0 to avoid bad labelling in the labels. The augmentations are:

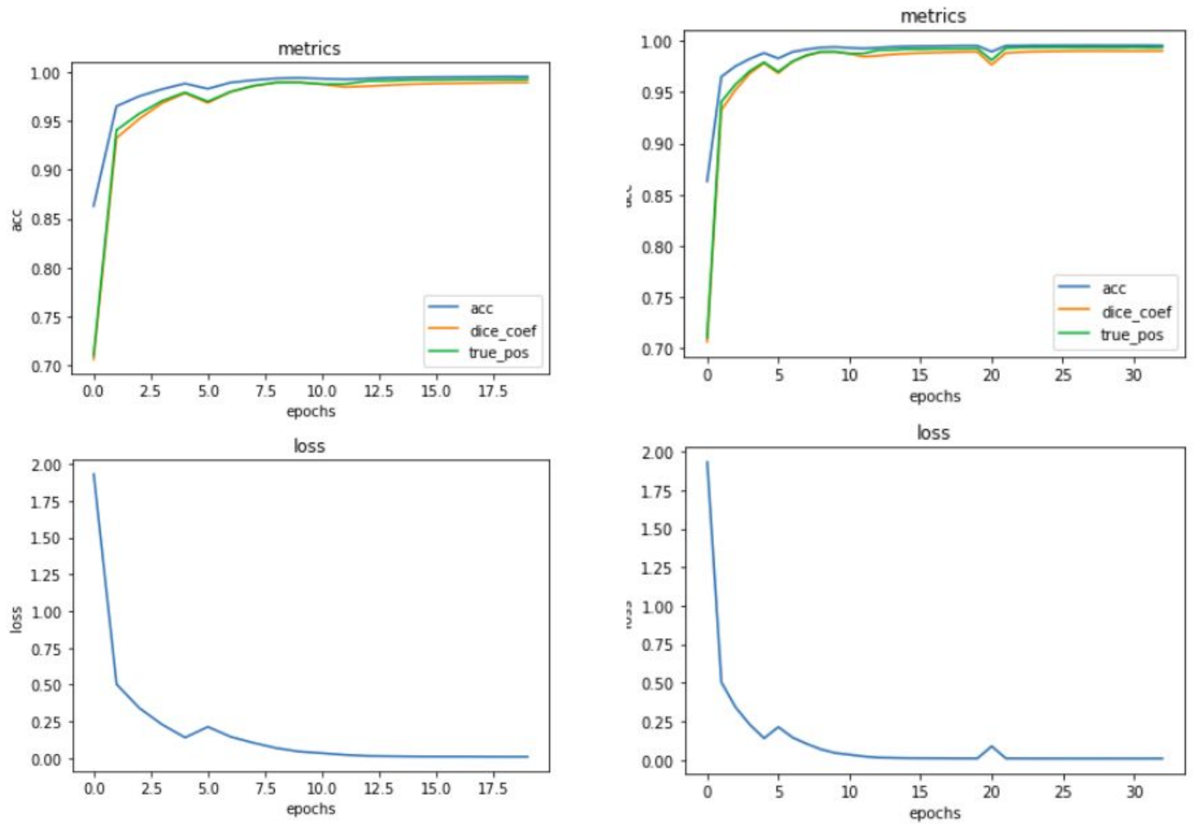
- Rotate a random degree in range [-30, 30]
- Width shift and height shift with factor 0.01
- Zoom with a random factor in range [0.9, 1.1]
- Change brightness with range [0.95, 1.05]

Fine-tune method include two main phase:

In phase 1, we frozend the VGG16 layers and only train on the decoder layers. We use Adam optimizer with learning rate 0.0001, reduce learning rate 10 times when the loss isn't improved for 2 epochs.

In phase 2, we unfrozen the VGG16 layers, and train on the whole layers. This time, we want to keep the weights learning from previous training phase, so we only use Adam optimizer with small learning rate 1e-6 and weight decay 1e-8.

In both phases, early stopping and model checkpoint callbacks are applied, with the monitoring metric is the loss value.



(a) *(b)*

Fig .4: Loss and metrics observed during training time: (4.a): phase 1; (4.b): all

It seems that with the frozen VGG16 layers, the VGG16 feature extraction is good, as we see the metrics quickly increase. The final result, we obtain the pixel accuracy 99.57%, dice coefficient 99.02% and the true positives rate 99.40% on training set.

3.2. CNNs and image processing phase

3.2.1. Image processing

Firstly, after getting the mask which output from semantic segmentation phase above we use image processing techniques to cut the ID card area from the photo. All steps comprises:

a. Morphological Transformations **[5]**: Erosion, Dilation, Opening

- Erosion: It is useful for removing small white noises, detach two connected objects, etc.
- Dilation: It is also useful in joining broken parts of an object.
- Opening: We use Opening which is just another name of erosion followed by dilation.

Because of the segmentation phase is not 100% accurate, there will be some noises in the output mask. The method above is very useful in removing noises. Some results are visualized as bellow:

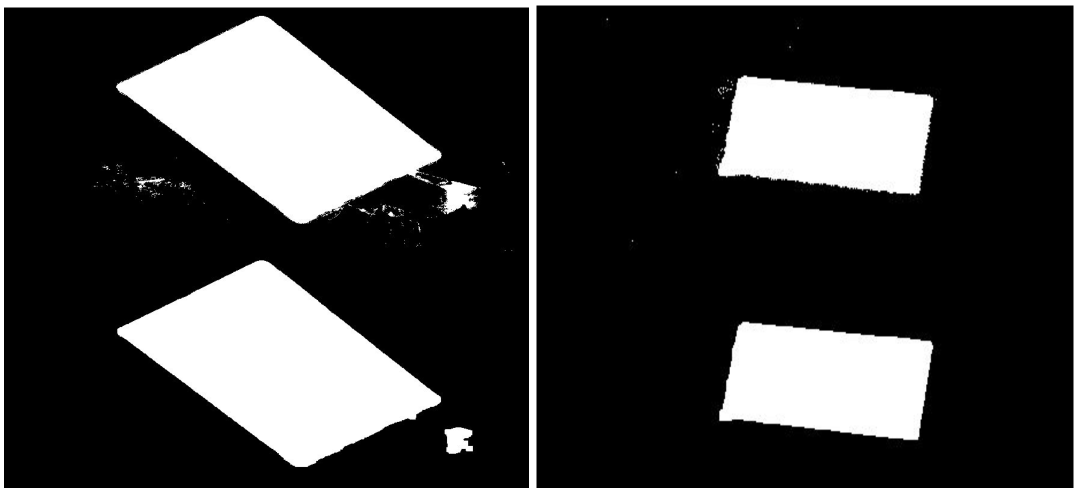
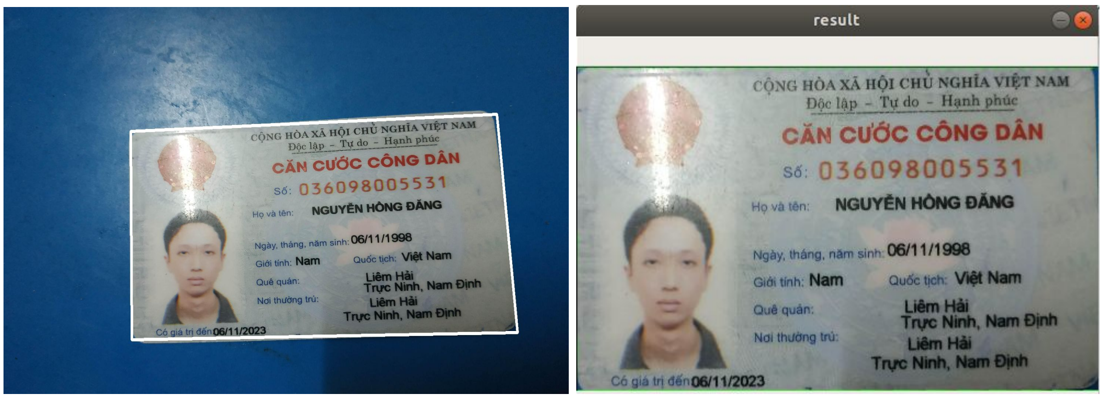


Fig. 5: Result after Opening

b. Blur image: we use kernel size = (15, 15). If the output of segmentation phase is good, we can use smaller kernel size to reduce the expensive computation.

c. Contours: Finding contours to bound the ID card area, then cut that area from image. When we get 4 points of input image corresponding points in output image, we apply geometric transformations of image to find transformation matrix.

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for edge detection, shape analysis and object detection.



CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc
CĂN CƯỚC CÔNG DÂN
Số: 036098005531
Họ và tên: NGUYỄN HỒNG ĐĂNG
Ngày, tháng, năm sinh: 06/11/1998
Giới tính: Nam Quốc tịch: Việt Nam
Quê quán: Liêm Hải Trực Ninh, Nam Định
Nơi thường trú: Liêm Hải Trực Ninh, Nam Định
Có giá trị đến 06/11/2023

Fig. 6: ID card area output

3.2.2. Prepare data for CNNs's training process

After the above step, we get the ID card area, however the ID card can be rotated in four directions. In order to solve this hindrance, we use CNNs to classify 4 specific classes corresponding to the direction of rotation. Because CNNs have ability to self learning the features extraction of images in order to classify categories. This is a basic task for CNNs and CNNs can be easy learning features to classify them. Although, we can use some classic machine learning algorithm with feature extraction to get such a high accuracy too.

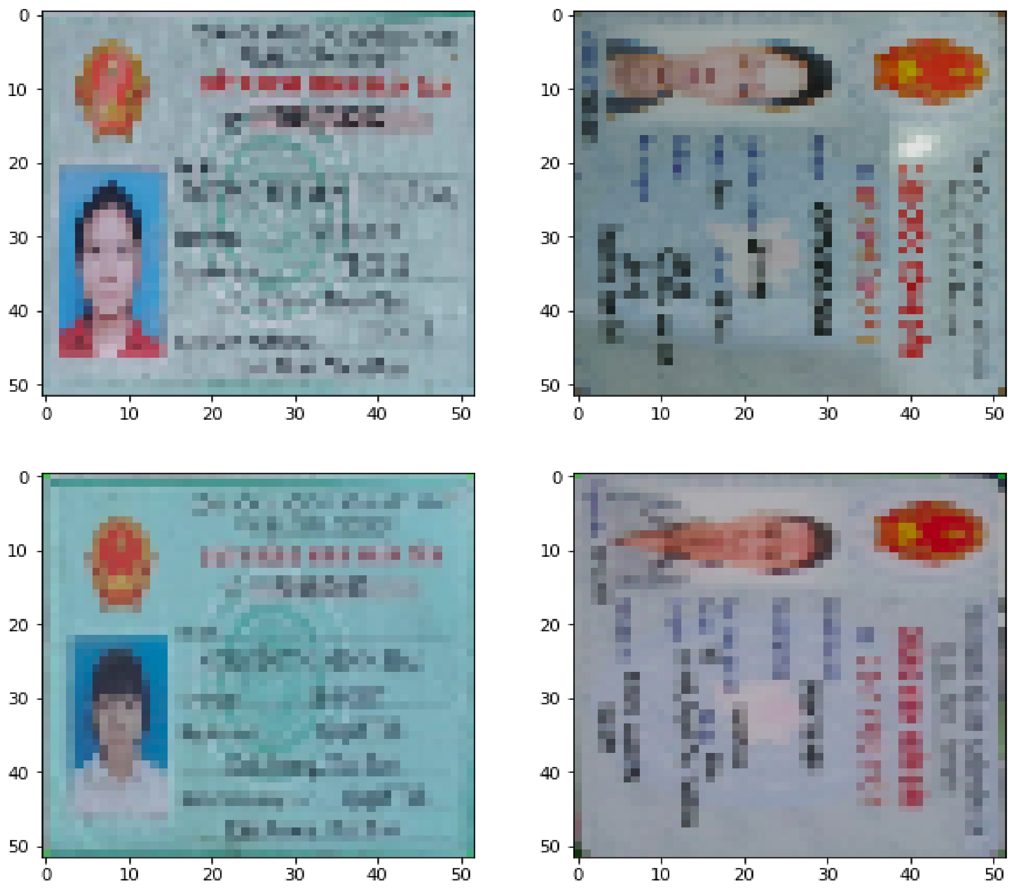


Fig.7 : Example of dataset for training CNN

We propose input image size should be 52x52, the resolution of the images can be decreased strongly, but that would be sufficient to keep enough clear information to classify. Our dataset for this classification task is 1452 images, about 363 image for each class in order to avoid imbalance data. We divide data set, 68% images for training set, 17% for validation set, and the rest 15% for test set.

a. The preprocessing step is quite simple: normalizing data to [0, 1] range, we divide each pixel value of over all images for 255.
b. Data augmentation: Due to the small amount of training data, we apply data augmentation techniques to increase the amount of training samples in order to avoid overfitting and improve classification accuracy. In our problem, for each image, we only add noises.

3.2.3. Classification CNNs architecture and training process

Input (52 x 52 x 3)
Conv5-32, s-1
Maxpool (2 x 2, s-2)
Batch Normalization
Conv3-32, s-1
Maxpool (2 x 2, s-2)
Batch Normalization
Dropout(0.5)
FC-32
FC-4

Fig. 7: Simple CNNs architect for rotation classification

This is a quite simple network, it only has 2 convolution layers and 1 dense layer. In training phase. Since we have a small data set to training in order to avoid network can be overfitting, we propose simple above architecture. Well, this network look like quite nice.

We train over 40 epochs and minimize the loss function by using Adam and back-propagation algorithm. We use training tricks such as reduce learning rate, early stopping, dropout with rate set 0.5 to avoid overfitting. The batch size is 64s. End the training process, the accuracy that model achieve is 99.3% in validation set and 100% in the test set.

IV. RESULTS ON OUR MODEL

Since our training set contains only 263 image, it can be a huge limitation for our model to be general and robust. Also, any evaluation can't be accurate enough for the real problem with several cases, different brightness, complex backgrounds and noises. Our result and evaluation only shows a perspective around this problem, with the idea of semantic segmentation with U-Net.

4.1. Result on U-Net model for semantic segmentation phase

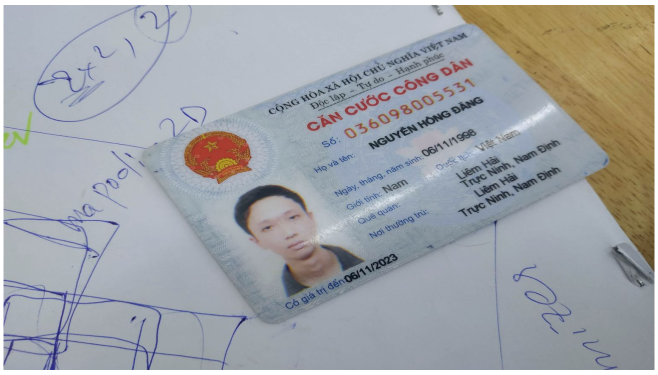
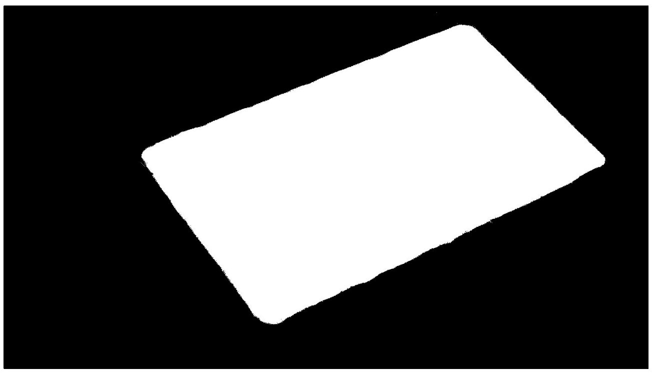
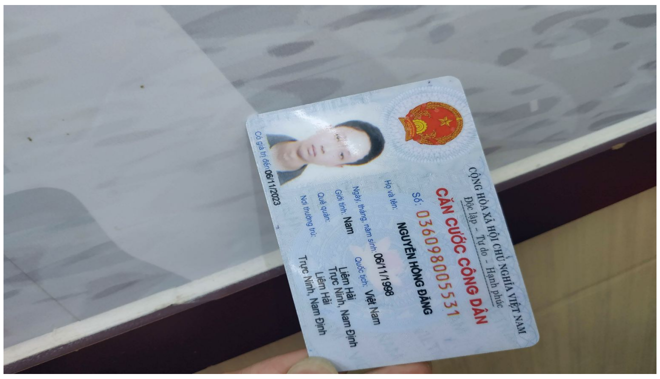
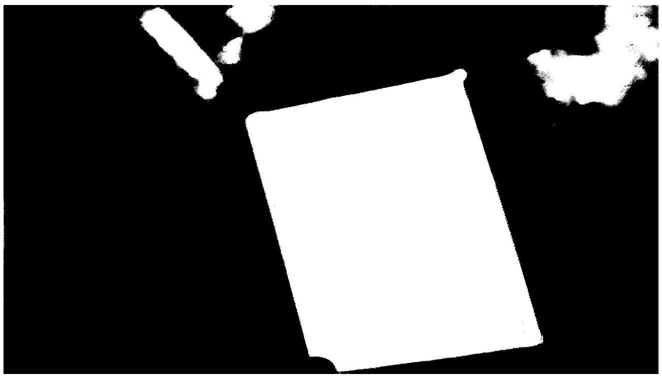
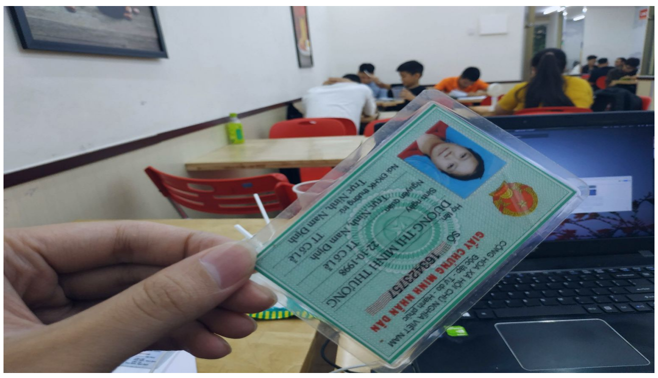
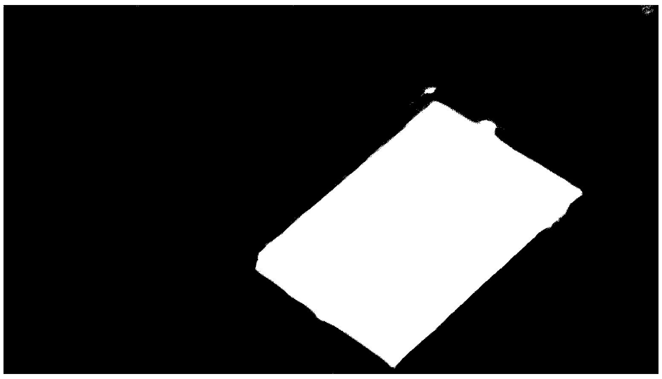
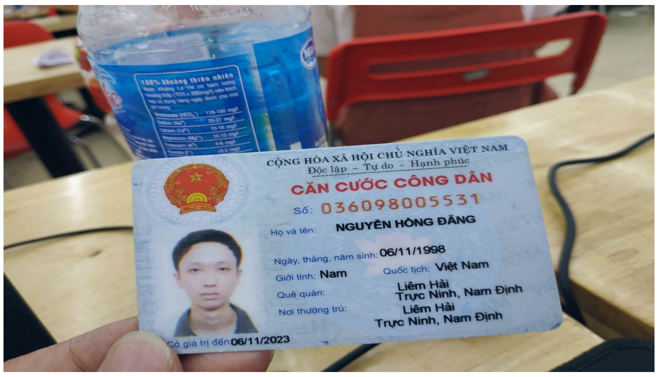
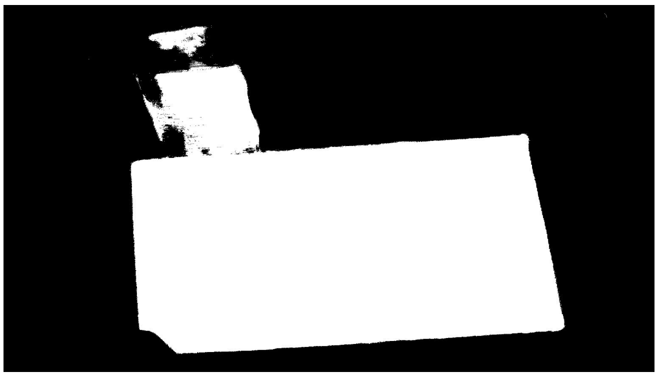
We test on a test set that comprise 53 images. As show in the training progress part, our model perform pretty good on training set. This may be caused by the overfitting problem when the model is very deep while the training set is too small and not general enough.

After apply augmentation technical to increase the test set size to 1000, we evaluate the performance on test set, and get this results:

- Accuracy over all pixels: 99.10%
- Dice coefficient: 98.51%
- True positives rate: 99.53%

The test show that our model give absolute accuracy on images with simple background with 1 or 2 colors. However, with complex background, it’s performance isn't very stable, but in general, it's good in several cases.

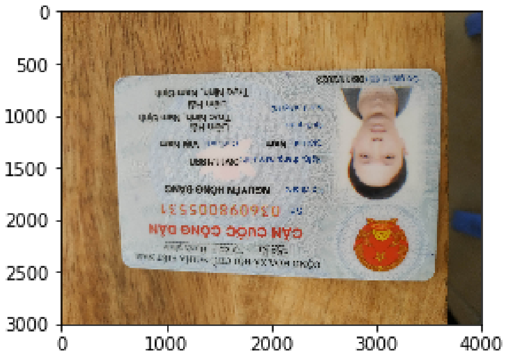
Some results on test set:

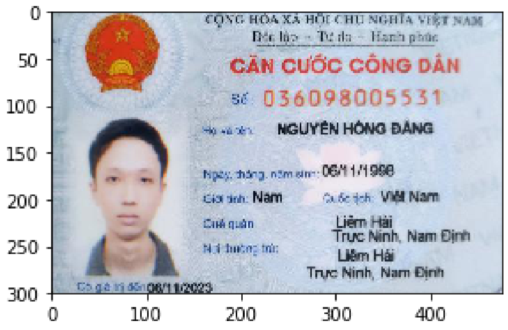









4.2. Result on all system

Our experiments show that this model seem to be not overfitting, and have a good performance. We try to locate ID card on 55 images, and our system produce good results as expected. However, the execution time is high. So, this approach can work only on a system which accept the time consuming. We tested in 55 images, the average time is 1.05s for each image with absolute accuracy.

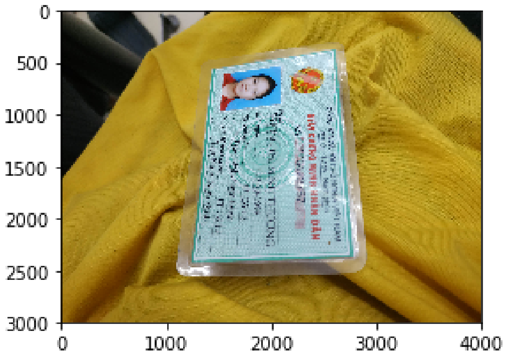
We use Google Colab with Tesla K80 GPU to perform all phases: build model, training, testing. The following images are results of model:

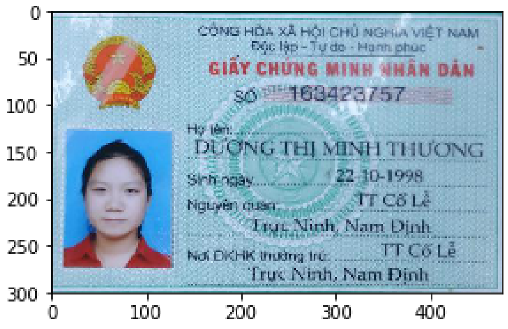




0
50
100
150
200
250
300
0
100
200
300
400

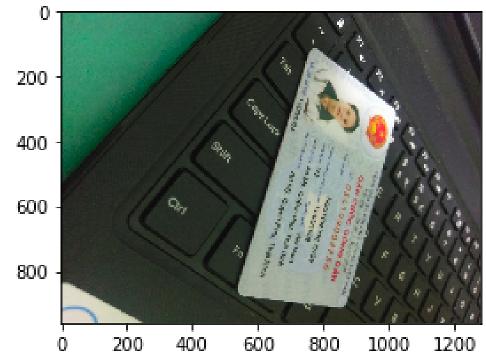
CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc
CĂN CƯỚC CÔNG DÂN
Số: 036098005531
Họ và tên: NGUYỄN HỒNG ĐĂNG
Ngày, tháng, năm sinh: 06/11/1998
Giới tính: Nam Quốc tịch: Việt Nam
Quê quán: Liêm Hải Trực Ninh, Nam Định
Nơi thường trú: Liêm Hải Trực Ninh, Nam Định
Có giá trị đến: 06/11/2023

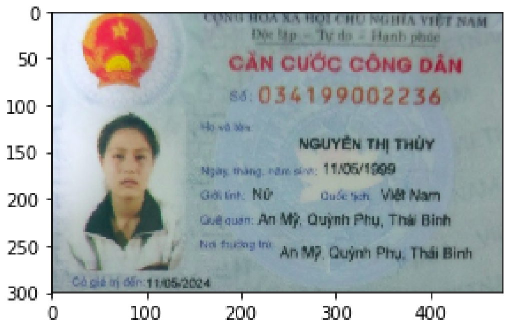




0
50
100
150
200
250
300
0
100
200
300
400

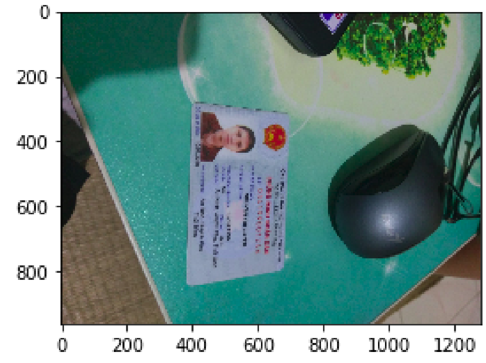
CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc
GIẤY CHỨNG MINH NHÂN DÂN
SỐ 163423757
Họ tên: DƯƠNG THỊ MINH THƯƠNG
Sinh ngày: 22-10-1998
Nguyên quán: TT Cổ Lễ Trực Ninh, Nam Định
Nơi ĐKHK thường trú: TT Cổ Lễ Trực Ninh, Nam Định

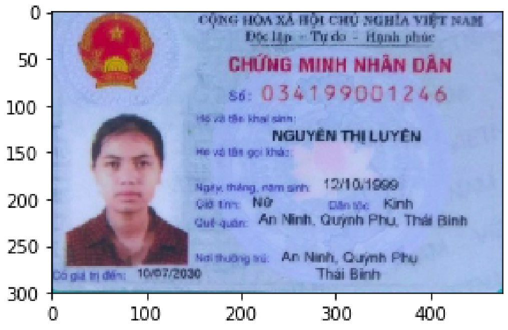




0
50
100
150
200
250
300
0
100
200
300
400

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc
CĂN CƯỚC CÔNG DÂN
Số: 034199002236
Họ và tên: NGUYỄN THỊ THỦY
Ngày, tháng, năm sinh: 11/05/1999
Giới tính: Nữ Quốc tịch: Việt Nam
Quê quán: An Mỹ, Quỳnh Phụ, Thái Bình
Nơi thường trú: An Mỹ, Quỳnh Phụ, Thái Bình
Có giá trị đến: 11/05/2024





0
50
100
150
200
250
300
0
100
200
300
400

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc
CHỨNG MINH NHÂN DÂN
Số: 034199001246
Họ và tên khai sinh: NGUYỄN THỊ LUYẾN
Họ và tên gọi khác:
Ngày, tháng, năm sinh: 12/10/1999
Giới tính: Nữ Dân tộc: Kinh
Quê quán: An Ninh, Quỳnh Phụ, Thái Bình
Nơi thường trú: An Ninh, Quỳnh Phụ Thái Bình
Có giá trị đến: 10/07/2030

V. CONCLUSION AND FUTURE WORK

This research is only a perspective for our problem. We try to use U-Net with the target that the model can be general and robust. The experiment shows that it can be a solution, however the execution time is high. But, the model performs such an impression with some images with complex background and noisy. The limited time we work on this project, and the limited dataset, we propose some future works, which may, or may not improve our model performance as below:

5.1. Using weighted loss

As in the original paper of U-Net, the author use weighted-loss to penalizes at each pixel by a computed weight map for each sample. Mean that, each pixel loss contribute a different importance to the final result, some pixels are need to be very well classified.
Our method, the suitable metric is very important, because our model need not to perform accurate on all pixels, the image processing after that will do the correct things. For example, the output mask produced by U-Net can be small noisy in background area, or some region inside the ID card area can be miss classified, but with the image processing, we still get the correct result.

A suitable weight map can be used, such as pixels around the edge of ID card, is higher than the other. Hence, it contributes more and more to the model loss, and it forces the model to perform high accurate on these pixels.

5.2. Object detection combination

In the case the background is very complex and noisy, our result with U-Net can be bad anyway. A simple idea is first doing the object detection to detect the region of interest, then perform segmentation on only that region. With more data, we can try to use Mask-RCNN for the combination of object detection and semantic segmentation.

VI. REFERENCES

[1] Technical views about the OCR for Vietnam ID card information extraction
https://techinsight.com.vn/goc-nhin-cong-nghe-ve-fvi-he-thong-ocr-nhan-dien-chung-minh-thu-nhan-dan-viet-nam/

[2] Ketaro Wada. labelme: Image Polygonal Annotation with Python
https://github.com/wkentaro/labelme

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation

https://arxiv.org/pdf/1505.04597.pdf

[4] Karen Simonyan, Andrew Zisserman. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION
https://arxiv.org/pdf/1409.1556.pdf

[5] Morphological Image Processing
https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm