이학박사 학위논문

Language classification of natural scene text image patches

(일반적인 문자 이미지의 언어분류)

2021년 2월

서울대학교 대학원

협동과정 계산과학전공

장 필 훈

Language Classification of Natural Scene Text Image Patches

(일반적인 문자이미지의 언어분류)

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2020년 11월

서울대학교 대학원
협동과정 계산과학전공
장필훈

장필훈의 이학박사 학위논문을 인준함

2020년 11월

위 원 장 국 웅 (인)

부위원장 강 명주 (인)

위 원 김 용대 (인)

위 원 곽 지훈 (인)

위 원 이병준 (인)

Language classification of natural scene text image patches

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Pilhoon Jang

Dissertation Director : Professor Myungjoo Kang

Department of Computational Science and Technology
Seoul National University

February 2021

Abstract

As other machine learning fields, there has been a lot of progress in text detection and recognition to obtain text information contained in images since the deep learning era. When multiple languages are mixed in the image, the process of recognition typically goes through a detection, language classification and recognition. This dissertation aims to classify languages of image patches which are the results of text detection. As far as we know, there are no prior research exactly targeting language classification of images. So we started from basic backbone networks that are used commonly in many other general object detection fields. With a ResNeSt-based network which is based on Resnet and automated pre-processing of ground-truth data to improve classification performance, we can achieve state of the art record of this task with a public benchmark dataset.

Keywords: deep learning, optical character detection, character recognition, multi-language image patch, classification, image processing
Student Number: 2013-23012

Contents

Chapter 1

Introduction

1.1 Optical Character Recognition

Text recognition is one part of pattern recognition, which aims to recognize characters on images automatically. Here, the images may not only include easily readable texts like newspapers or books but also design letters or handwritings printed on signs, advertisements, clothes, etc. They also include changes in the light contained in the image with letters, curvature of the surface, and partial occlusion. These various objects can be captured by 2 or more(2 is for black and white, and 3 is for colored. Usually 3 colors(red, green, blue) compose a pixel, but 4 is possible(CMYK). If transparency added, there are 4 channels.) dimensional digital image and every pixel's capacity can vary from 2 to 256 ordinarily. There can be infinite methods to represent an image as a bunch of numbers, and numerous ways are. In brief, the whole process of text recognition is to extract character information from those various images and to accurately output the text it contains.

In a variety of situations, humans do not have much difficulty in recognizing characters, but the recognition ability of machine is largely inferior to humans even recently. However, as with other fields of machine learning, neural networks began to be applied to image analysis, showing remarkable improvement in accuracy. As many networks have been released and applied to many tasks in recent years, we could test and modify some well-known nets like vgg[1], resnet[2], ResNeSt[3], ResNeXt[4] to this task.

1.2 Deep Learning

Perceptron(made of neurons) and the backpropagation technic, which appeared in the 1960s, have made breakthrough performance in tandem with the rapidly developing hardware in the 2000s. Before that, there was no hardware capable of calculating the weights of tens of millions of neurons quickly enough. From when it became possible, multilayered countless perceptrons can be used in various combinations. We call the technic as deep learning.

Multi-layer perceptron is a stack of several layers of perceptrons with activation functions. Linear perceptron is just like a matrix transformation (multiplication), it is the same as an arbitrary matrix that takes a certain matrix $\mathbf{A}$ as an input and converts it into a matrix $\mathbf{B}$. By applying the activation function to this, non-linearity can be obtained and the multi-layer perceptron can separate a data set that cannot be linearly separated. Let the activation function is σ and the input matrix is $\mathbf{A}$, the classifier consisting of one layer can be formulated as follows.

$$\mathbf{B} = \sigma(\mathbf{MA})$$

where $\mathbf{M}$ is a matrix. Transformation can be simply represented as a function. So, more generally the above can be expressed as follows.

$$\mathbf{B} = \sigma(f(\mathbf{A}))$$

When stacked in several layers, it becomes:

$$\mathbf{B}' = \sigma(\cdots\sigma(h(\sigma(g(\sigma(f(\mathbf{A}))))))))$$

As for the activation function, nonlinear functions that can be differentiated can be used. A tanh or sigmoid function is usually used. A sigmoid function is as follows

$$y = \frac{1}{1 + e^{-x}}$$

By stacking layers very deeply, these whole transformation can learn abstractions that were previously thought to be almost impossible to be learned, and the term 'deep learning' comes from this deep stacking.

Machine learning(especially, the pattern recognition) is substantially a task to make computer to perform abstraction from various conditions automatically. The deep learning shows overwhelming performance in pattern recognition, which is not different from the abstraction.

Until recently, deep learning has shown the best performance, and it will continue for a while. So we used the technic for this task.

Chapter 2

Backgrounds

2.1 Detection

Detection refers to specifying the position of a character or a word on an image. As a method of detection, drawing a **bounding box** is the most widely used. The box is represented by four(x,y,w,h) or five(x,y,w,h,theta) coordinates where x and y are Cartesian coordinates, w is width, h is height, theta is an angle of rotation. The box may go out of the image, and the part outside of the image is usually excluded from the evaluation. A more detailed format than a rotated bounding box is a quadrilateral, represented by four x,y coordinates. The main difference from the previous methods is that they can be trapezoidal. Quadrilaterals, however, can't represent texts with severe rotation or deformation, such as design letters or trademark letters(like Starbucks logo). Such forms are represented by a polygon. How many points can be used depends on how the detector is designed. Usually the detector gives 4, 8, 12, 14, 16 points.

There is an another form of output called **mask** which is more abundant of information than the above methods. While polygons inevitably have the disadvantage of including some non-letter areas, masks can accurately depict only the letters' area. Though this is not a perfect way because the boundary between the text and the background is often ambiguous, but it provides much more detailed information than polygons, rectangles, and bounding boxes. Among the public datasets, Totaltext[5] provides this data(Figure 2.1). And many other detectors have improved performance remarkably by processing ground truths in the form of bounding boxes into masks. For example, by reducing the size of the bounding box or making a map according

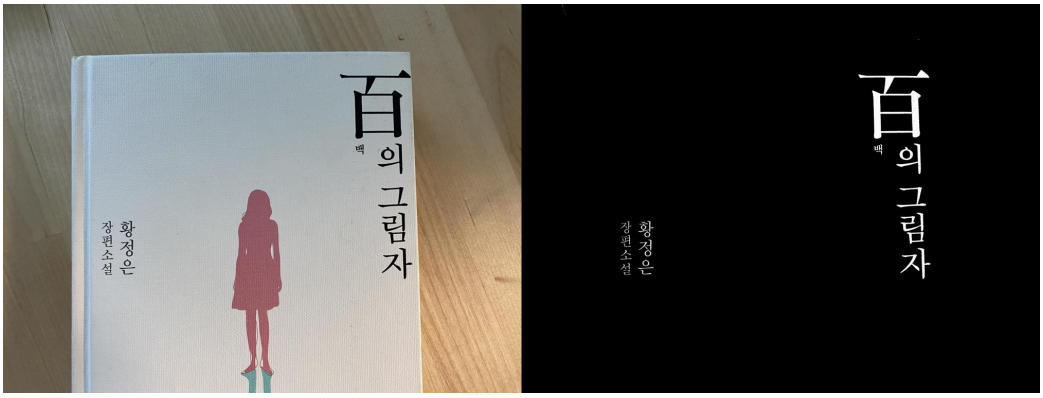

Figure 2.1: Example of mask ground truth. In the image on the right, white pixels represents text area of left raw image.

to the number of characters, ground truths can contain as much information as possible like Figure 2.2

2.2 Recognition

Recognition is a process after detection, and it is a classifier to classify characters that are detected by a detector. Therefore there are as many outputs as classifiable characters. The input format of the recognizer is determined by the way of the detector giving the input image. Concretely bounding boxes remains as it is(because a bounding box can produce an image patch without any modification), and rotated bounding boxes can be handled easily(just like bounding boxes' cases only with the rotation). In the case of rectangles or quadrilaterals, there are various treatments and many researches how to stretch or reform an arbitrary shaped polygon to a rectagle. One of the most common ways is taking a bounding box or `minarearect`[1] covering all vertexes and treating it as a general rectangle. As far as we know, a method of putting the mask output as the input of the recognizer with little modification has not been studied.

After proper processing, due to the nature of the character notation, input images are usually horizontally long or vertically(in the case of vertical writing) long, which enters to the recognizer as an input. Because most recognizers take images of fixed sized as an input, too much deformation decreases

[1] An OpenCV[7] function to get a rectangle coordinates that covers all input points.

Figure 2.2: An example showing the Craft[6]'s idea of processing a ground truth. A bounding box is transformed to a pseudo heatmap(stretched 2 dimensional gaussian, not generated by a heat equation.)

recognition accuracy. So additional processing is required, such as cutting the input image into several pieces of appropriate size and giving them multiple times to the recognizer. In this dissertation, we tried 'divide and stack' method and will explain it in Chapter 3.2.2 in detail. To compensate for this shortcoming, recognizers that receives a variable sized image using recurrent neural networks(RNN) are becoming mainstream. The result passing through the recognizer becomes the final texts of character recognition.

Recently, methods of binding both detection and recognition into one have been attempted, and these end-to-end methods are showing good performance. More details are in Chapter 2.7.3.

2.3 Language Classification

Language classification is placed between the detector and the recognizer. As mentioned in 2.2, this classifier is not used for the end-to-end recognizer, and is placed between the detector and the recognizer when they are separated in traditional form.

When detection or recognition is for an image composed of only one language, we do not need this process. But usually two or more languages are mixed, and Latin characters exists regardless of the language. In addition, even if only one language is targeted, it is necessary to determine whether the detected character is recognizable or not, so at this time the language

classification should be used to prevent false positives. This research focuses only on this process.

2.4 Multi-lingual Text(MLT)

As far as we know, no single study for the 2.3's language classification has been preceded. This study solves by applying the techniques well-known for general object detection problem. For the comparison of performance, the most recent version of ICDAR 2019[8] data, which is the most widely(and maybe only) used data for this task, is used and the performance will be compared with the records of the official leaderboard.

Rank	**Method**	**Accuracy**
1	Tencent-DPPR Team	94.03%
2	SOT: CNN-based Classifier	91.66%
3	GSPA_HUST	91.02%
3	SCUT-DLVC-Lab	90.97%

Table 2.1: Highest accuracies of MLT19 task2 on the official leaderboard. The same rank means there are meaningless difference in performance despite the difference in records. This is due to the faulty ground truths, and was decided by the task organizers(ICDAR committee). *Note* Reprinted from [8] table II.

2.5 Convolutional Neural Network(CNN)

There are many ways to apply the perceptron described 1.2 to an image. In the simplest case, perceptron can be arranged for each pixel and calculated for every pixel, which is called a fully connected network(FCN). Since the pixel number of an image is usually tens of millions, it is rare to use FCN directly to the input. Instead, a method of creating a smaller filter than the image size and sliding the entire image is used. In this case, since con-

volution operation[2] is used as the basis, it is called a convolutional neural network(CNN) and is the de facto standard for networks targeting images. This method scans the entire image with a filter (usually 3 dimensions in the case of an image. The convolution operation can be used not only for images, but also for sounds.) and performs calculations. There is a design insight to grasp the characteristics of the whole image as a single consistent criterion among various input values by sharing the filters' weights. The individual numbers passing through the filter are gradually stepped up to form the condensed and refined information. Although the number of parameters has been drastically reduced compared to that of multi-layered perceptron, CNN yet has a considerable amount of computation. So the research has been slow until the hardware is sufficiently developed. In the past 20 years, the cost of computation has been dropped dramatically(because of population of cheap GPUs and improvement of performance), and the related researches began to pour out.

2.6 Attention Mechanism

Attention mechanism is basically used on sequence to sequence learning. The idea is to provide a context information to the decoder for every inference. The attention weight which is emphasizing important parts is given to make the network focus on more valuable information. It is known that Bahdanau[9] first proposed the concept of attention, and it is being applied a lot to the vision field, which is also called visual attention. The attention mechanism can be applied to almost all network structures that emphasize only a part rather than the entire input, and it gives good results.

Resnest[3] is an application of the attention to the Resnet[2]. This research uses this network as one of backbone networks, and the Resnet and Resnest will be explained at 3.1.3, 3.1.6.

[2]The convolution operation is a Cauchy product.

$$\left(\sum_{i=0}^{\inf} a_i\right) \cdot \left(\sum_{j=0}^{\inf} b_j\right) = \sum_{k=0}^{\inf} c_k \quad \text{where} \quad c_k = \sum_{l=0}^{k} a_l b_{k-l}$$

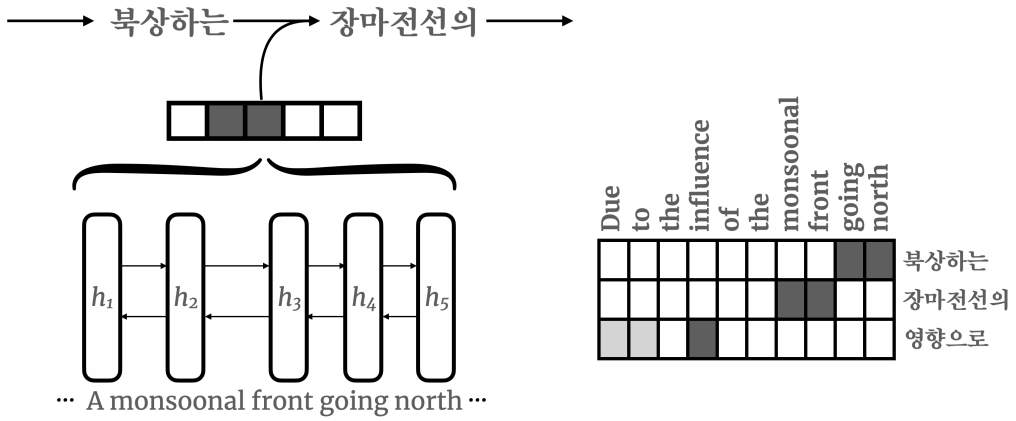

Figure 2.3: The graphical illustration of Bahdanau model. h_i represents latent vectors which are encoded outputs of input words. The output words are generated sequentially according to the attention weights and preceding word's encoded vector. Attention weights let the network know where it should focus when producing the next translated word. Right picture shows the stacked attention weight used when every each word is translated. We can see the relationship between the words which have the similar meanings.

2.7 Related Works

From this section, we will consider only studies since the emergence of deep learning. Previous studies are currently rarely used. Since there are no prior studies focusing only to MLT(Multi-lingual Text), OCR studies that are necessary for understanding this study will be explained.

2.7.1 Detectors

Regression based methods

R-CNN(Regions with CNN features.[10]) has been applied to a wide range of fields since the release. Above all, it performance is overwhelming and it converges relatively well (It is common for some networks too hard to converge). First, ROI(Region of interests)s which can be considered as a kind of candidates are extracted by the RPN(Region Proposal Network). ROIs come out in the form of bounding boxes. Classification is performed for each of these boxes through the ROI pooling process to produce the final

result. ROI pooling means resizing regoins to a certain size and putting them to the classifier. If the RPN is designed to output rotated boxes, the outputs must be rotated before getting into the classifier to increase the accuracy[11]. There are many ways to deal with overlapping boxes, but traditionally NMS(Non-maximum Suppression) is used. The structure is as Fig 2.4.

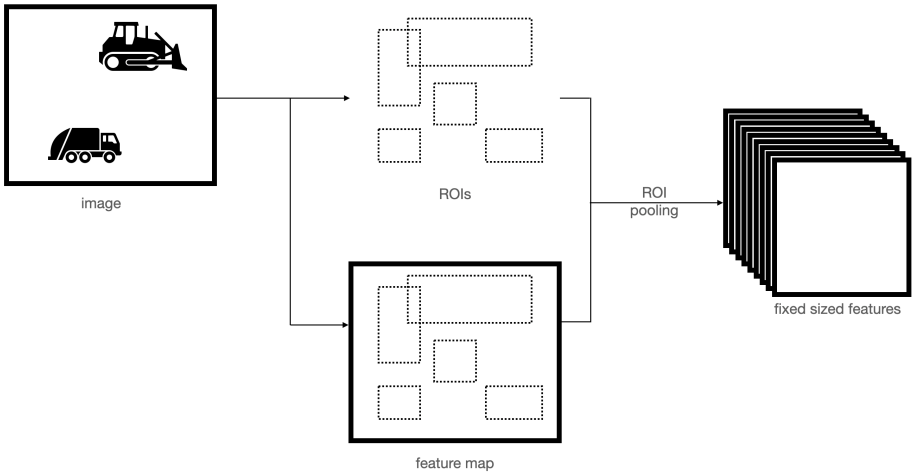

Figure 2.4: ROI pooling. A feature map is extracted for the whole image. When the RPN network outputs ROIs, feature maps for each ROI are extracted from the shared feature map, and it is cut according to coordinates of bounding boxes as if the shared feature map is an image. And then, each feature maps are resized to a fixed size.

This is the same when applied to detection, except that the classification is much simpler. In general object detection such as Imagenet[12], the number of classes is large. In the case of character detection, only two classes are required(the positive and the negative. So this stage can be replaced by the logistic regression.), so the number of parameters in the output layer of the net is small. However, in the case of recognition, as the number of characters increases. So thousands or more classes are possible. We cannot get a high-performance detector just by adjusting the number of output channels. Currently well-working detectors of R-CNN base are usually attached together with a recognizer when training(this is called an 'end-to-end' method) or obtain a result with R-CNN first, then post-processing for higher

accuracy. As far as we know a high-performance detector made of pure R-CNN only does not exist. Researches about end-to-end training is covered in 2.7.3 below.

[13] and [14] are well-known detectors of R-CNN style. Both get boxes and then refine them. In the case of the traditional R-CNN, only the overlapping boxes are filtered by NMS. But in the case of [13], segmentation is also performed on the resulting bounding box, and the confidence value is determined as a final result. If the segmentation result is not good, the box is also removed. In the case of [14], the shape of a box is refined several times using LSTM[15] to obtain an arbitrary shaped box. It decreases the distance of the upper and lower points by moving little by little from left to right. At this time, the coarse ROIs can be filtered in that refinement process even they are obtained from a relatively inferior light-weight net. So the size of RPN an be small. And moreover, unlike other detectors, it is possible to obtain an arbitrary box without any masks.

Because of the similarity between structures, nets such as LOMO[16] that apply the same principle to box regression can be classified into this category. But the ROI pooling is also a key feature of the R-CNN family, so LOMO should be classified as segmentation method. The difference between the two is whether the anchor is partially used with the process like `topk`[3] or confidence scores or used without any drop of candidates. More specifically, as shown in the rcnn figure(Fig. 2.4), ROIs are collected and resized to a certain size, then filtered. Without this process, the number of filters (channels) in CNN can be used to represent the coordinates of boxes, and the rotation angle of them for every pixel in the image. For example, if there are five channels, each channel represents x-coordinate, y-coordinate, width, height, and a rotation angle. In this way, we do not need ROI pooling process even with the principle of using anchors.

Segmentation based methods

Many detectors use the segmentation method, and most modifications for performance improvement are made on how to get accurate box coordinates. There are mostly two ways to obtain box coordinates, one is to configure the output layer to produce coordinates itself, and the other is to classify into two categories(internal and external of a box), and then process only the

[3] A function that returns largest(or smallest) `k` elements of the input. Almost all deep learning libraries offer this function.

points inside the box appropriately to draw a box. The latter structure is to obtain a kind of map using CNN, and the result is a mask. We can think of it as performing pixel classification.

EAST[17] is maybe the simplest and best performing detector of former type. As the first image passes through FCN(Fully Connected Network), the box coordinates followed by NMS for filtering noise are produced. A peculiar point is that the format of the coordinates is not only x,y,w,h, but also four arbitrary points(8 numbers), so that the network can output arbitrary rectangles even after NMS. YOLO[18] is similar, but consists only of the CNN layers at the front and the fully connected layer of the output. The classes and bounding boxes are predicted for each channel. After going through the NMS, final results are obtained. The performance is not so good as the fast(er) rcnn series, but it has a simple structure and on par performance, so its application range is wide. Textboxes++[19] has the same structure, but the output contains an additional channel for rotation angle. Therefore, rotated bounding boxes can be obtained.

A representative case of the second kind is LOMO[16], which was described earlier. And DB[20] is implemented with the idea of learning thresholds that are indispensable in the process of segmentation. In most cases, whether each pixel corresponds to text area or not will eventually decides using a scalar value, which is usually set to an intermediate value. But in order to implement a high-performance detector, the best value is experimentally found and fixed. However, in the case of DB, there is a mechanism that can make the net learn the threshold itself. TextCohesion[21] determines whether a pixel belongs to a text area by creating multiple segmentation maps and voting them. In the case of PSEnet[22], the output layer gives the disk size that can be drawn at that location for every pixels. The output has multiple channels and each channel represents the disk sizes of that point. For example, the channel representing the smallest disk size make out the part that crosses the center of the text, and the channel representing the largest disk wraps the entire text region. This helps to solve the chronic and typical problem that several lines are easily clustered together during the segmentation process. This usually occurs when the text is small or when various sized texts are attached. Considering that there is a detector as a preliminary step before the recognizer, the clumped text will hardly be recognized and thus has no meaning of detection. However, in the case of PSEnet, the region is expanded based on the channel predicting the center line of the text area, so even if the boxes are clustered in the last channel(predicting largest

disks channel), they can be distinguished. Another form is Textsnake[23]. The Textsnake gets the disk radius and center point in the fully connected network layer, and then cluster them to shape an arbitrary text area. It does not use a box, but produces a lot of disks, so it's essentially not a different process with NMS in case of boxes. However, by obtaining the center point, it is easier to separate text areas as in PSEnet above.

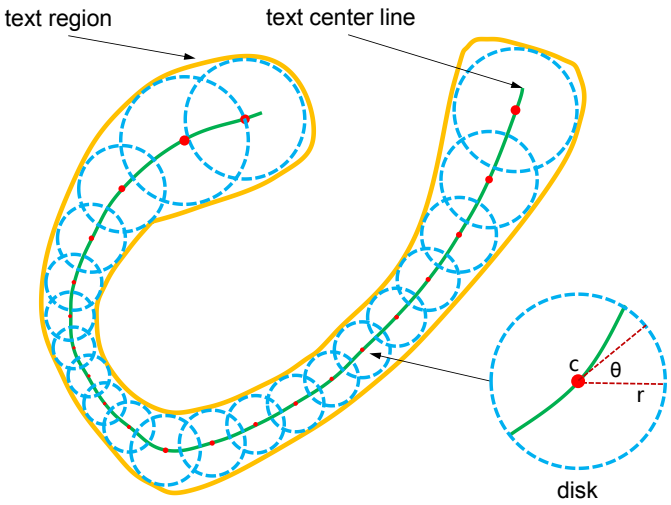

Figure 2.5: The basic idea of Textsnake which is one of the segmentation methods. The text region is constructed based on the text center line and the predicted radius at that line. *Note.* Reprinted from Textsnake[23]

Another common way is to use *link*s. The most representative one is Seglink[24]. First, candidate boxes are obtained using several CNN layers. It is characteristic that these candidate boxes do not appear by character, word, or line units. This means that even 1.5 characters can exist in a box. The idea is to detect a long box using the connection information between the boxes regardless of character units in each boxes. This connection information is also obtained through the previous layers. Pixellink[25] also uses the idea of links. However, as the name suggests, Pixellink obtains the connection information between pixels, not between boxes. And it collects pixels based on the connection information to perform segmentation. As

with all segmentation based methods, the backbone can be chosen from any of the existing ones. The concept of link also appears in CRAFT[6], where it represents the connection between characters(not pixels or boxes). Basically, after character detection, the link information is used as a method of grouping each character in word units. The unique thing about CRAFT is that it processes the ground truth to obtain the heatmap-like form.(Fig. 2.2)

Compounding methods

In rare cases, there are a methods of mixing both(segmentation and box regression). In Corner-localization[26], the two branches perform segmentaion and box regression respectively, and the results are combined to obtain the last box. Using NMS in the process of refining the last box is the same as segmentation based methods.

2.7.2 Recognizers

There are many researches that deal with recognition only, but we do not describe them in detail as they are not closely related to this research. In [27] and [28], there are concise explanations and anatomy of the studies dealing with that theme. They handles detection studies as well.

2.7.3 End-to-end methods (detector + recognizer)

There are recognizers that perform both detection and recognition together. Of course, if you connect the detector and the recognizer, it becomes a simple net that performs both. But the criteria for judging a net is end-to-end is whether its training is performed simultaneously with a common loss. Mask-textspotter[29] is based on mask-rcnn[30]. Common features are drawn using FPN, and the character's position in a box as well. With these results from FPN, recognition is performed using RNN(Recurrent Neural Nets). The key feature is that the attention mechanism is applied to the RNN part, and the same principle is applied to the Resnest which is one of used nets in this study. The attention mechanism was originally a concept in natural language processing and widely used. Concretely when predicting each word one by one during the translation process, the attention mechanism was put into the network structure to see which part of each sentence should be more focused. It is known to work well for translation as well as object

detection. FOTS[11] has a structure similar to Mask-textspotter[29], but is characterized by the introduction of a ROI rotation. Therefore, a rotated box is possible for a detection result. [31] has a similar structure, but there is an additional warping part before putting it into the recognizer. That is, ROIs are obtained with the FPN, then after arbitrary ROI alignment, the boundary point is finely adjusted to obtain an accurate box shape. After that, the image is warped according to the boundary point and put into the recognizer. The attention mechanism is also applied there. ASTS[32] is based on the Mask-rcnn and is similar to Mask-textspotter, but it goes through several branches in the segmentation stage and uses a large and complex net in the recognition process to improve performance.

2.8 Dataset

We used MLT2019 data for this research. In order to set a record on ICDAR's leaderboard, It is not necessary to use only the data provided by the ICDAR, and moreover the dataset does not have to be public.(But it should be specified when submitting.) So, for a fair comparison, teams using different datasets should be evaluated separately. In Table 2.1, the top team used their own private data and achieved the record, so the 2nd team recorded the highest score(91.66%) among the teams that used only the dataset provided by the organizer.

2.8.1 ICDAR MLT

There are a few published datasets for detection (MSRA-TD500[32], ICDAR[9], Total Text[33] etc). However, as far as we know, there are no meaningful datasets other than the data provided by the ICDAR. There are two datasets of 2017 competition and 2019 competition. The 2017's data consists of 68,613 training data and 16,255 validation data. All images consist of only one word and may contain some margin, but basically it is a full image of words. The language is divided into 7 classes as Table 2.2. The 2019's data is basically same format as 2017's data and consists of 8 languages.

language	number of images	
	2017	2019
Arabic	3,711	1,784
Bangla	3,237	4,701
Chinese	2,702	3,935
Japanese	4,633	3,933
Korean	5,631	5,945
Latin	47,452	6,748
Symbols	1,247	58,541
Hindi	N/A	3,289

Table 2.2: Language distribution of ICDAR 2017 and 2019 data. The effects of unbalanced distribution will be discussed later.

year	2017	2019	2019 - 2017
Symbols	1.8%	65.9%	64.1%
Arabic	5.4%	2.0%	-3.4%
Bangla	4.7%	5.3%	0.6%
Chinese	3.9%	4.4%	0.5%
Japanese	6.8%	4.4%	-2.4%
Korean	8.2%	6.7%	-1.5%
Latin	69.2%	7.6%	-61.6%
Hindi	0%	3.7%	3.7%

Table 2.3: Changes in icdar data. We can see that the symbol and Latin have been dramatically reversed.

2.8.2 Synthetic data : Gupta

Infinite number of images can created by only printing any texts on images that does not contain texts. However, images generated like this is not very helpful for learning because the fonts on that images are monotonous(in size and typography), and the images does not reflect all various backgrounds, occlusion which is always a main problem in image recognition. A number of methods have been proposed to overcome these shortcomings, and the representative one is Gupta[33]. Gupta first performs segmentation to identify homogeneous parts on the target image which will be used as background. Because texts that we can actually observe in nature is naturally are written on a specific object even though many various objects can be on the image. This is intended to reproduce this *real* environment. For example, in a person's picture against the sky, texts that spans the person and the sky hardly can be observed. So synthetic texts must be generated on one object: the sky or a person in this case. Moreover, various variations are applied when synthesizing like changing sizes, modifying colors, various transformations considering background, and warping according to background shapes.

2.8.3 COCO-Text

The COCO dataset is a dataset released by Microsoft, and has a vast amount of images across various circumstances. In the dataset, there is a subset called COCO-Text which is text annotated dataset of natural scene. Although the amount is vast, natural scenes have a intrinsic limitation that annotations cannot be accurate. There are a lot of blurred text and a lot of half-cut text, and unrecognizable texts which are people can fully recognize based on the surrounding context. A total of 239,506 text instances can be obtained. But it is not used in this thesis.

Chapter 3

Proposed Methods

3.1 Base Network Selection

A backbone network is a core part of a big network. A backbone and additional ancillary networks are put together to form the entire network. Usually a backbone is repeated and stacked to a deeper and more complex application network. Vgg[1] and Resnet[2] are the most widely used backbone networks(hereinafter 'net'), and the nets pretrained with Imagenet[12] which help the net get better performance are public. We tested some networks(Vgg, Resnet, ResNeXt[4], ResNeSt[3], Efficient net[34]) and finally adopted ResNeSt. Clearly there are differences in performance depending on the backbones, as expected. It showed the best performance with a skip connection and the attention mechanism. The initial learning results of candidate networks when selecting the backbone are shown in the Fig. 3.1.

3.1.1 Googlenet

From the beginning of deep learning, it was widely known that increasing the size of the network is advantageous, but since it cannot increase the amount of computation infinitely, the problem of designing a network with comparable performace with a small amount of computation by adjusting the structure appropriately has emerged. This process is sometimes called to 'the compression of a network'. It is reasonable to think that the numerous parameters of the network will not all have the same degree of significance, so it would be best if only the important parts could be picked and made dense to a smaller network. Namely, the ultimate goal is to compress the

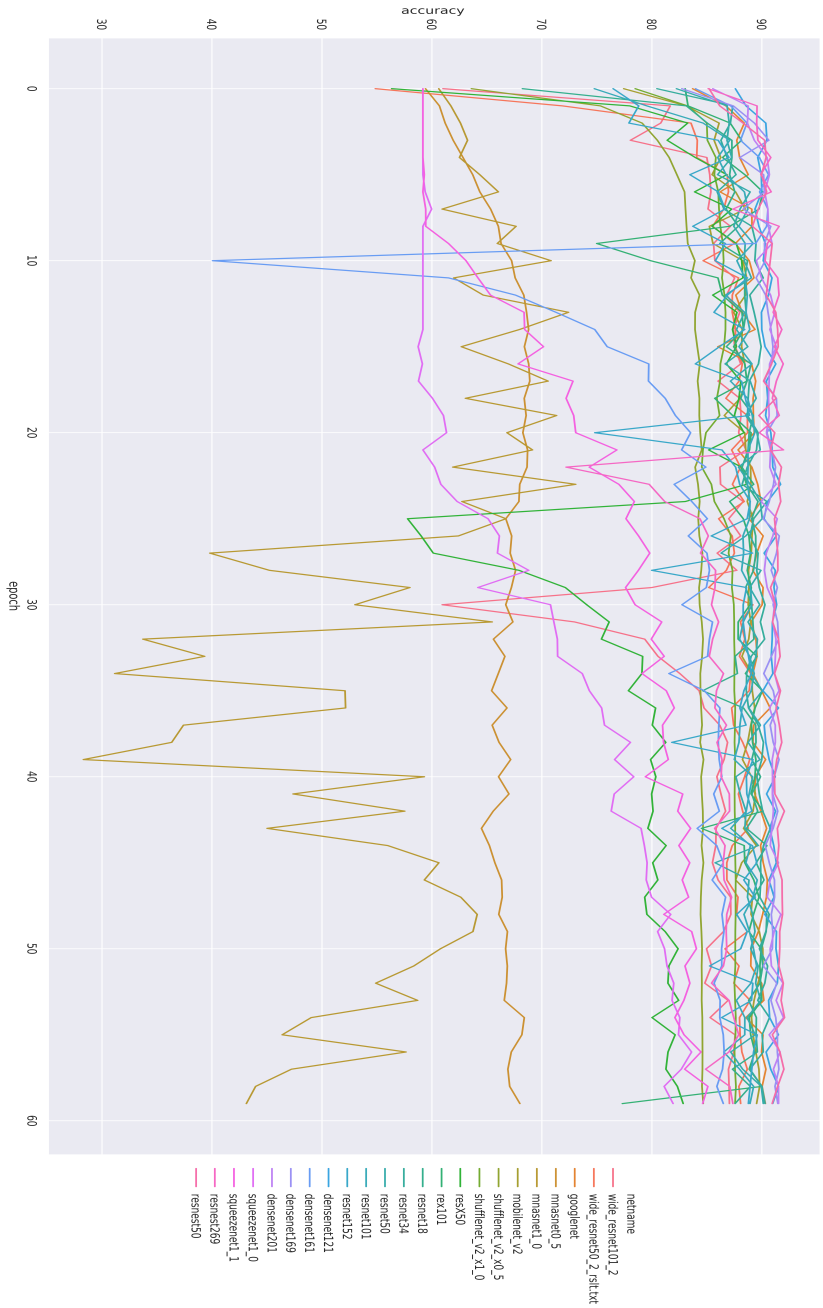

Figure 3.1: Some basic networks' initial learning experiment results. As other studies, we experimented with well-known feature extraction networks to find out which network is more suitable for this research.

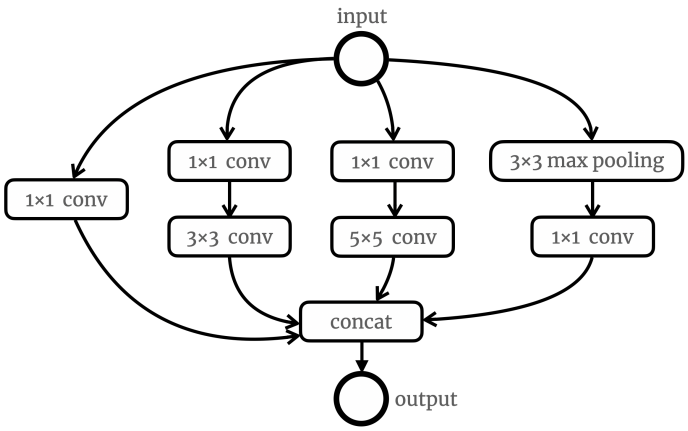

Figure 3.2: The network used as a basic block in Googlenet. Also called the inception block.

original network. According to the authors of [35], many of the previously studied backbone networks have already reached local optimum. Therefore, they do not try to find the optimal network from scratch and try to find a well-working backbone by the same way of former researches. A network can have infinite combinations of small blocks, but not all can be tried, so many researchers reassemble well-known small blocks to design a network. Because a few blocks are known to work well and have reached a local optimimum, so it makes sense to do so. In Googlenet, the basic blocks are 1×1, 3×3, 5×5 convolution blocks, and authors studied how to combine those blocks to get maximum performance. Like other studies, they designed the entire net by finding a local structure and repeating the structure (the authors called this small structure 'inception block'). Eventually, in the final stage of network design, it was not automated, but handcrafted. Figure 3.2 depicts the optimal block found by the authors.

3.1.2 Shufflenet V2

The goal of the Shufflenet[36] is to find an efficient(≈fast, lightweight) network. While other studies are primarily aimed at finding a balanced network between performance and efficiency, Shufflenet takes efficiency more seriously. Therefore, a smaller network with the same performance is defined as good,

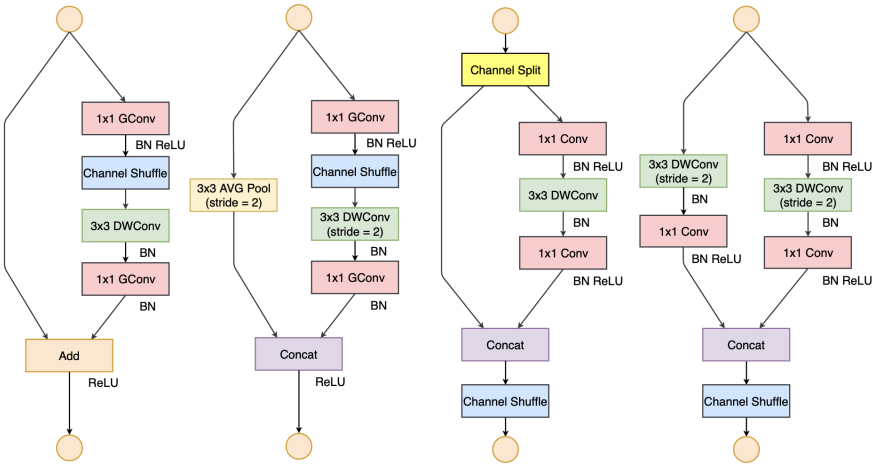

Figure 3.3: The network used as a basic block in Shufflenet. 'DWConv' represents 'depthwise convolution', and 'GConv' means 'group convolution'. Reprinted from [36].

and a network is better when the network size is remarkably small even if the performance is somewhat insufficient. Since the language classification task is aimed at highest accuracy, Shufflenet may not be suitable to our task, but since the task itself is not that complicated, this net was not ruled out because a small network could show good performance. Like other studies, the authors tried to optimize a basic unit block, and there are several sub-networks found in that way. All networks of Fig. 3.3 are selected based on computational efficiency.

3.1.3 Resnet

The structure of Resnet50 is shown in the Figure 3.4. 3 channel image of 1000 output classes, 255×255(width×height) is assumed as input. Resnet18, Resnet34, Resnet101 and others have more stacked layers of same structure without a meaningful difference.

In the original paper, a block is expressed as Fig 3.5. In the figure, the skip connection, which appears as 'identity', is to alleviate the phenomenon that the training error becomes larger as the layer becomes deeper. Assuming that the deeper the layer, the more accurately the original distribution is

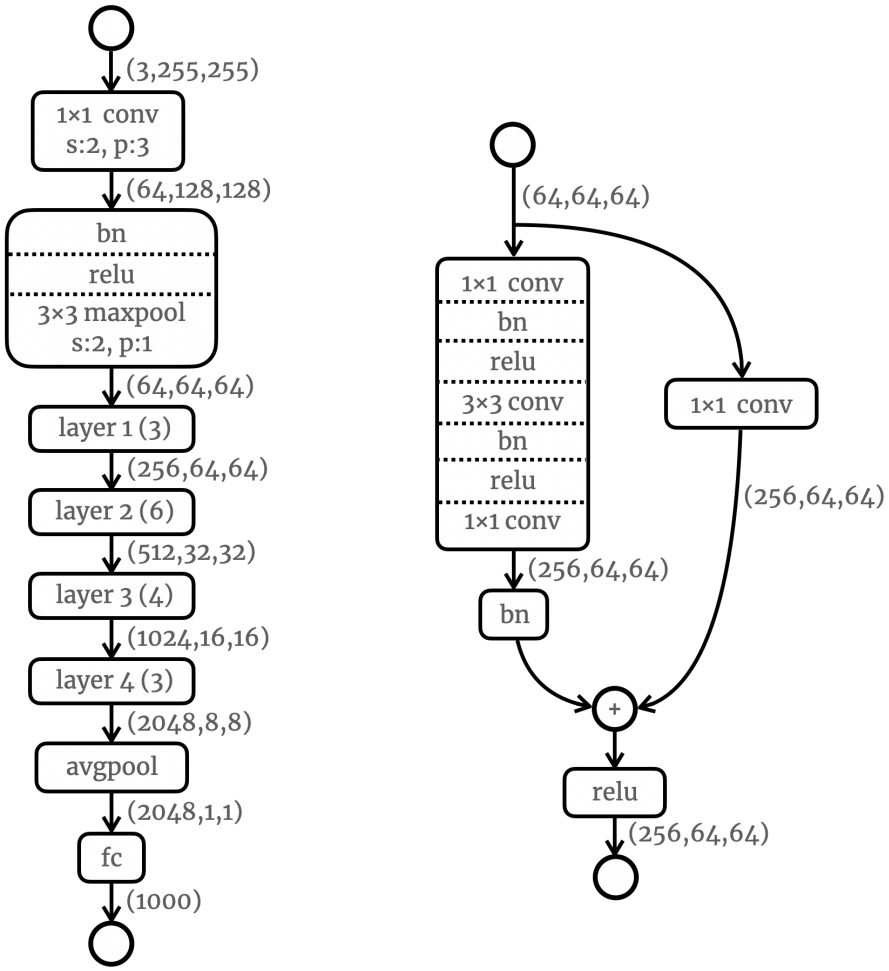

Figure 3.4: A brief structure and data flow of Resnet50. Numbers in parentheses is the shape of the data. bn: batch normalization layer, fc: fully connected layer, avgpool: average pooling layer, 's:2' means '2 pixel stride', 'p:1' means '1 pixel padding'. Middle layers on the left figure are composed of some small blocks. Concretely, 'layer 1 (3)' means 'this layer is composed of 3 sequential *small block*s' (layer number is just for distinction with other layers). The *small block* is as shown on right figure. Fig. 3.5 is exactly the same with this.

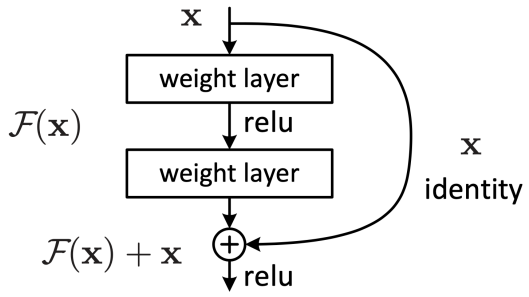

Figure 3.5: A block of Resnet. Reprinted from [2]

expressed, it cannot be explained that the error of a light-weight net is smaller even after training is sufficiently performed. Assuming you have added a few layers after optimizing with fewer layers, the error of this new network should be bounded to the existing network. In fact, empirically, the opposite phenomenon is observed. In order to alleviate that, a connection that passes the previous learning result as it is could be added to prevent increasing training error. This idea works well and is still in use today because of its good results.

3.1.4 Wide Resnet

The observation that network performance does not improve even if the network is very enlarged with a huge amount of computation has become almost meaningless since the advent of the 'skip connection', and extremely deep networks of more than 1k layers can be implemented using the skip connection trick. Afterwards, researchers began to study how to adjust the order of activation or to apply the drop-out technique to skip connections, and they also focused on designing residual blocks (this trend is still dominant now). On the other hand, there is a research to improve the performance by making the network wider. Wide resnet[37] is a prime example. The idea is to connect the same network structure in parallel to take an effect similar to an ensemble network. It is known that this idea reduces the number of parameters of a network and naturally speeds it up. Experiments have shown that evan a thousand layers can be reduced to 16 while maintaining the performance. Previous studies focused on depth, which means same structure blocks stacked repeatedly. But Wide-resnet want to make the net wider in parallel. The efficient net(3.1.8), which will come out later, integrate these

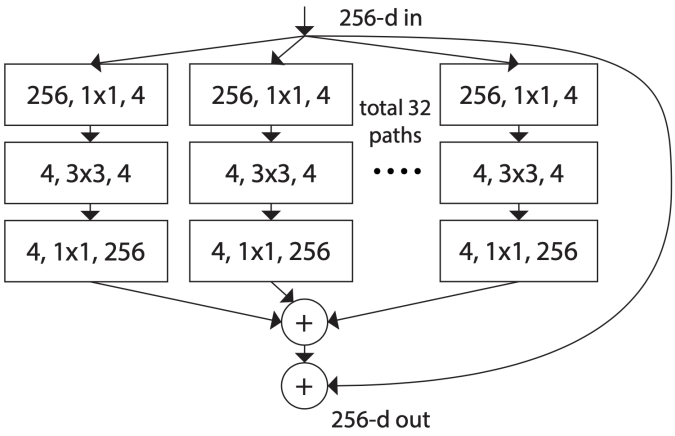

Figure 3.6: A block of ResNeXt. Reprinted from [4]

ideas and go in the direction of finding the optimal ratio between width and depth.

3.1.5 ResNeXt

Resnext[4] is "modularized network architecture for image classification". This net, too, aims to organize properly designed building blocks. Specifically, 1) multi-branch 2) grouped convolution was used. Multi-branch, also called *cardinality* or *radix*, is known to have the effect of ensemble. The representative example of the grouped convolution is *channel-wise convolution*(this also called *separable convolution* or *depth-wise convolution*), which omits part of the convolution operation. Since a vast amount of calculations are performed, it is known that comparable performance can be obtained even if part of the intermediate calculation is omitted and the operation is simplified. Figure 3.6 shows the basic block of the Resnext.

3.1.6 ResNeSt(Split-Attention network)

The structure is almost the same as that of Resnet(Fig. 3.4). The differences are that the conv-bn-relu layer(The convolution layer in the middle of left side of Fig. 3.7 maintains the same number of channels.) which appears first is more complicated then Resnet, and the attention branch applied to the every bottleneck blocks. The right side of Fig. 3.7 shows when the

bottleneck's radix is 2. Briefly, the 'radix' means how many branches are split for the attention mechanism.

In this research, the output layer was limited to 8 classes except the third layer in the ResNeSt presented in 3.1.6. Since the target problem is relatively simple, the layers do not need to be too deep, and as expected, we could not observe a significant difference according to depth of the backbones. We used a model pretrained with ImageNet data.[1]

3.1.7 Densenet

This can be seen as an application of the skip connection. In the case of resnet[2], only the input and output of a block is connected, but in the case of densenet[38], inputs and ouputs of multiple blocks are connected with multiple skip connections. Another major difference is that the identity connection is not added but concatenated to form multiple channels.

3.1.8 EfficientNet

Efficient-net[34] is not the result of implementing a concrete idea, but it is the result of efforts to find more general rules for designing the network. Many studies using deep learning, including this one, implement a specific idea to check the results with many trial and errors, but it does not form a systematic knowledge, and each research merely explains its own. This is a kind of art rather than an academic one, so in addition to too general theories such as universal appoximation theorem, more technical aspects need to be systematically organized. So, the efficient net made a simple hypothesis and verified that idea about how the net should be further expanded, starting with the basic network. The rule for extending the net from the basic structure is as follows.

$$\begin{aligned} \text{the network's depth} &= \alpha^{\phi} \\ \text{width} &= \beta^{\phi} \\ \text{resolution} &= \gamma^{\phi} \\ \text{s.t.} \quad & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3.1}$$

[1]Most deep learning libraries offer pretrained models. We used PyTorch. `https://pytorch.org`. Pretrained models can be used by importing `torchvision.models` module in PyTorch.

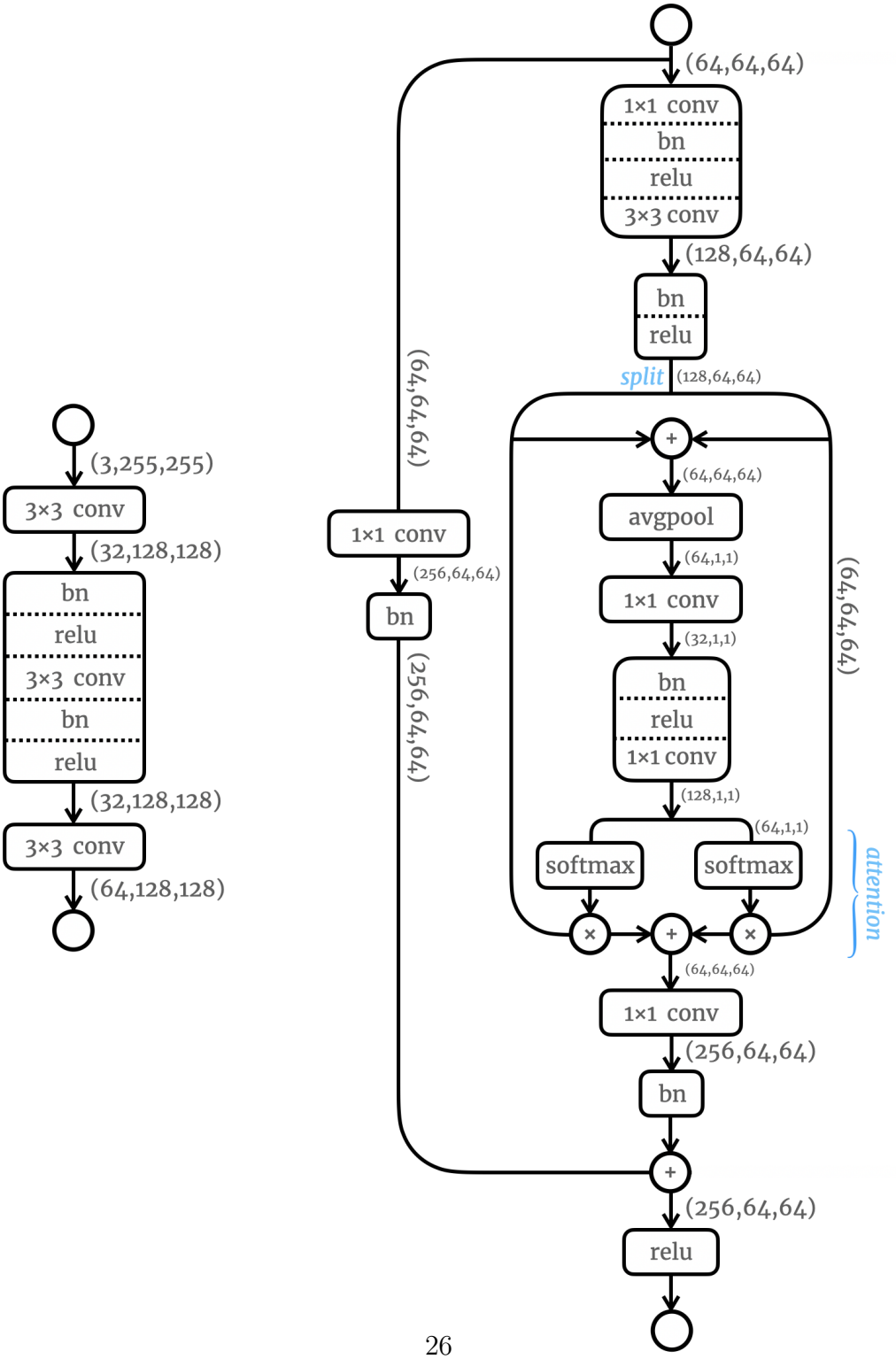

Figure 3.7: Left: The entry point of ResNeSt[3].
Right: A brief structure of ResNeSt when the *radix* is 2.

where α, β, γ are found by grid search and ϕ is a hyper parameter.(Refer to Table 1 of the paper[34] for *the basic structure*.)

The experimental results showed an overwhelmingly fast convergence speed.(We used version 8, namely Efficientnet-b8.) However, the limitation of the efficient net is that it is an expansion from 'the basic structure'. Even if it is admitted that the best rule of designing a neural net is the suggested way above(extending width, depth and resolution from a basic network), it is not possible to integrate all the current research results that are composed of complex nets such as ResNeSt and Resnext. Apart from the academic limitations of the efficient net, the initial performance was sufficient to be used in this study, we experimented this network as a backbone as well as above two networks. So all three were basically tested as feature extraction backbones for all experiments, and the best record was selected.

3.1.9 Automatic search : AutoSTR

There have been many attempts to automatically find the optimal network structure(NAS: Neural Architecture Search). AutoSTR[39] is an application of this works in relation to OCR field. More specifically, the study explores possible backbones for the recognizer. Since the number of combinations of neural connections is infinite, it is important to limit the search range and conditions. Because, In the end, you should test all possible structures. However, we can save time by experimenting only a little (1$\sim$2 epoch) in the beginning without the entire training process to the peak performance. Empirically, we can considere this true.

AutoSTR process is divided into operation and downsampling. Downsampling is just a pooling, and operation is predefined several types of convolution. Of course, the search space is infinite, so with limitting it like this, we may not be able to find the optimal network. However, it is more realistic to limit the amount of calculations and find the net as close as possible to the optimum. Still, there are a lot of possibilities. In this AutoSTR research, since 7 operations are searched for 15 layers, 7^{15} combinations are possible. And the research says "Empirical observation that choices of the downsampling path and the operations are almost orthogonal with each other".

Observing the found net, the deeper is better, and the simpler the operation at the rear end is better. This trend is consistent with the recently hand-designed net like SCRN[40]. And this research noted that the architecture found in other work was not very good to apply to another task. This

indirectly supports the intuition that the ideal network(if it exists) structure will depend on the data of a particular task.

3.2 Methods

We propose methods not only applicable only to this task, but including many known techniques. These listed methods are in the order of effective ones, so even one method is very original and do not work, it is listed later. Practically, most of the performance improvement was obtained from the previous four methods, and from 3.2.5 there was little effect.

3.2.1 Ground truth cleansing

filename	ground truth text
word_42191.png	02-803-8275
word_44983.png	6:30 24:00
word_45563.png	No.1
word_45661.png	A-m
word_47042.png	Congratulations!!
word_48993.png	1F
word_49705.png	paice
word_49761.png	TEL:
word_52839.png	8:00 21:00
word_52844.png	8:00 21:00
word_52937.png	8:00 20:00
word_53783.png	No.3
word_72963.png	055-21

Table 3.1: These are all labed as Japanese, but only composed with Latin characters.

error correction

Examining the training data of MLT2019, we can find some errors. For example, in the case of Fig. 3.8, the image is labeled as Latin(It should be

Figure 3.8: An example image that is labeled to Latin. word_88976.png

corrected to Hindi). It is an obvious error to be corrected. Typical errors are shown in Table 3.2.

Latin filtering

And Latin tends to be included in all languages, so Latin is often classified in any language. it's certainly not an error, but it's a bad data for the classifier that needs definite ground truths for more better accuracy. In the Table 3.1, images only includes Latin characters, but they are classified to Japanese. Latins can be filtered across all languages, and it is a process that can be easily automated without manual intervention.

CJK issue

In the Chinese-character culture(Korea, China, and Chinese), many Chinese characters are used in common, so another problem arises. Here too, we used a simple solution that could be automated. in the case of Japanese, we filtered images that includes katakana and hiragana, and the rest was labeled Latin or Chinese. It might be possible to classify images that includes Chinese characters that are only used in Japan nowadays(for example, ancient Chinese) as Japanese. But since it costs too much to process them separately, we have not done so.) In the case of Korean, labels containing only Chinese characters are revised from Korean to Chinese(Fig. 3.9). This can be easily done because we can classify many languages according to the Unicode of a character. If an image includes Hangul and Chinese characters as well, it's label was kept as it was(Korean).

The number of data that can be modified in these way is 162 totally, which is only about 0.2% of the total. But as a result in experiment, the accuracy is improved by more than 1%. It can be seen that several outliers significantly influence the training results. After obtaining the motif from

File name	Original annotation	Modified to	Text
word_34149.png	Korean	Latin	Step3.
word_34162.png	Korean	Chinese	意
word_34163.png	Korean	Chinese	改善了
word_34178.png	Korean	Chinese	理
word_34179.png	Korean	Chinese	3分
word_34180.png	Korean	Chinese	性
word_34298.png	Korean	Chinese	(憲宗)
word_34300.png	Korean	Chinese	(金在
word_34301.png	Korean	Chinese	(慶嬪)
word_34302.png	Korean	Chinese	(和)
word_34498.png	Korean	Chinese	大
word_34500.png	Korean	Chinese	中
word_34501.png	Korean	Chinese	小
word_34504.png	Korean	Chinese	大
word_34505.png	Korean	Chinese	中
word_34506.png	Korean	Chinese	小
word_35253.png	Latin	Korean	12.000원
word_35283.png	Korean	Chinese	大
word_35393.png	Korean	Latin	C
word_35836.png	Japanese	Korean	스팀
word_36101.png	Korean	Latin	5F
word_36387.png	Korean	Chinese	俊
word_36471.png	Korean	Latin	3.4F
word_36484.png	Korean	Latin	4F · 3F
word_36612.png	Korean	Chinese	(蘭谷)
word_37642.png	Korean	Japanese	日本
word_38053.png	Korean	Chinese	悲
word_38277.png	Korean	Chinese	生
word_38756.png	Korean	Chinese	中
word_39201.png	Latin	Korean	현위치
word_39336.png	Latin	Korean	한국공에디자인
word_39337.png	Latin	Korean	아트샵
word_39338.png	Latin	Korean	공평아트
word_39746.png	Latin	Korean	료
word_39747.png	Latin	Korean	주차장

Table 3.2: Errors modified by character code value.

word_39906.png	Korean	Chinese	微信一
word_39907.png	Korean	Chinese	航
word_39908.png	Korean	Chinese	惠
word_39909.png	Korean	Chinese	海
word_39910.png	Korean	Chinese	限首
word_39911.png	Korean	Chinese	每惠
word_39912.png	Korean	Chinese	最多惠
word_39914.png	Korean	Chinese	入金
word_39915.png	Korean	Chinese	微信一
word_39916.png	Korean	Chinese	使用
word_39917.png	Korean	Chinese	最多直
word_39918.png	Korean	Chinese	元
word_39922.png	Korean	Chinese	暖游
word_39926.png	Korean	Chinese	夏威夷科
word_39928.png	Korean	Chinese	海洋深水
word_40210.png	Korean	Latin	TEL:
word_40211.png	Korean	Latin	FAX:
word_40212.png	Korean	Latin	823-5590/825-55
word_40389.png	Latin	Korean	총무과장
word_41010.png	Korean	Latin	www.daeboro.co.kr
word_42849.png	Korean	Chinese	受理
word_43320.png	Latin	Korean	2월
word_43321.png	Latin	Korean	16일
word_44849.png	Latin	Japanese	15分~¥1.000
word_44850.png	Latin	Japanese	30分~¥2.000
word_44851.png	Latin	Japanese	45分~¥3.000
word_44852.png	Latin	Japanese	60分~¥4.000
word_46810.png	Korean	Japanese	駐車料金のお支い
word_49521.png	Japanese	Korean	시즈오카현청
word_51518.png	Korean	Japanese	丹魂反中越
word_54030.png	Latin	Chinese	新年音
word_54055.png	Latin	Chinese	之音院
word_88975.png	Latin	Hindi	
word_88976.png	Latin	Hindi	

Table 3.3: (cont.) Errors modified by character code value.

Figure 3.9: This is labeled as Korean. Surely, Korean uses Chinese characters, but images like this are better to be labeled as Chinese.

this, an experiment that exclude very hard items when training was designed and tested. It will be covered in the 3.2.4.

3.2.2 Divide-and-stack

Before a training image is entered into the network, it should be resized to a square. However, some of the training images are extremely long, and after resizing, the original letters are distorted badly. Table 3.4 and Fig. 3.10 show the ratio distribution of training data.

As you can see from the data distribution, the images that have the ratio over 3 is more than 30%.(Of the 89,177 images in total, 27,124 images are in this case, which is 30.42% of the total). In these cases, after resizing, it is very difficult to recognize them as Fig. 3.12. In order to improve the accuracy, those with a ratio of 3 or more were moderately cut and stacked and put as input so that the ratio of the characters was not excessively distorted. As a result, this method was successful to improve the performance, and there was an improvement of 0.46% compared to the cases without 'divide and stack'(3.2.2, hereinafter DNS).

Algorithm 1: Divide and stack

Result: Moderately divided and stacked image
ratio = input image's width per height;
if *ratio is between* 3 *and* 0.3 **then**
    return input image as it is. `// do nothing`
end
if *horizontally long image* **then**
    divide number = max(2, $\sqrt{ratio}$);
    divide image(by divide number times);
    concatenate them and return;
else
    do almost same as horizontal case;
end

ratio (width/height)	number	percentage
∼1/6	298	0.334
1/6∼1/5	180	0.202
1/5∼1/4	341	0.382
1/4∼1/3	684	0.767
1/3∼1/2	1,432	1.601
1/2∼1	9,923	11.127
1∼2	27,754	31.123
2∼3	22,595	25.338
3∼4	12,943	14.514
4∼5	6,352	7.123
5∼6	3,132	3.512
6∼	3,542	3.972

Table 3.4: The distribution of image patches of MLT2019 task 2 data.

3.2.3 Using additional data

Synthetic data

If synthetic data is added, it is expected to improve performance by 1 to 2%. In this study, there was no significant performance difference. In the present case, it is understood that there are no additional performance improvements because there are already enough 89,000 training data. If more data that can improve performance enough must be added, firstly we must figure out the characteristics of the images that are not classified well and use similar data for further learning. As of now, there seems to be no special solution other than collecting new data.

Available public data

Besides the MLT, there are some other text datasets. However, most of them are a small amount of data for text detection, and the data for language classification is almost unique 2017MLT. The best results were obtained here. The score is 93.25%(0.42% improvement over results obtained using only MLT19 data), which is a great level, but we think it is not a meaningful discovery because getting better result by adding more data is not considered

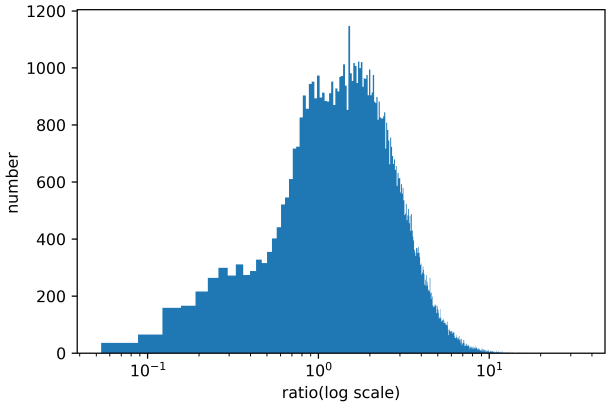

Figure 3.10: The data distribution of Table 3.4. The more it goes to the right, the longer the image is horizontally.

Figure 3.11: An example of a long image. The length-to-height ratio of this image is over 30. (word_28177.png)

new.

The top team in Table 2.1 achieved the highest score(94.03%) in this way. (In the description of submission, they say "We first recognize text lines and their character-level language types using ensemble results of several recognition models, which based on CTC/Seq2Seq and CNN with self-attention/RNN. After that, we identify the language types of recognized results based on statics of MLT-2019 and Wikipedia corpus". And It is not known how much more data was used.)

3.2.4 OHEM

OHEM(Online Hard Example Mining)[41] is a method to learn from only 'hard examples' after some epoches of training. Specifically, the network does not learn from all the samples but the part of training data that have the

Figure 3.12: Comparison when 'divide-and-stack' is applied to Figure 3.11 and not.

larger loss compared to other samples.(The network calculates losses for all samples, and the losses are ordered in descending order. Finally, the network back-propagates only for samples with larger loss.) There is no clear rule how to set the threshold, but it is known that it does not depend heavily on the threshold, so it is usually set to about top 20%. We tested ResNeSt200 based network. (The number is proportional to the depth of the layers and there are 50, 101, 200, and 269. It is possible to make it deeper by just stacking more blocks like common Resnets.) Our target network have batch norm and relu activation immediately before the last fully connected output layer. And the DNS also applied. As a result, the accuracy is improved by 0.29% compared to OHEM not applied method(from 92.54% to 92.83%).

3.2.5 Network using the number of characters

We designed and tested a network using additional information. MLT19 provides text as well as script language, and this information can be used to find out whether there is an improvement in performance when using a network that predicts how many characters are contained or which characters are contained as well as language classes. First, we take a feature with length 1,000 from the backbone, and create two outputs that predict the class and number of characters. The loss was the sum of the two outputs' losses. Experiments were conducted using two types of character output,

one of which is the channel type that each channel represents the number of characters(this uses crossentropy loss), and the other is the number itself(in this case, the root mean square error is used). We tried to adjust the ratio of the two losses from 3:1 to 0.3:1, but did not conduct further research because the performance(accuracy below 65%) was inferior to the network that does not use additional information. Secondly, We tried to predict the letter itself like the recognizer (the number of output is 8392. It is after the combining half-width or full-width characters, special symbols of similar shape, etc. But it didn't work as expected. We think it was because it was difficult to converge to an appropriate point because the number of data was small compared to the number of letters. It was the same even after applying class balancing.

3.2.6 Use of R-CNN structure

Like many of the detectors mentioned in 2.7.1 above, it is possible to apply R-CNN. In order to check whether this method is suitable for this study, we made a basic network and tested it(Figure. 3.13). We used **ResNeXt50_32x4d** as the backbone network to extract features. In the network name, '50' represents the number of layers and '4d' represents that the radix is 4('radix' is described in 3.1.6. The term has the same meaning in both ResNeXt and ResNeSt. To put it roughly, applying attention to ResNeXt is ResNeSt.). The reason for using the net as a feature network is that when we experimented with **ResNeXt101_32x8d** its accuracy is about 90% without RPN(Region Proposal Network). It is remarkable score compared to other base networks(vgg, densenet, etc...).(See Fig. 3.1)

After extracting the feature of an input image, a network for extracting coordinates was attached to extract the ROIs, and the size of an ROI was fixed to 16 both horizontally and vertically to simplify the network(Fig. 3.13). So this is more likely classifying a letters by picking up a character in some part of the long image(The height of the 2D feature is 16), rather than classifying a patch by taking an appropriate spot. The classifier part was also designed as 3.13 and experimented. In the ROI pooling layer, unlike Faster R-CNN[42], **ROI align** of Mask R-CNN[30] is used because it is known to show better performance. The difference between the two is that ROI_align includes interpolation operations to minimize numerical errors during feature extraction.

As a result, this method did not produce good results. At epoch 15,

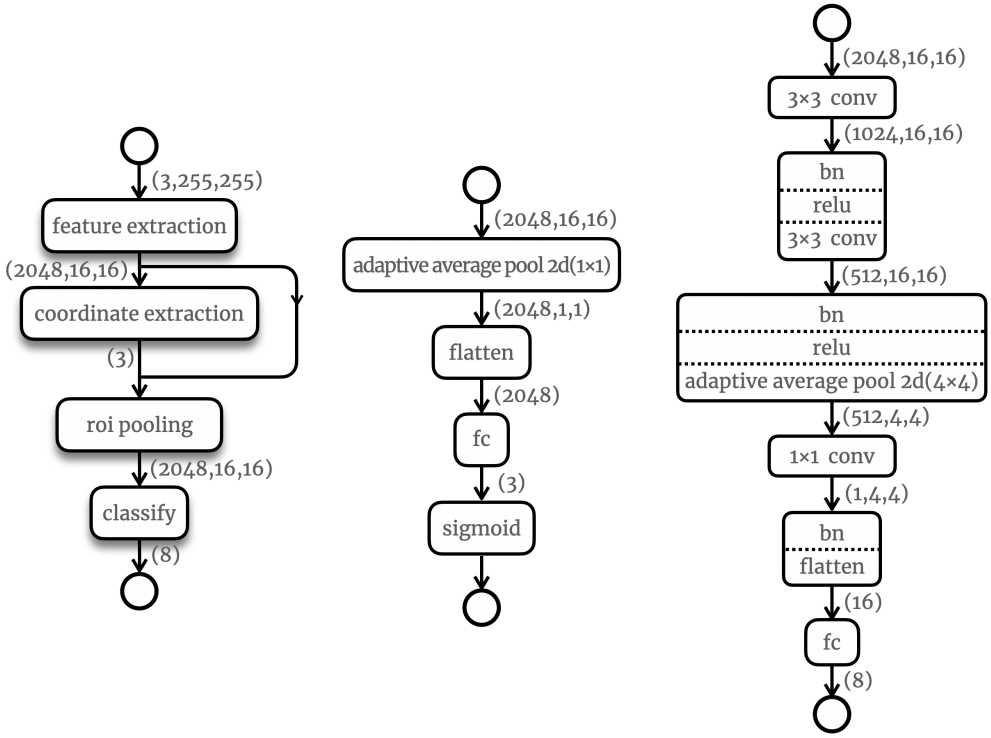

Figure 3.13: The overall structure of the network used in the R-CNN application experiment of 3.2.6 and subnetworks. The coordinate extraction layer is the middle figure, and the classifying layer is the right rigure. In the roi-pooling layer, **ROI_align** is performed.

accuracy reaches around 85% and converges at that level. The highest was 87%. The confusion matrix is the same as 3.14, and as is commonly observed in this research, Japanese is most confused with Latin.

To check whether picking a specific area has effect or not, we cropped the center of all images into a maximum square shape regardless of the ratio of the image. As the distribution of the image ratio is shown in Fig.3.10, you can check that most of the data is square, so the number of case of actually being the central crop is about 30% of the whole(27486/89177. Up to ratio 2 the image is regarded as square because even if a 2:1 image is pressed as a 1:1 image, there will be no difficulty in recognizing the text. Reflecting this, DNS only cuts images that are 3:1 or higher). As a result of this experiment, we can get an accuracy of 89.07% which is much better

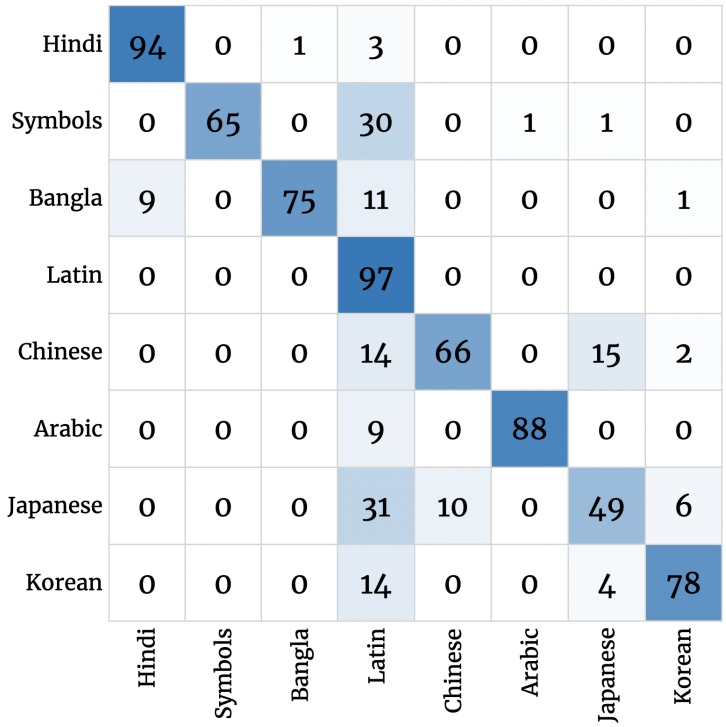

Figure 3.14: A confusion matrix showing the results of a network using RPN according to the language. The sum of each row is 100 or less(Since values less than 1 are all truncated). It is common for all languages to be confused with Latin, especially in Japanese.

value than the RPN using experiment. So we can conclude that RPN is not very helpful in language classification task.

With DNS

We can consider the method to applying DNS while using RPN. Using ROI method is a way of detecting an important part of an image and looking at that part in detail(there is a similarity with the attention mechanism in this respect). Since DNS maintains the original proportion of the image as much as possible, we can expect that good ROIs from a divided-and-stacked image can be a benefit to the classification.

As a result of experiments, the highest record was 87.59%, which proved that DNS was helpful (+0.59%), but due to the limitations of the network

using RPN, best accuracy was not obtained.

3.2.7 High resolution input

When it is necessary to detect both large and small objects simultaneously, designing the detector in multi-scale form or receiving a largely resized input helps. However, in this task, inputs are relatively uniform and there are very few exceptional cases, so the performance improvement expected by multi-scaling is not significant. Of course, a multi-scale detector can improve the performance when it comes with the extreme ratio cases even a little. But since we decide to solve that problem by DNS(3.2.2), it is unlikely to expect further improvement.

Surely after the DNS, the size of the letters may vary. But we use not only DNS but also the Attention Mechanism by ResNeSt, It is not likely for a high resolution to show more higher score.

As expected, even double sized input(512×512) was given, there was no meaningful improvement in performance.

3.2.8 Handling outliers using variant of OHEM

It is already confirmed in 3.2.4 that OHEM works. And because the MLT19 data is not perfect as shown in 3.2.1, it is possible to learn by slightly modifying OHEM to exclude not only *easy* examples, but also most *hard* data. In other words, the most difficult(=largest loss) data is considered errorneous and excluded from learning. The idea is to put a routine that takes only the smaller losses after normal OHEM which takes only the larger losses. Without having to check every samples manually, we can assume a small number of samples of the whole(like around 1% or 5%) to be false. As a result, the experiment was unsuccessful. The accuracy did not exceed 82%. When selecting a part of the sample, we experimented many cases by grid search to find the working ratio such as the top $x\%\sim y\%$, but most did not differ from each other. This is consistent with what is known to be that OHEM is fairly robust to thresholds. Eventually, unlike what we assume that the extremely most difficult data is an error, we can see that a lot of difficult (=important for learning) data is removed, not actually an error.

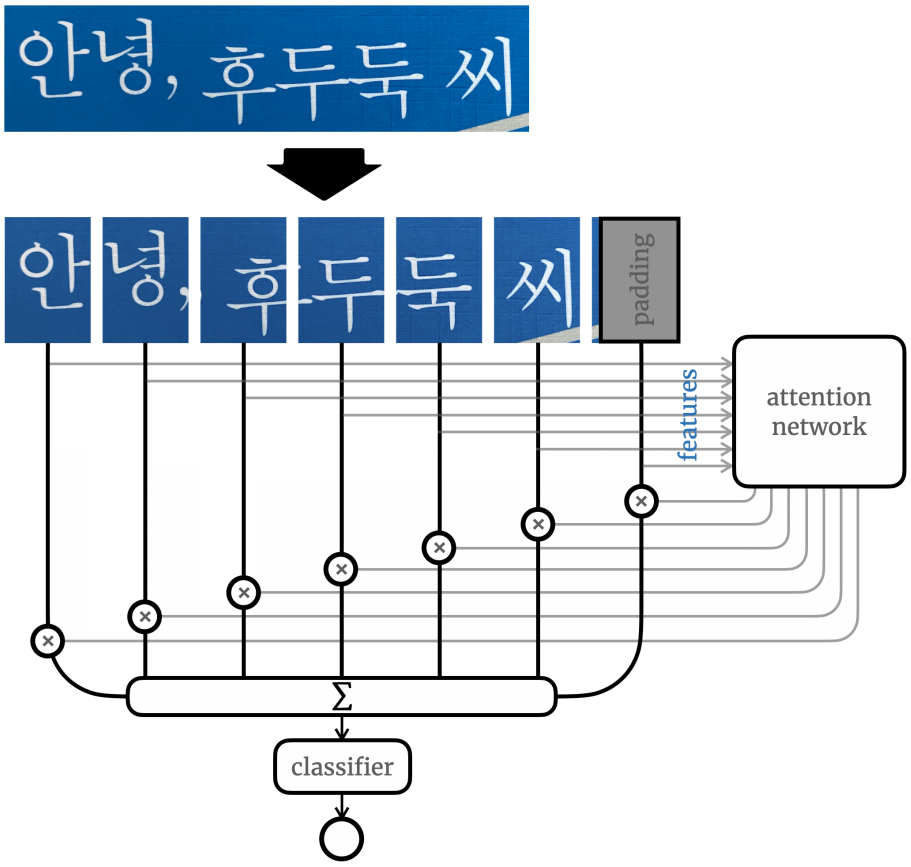

Figure 3.15: This shows the network's structure used in 3.2.9. A long image is divided into smaller patches and the features are extracted for each. As shown in the image, the last patch that cannot be obtained with a designated ratio will be padded. Note that you may not get exactly one letter for a patch. This can be one of the critical reasons for not getting good results.

3.2.9 Variable sized input images using the attention

To use an image with an arbitrary length as an input without deforming, we apply square patches of the image to the feature network one by one, and get a weighted sum of features with weights that are taken from the attention network. In this way, a constant length's feature vector can be obtained from an image having an arbitrary length. We used **ResNeXt50_32x4d** as a backbone which showed good performance in other experiments.

As a result, this method achieved less than 50% of accuracy. The structure of the network(Fig. 3.15) seems to have no reason to significantly degrade performance regarding the steady decreasing of training loss during learning. But it seems that the network fails to learn in general because the batch size must be set to 1 when training. Due to the PyTorch's limitation, the batch size cannot be more than 1 if training data's shape is variable. And as far as we know, No library supports that feature. So the learning effect is greatly reduced.

After the inputes patch passing through the feature network, we tried to apply them to RNN one by one without obtaining a weighted sum. But in that case, the accuracy was further reduced. Since this task is to classifying *all* language included in the whole image to *one* class, it seems that the method of classifying from the left side of the image one by one is more disadvantageous than the method of classifying the whole as one.

3.2.10 Class balancing

When the number of samples is uneven in the multi-class problem, class balancing is frequently used to improve performance. Since the distribution of the current data is as shown in Table.2.2, it is natural to think that the performance can be improved by balancing the data. However, as a result of actual experiments, it did not help, but rather decreased the performance by 5∼6%.

There can be many ways to balance data, and we have experimented with two of them. One is to match the data count of all classes equally. The number of images is then adjusted based on the class that contains the least data. The second method is to give a constant buffer. It is a method to adjust the number of the smallest data and the largest number of data so that it does not exceed a certain ratio. In both experiments, class balancing was found to decrease accuracy. Even when the number of images per class

in the test set was also reduced forcefully to the same ratio of the traing set, the result was same.

As shown in 2.2, the image number of smallest class is 1,784 in Arabic. But examining the confusion matrix in various experiments, as in 3.14, the distinction between 'Japanese and Latin' is the poorest not between 'Arabic vs other' or 'Symbols vs other'. So it can be concluded that unbalance of training data do not significantly affect to the accuracy in this task.

3.2.11 Fine tuning on specific classes

In almost all experiments the precision of Japanese and Latin is not good. So after training for the whole, we tried fine tune the classifier by training a little more for only two(Japanese, Latin) languages. If the learning rate was not low($>$0.001) or the number of epochs was over one or two, the classifying ability for other classes was significantly reduced. Moreover, performance improvement could not be observed even with a very little training. In conclusion, it is not meaningful to fine-tune with only those classes to increase resolution between classes that cannot be distinguished.

3.2.12 Optimizer selection

As an optimizer, Adam[43] and SGD(Stochastic Gradient Descent) are most common, and the learning rate can be reduced by a certain epoch, or a scheduler can be applied like 'warm up learning rate'[44], the 'weight decay' are also common as well. There is no standard method or correct answer, so they are selected empirically and tried several times heuristically to produce good results. In this research, we tried to change the learning rate for each epoch, and used Adam/SGD with various settings. As shown in Fig. 3.1, after reaching a certain degree of saturation, no significant difference was found. Depending on the settings, convergence was a little faster or slower, but the accuracy could not be increased significantly. Because the accuracy has already reached mid-90%, further improvement seems to require a more fundamental solution.

3.3 Result

With an accuracy of 92.83%, we achieved the highest score among currently published[8] records which used only the given training data of MLT19. (2nd

place in total, 1st and 3rd are the same team. They used the data they collected separately and privately. This ranking can be updated at any time, and the most recent record can be found on the ICDAR Competition homepage leaderboard.[2]) We used ResNeSt200 and all the methods from 3.2. Before epoch 10 (about 2k iterations) the accuracy reaches around 90%. But the effect of OHEM tends to appear slowly, so the network was fully trained to the 100 epoches(about 20k iterations). For ResNeSt269, the results were almost the same, and the other networks that are investigated in this research all showed slightly inferior performance.

[2]https://rrc.cvc.uab.es/?ch=15&com=evaluation&task=2 [Accessed:23-Jul-2020]

Chapter 4

Conclusion

We performed the language classification with MLT2019 image data, and were able to achieve state of the art record by ①appropriate selection of suitable backbone, ②modification and refinement of ground truth, ③OHEM. Additionally, it has been shown that a better record can be achieved by ④ adding data. And we have observed the results of OHEM's variant. Through experiments, we tried to modify and apply various well-known networks to this task and recorded the result. Besides, we observed the effects of well-known methods such as resizing the input image, class balancing, and fine tuning. In the future, like 1st place of Table 2.1, there will be more trials to process all steps(detection - language classification - recognition) at once without dividing each steps because that method can cover more wide applicable range. However, as the demand for OCR research increases, study in various directions will be conducted. And we hope this research will be useful for one of them, and will satisfy some practical needs as well as academic needs.

Bibliography

[1] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[2] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[3] Hang Zhang et al. "Resnest: Split-attention networks". In: *arXiv preprint arXiv:2004.08955* (2020).

[4] Saining Xie et al. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[5] Chee Kheng Ch'ng and Chee Seng Chan. "Total-text: A comprehensive dataset for scene text detection and recognition". In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 1. IEEE. 2017, pp. 935–942.

[6] Youngmin Baek et al. "Character region awareness for text detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9365–9374.

[7] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[8] N. Nayef et al. "ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition — RRC-MLT-2019". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, pp. 1582–1587.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[10] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[11] Xuebo Liu et al. "Fots: Fast oriented text spotting with a unified network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5676–5685.

[12] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[13] Enze Xie et al. "Scene text detection with supervised pyramid context network". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 9038–9045.

[14] Xiaobing Wang et al. "Arbitrary shape scene text detection with adaptive text region representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6449–6458.

[15] Hasim Sak, Andrew W Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: (2014).

[16] Chengquan Zhang et al. "Look more than once: An accurate detector for text of arbitrary shapes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10552–10561.

[17] Xinyu Zhou et al. "East: an efficient and accurate scene text detector". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 5551–5560.

[18] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[19] Minghui Liao, Baoguang Shi, and Xiang Bai. "Textboxes++: A single-shot oriented scene text detector". In: *IEEE transactions on image processing* 27.8 (2018), pp. 3676–3690.

[20] Minghui Liao et al. "Real-Time Scene Text Detection with Differentiable Binarization." In: *AAAI*. 2020, pp. 11474–11481.

[21] Weijia Wu, Jici Xing, and Hong Zhou. "TextCohesion: Detecting Text for Arbitrary Shapes". In: *arXiv preprint arXiv:1904.12640* (2019).

[22] Wenhai Wang et al. “Shape robust text detection with progressive scale expansion network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9336–9345.

[23] Shangbang Long et al. “Textsnake: A flexible representation for detecting text of arbitrary shapes”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 20–36.

[24] Baoguang Shi, Xiang Bai, and Serge Belongie. “Detecting oriented text in natural images by linking segments”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2550–2558.

[25] Dan Deng et al. “Pixellink: Detecting scene text via instance segmentation”. In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[26] Pengyuan Lyu et al. “Multi-oriented scene text detection via corner localization and region segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7553–7563.

[27] Shangbang Long, Xin He, and Cong Yao. “Scene text detection and recognition: The deep learning era”. In: *arXiv preprint arXiv:1811.04256* (2018).

[28] Zobeir Raisi et al. “Text Detection and Recognition in the Wild: A Review”. In: *arXiv preprint arXiv:2006.04305* (2020).

[29] Minghui Liao et al. “Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes”. In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[30] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[31] Hao Wang et al. “All you need is boundary: Toward arbitrary-shaped text spotting”. In: *arXiv preprint arXiv:1911.09550* (2019).

[32] Juhua Liu et al. “ASTS: A Unified Framework for Arbitrary Shape Text Spotting”. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 5924–5936.

[33] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. "Synthetic data for text localisation in natural images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2315–2324.

[34] Mingxing Tan and Quoc V Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *arXiv preprint arXiv:1905.11946* (2019).

[35] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[36] Ningning Ma et al. "Shufflenet v2: Practical guidelines for efficient cnn architecture design". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 116–131.

[37] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[38] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[39] Hui Zhang et al. "Efficient Backbone Search for Scene Text Recognition". In: *arXiv preprint arXiv:2003.06567* (2020).

[40] Mingkun Yang et al. "Symmetry-constrained rectification network for scene text recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9147–9156.

[41] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. "Training region-based object detectors with online hard example mining". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 761–769.

[42] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[43] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[44] Akhilesh Gotmare et al. "A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation". In: *arXiv preprint arXiv:1810.13243* (2018).

국문초록

다른 기계학습분야와 마찬가지로, 이미지가 담고 있는 문자정보를 얻어내려는 문자인식 분야에서도 딥러닝 이후 많은 진전이 있었다. 인식의 과정은 통상적으로 문자검출, 문자인식의 과정을 차례로 거치는데, 다수의 언어가 혼재할 경우 검출과 인식 사이에 언어분류 단계를 한번 더 거치는 것이 보통이다. 본 연구는 문자 검출 이후의 단계에서 이미지 패치들을 각 언어에 따라 분류하는 것을 목표로 한다. 분류작업만을 전문적으로 다룬 선행연구가 없으므로, 일반적인 객체검출에서 쓰이는 네트워크 중에서 적절한 것을 선택하고 응용하였다. ResNeSt를 기반으로한 네트워크와 자동화된 전처리 과정을 통해 공개된 벤치마크 데이터셋을 기준으로 가장 좋은 기록을 달성할 수 있었다.

주요어: 딥러닝, 문자인식, 문자검출, 다국어 이미지, 분류기, 영상처리

학 번: 2013-23012