저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내야 할 뿐만 아니라 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer]
Deep Learning-Based
Lesion Detection

딥러닝 기반 병변 검출 기법

2018년 8월

서울대학교 대학원
전자-컴퓨터공학부
이성민
Deep Learning-Based
Lesion Detection

지도교수 윤 성 로

이 논문을 공학박사 학위논문으로 제출함
2018년 8월

서울대학교 대학원
전기·컴퓨터공학부
이 성 민

이성민의 박사 학위논문을 인준함
2018년 8월

위 원 장 ______________(인)
부위원장 ______________(인)
위 원 ______________(인)
위 원 ______________(인)
위 원 ______________(인)
Abstract

Advances in hardware and a large number of public datasets have brought about a renaissance of machine intelligence, with successful achievements in a variety of fields, for instance, image recognition, natural language processing, audio synthesis, and future scene prediction. In biomedical data analysis, various studies have also attempted to apply machine learning to effectively analyze biomedical big data, which are massively accumulated.

On the contrary, there are several issues to be overcome to effectively apply machine learning techniques to biomedical data. The first issue is that the rationale of machine predictions must be interpretable for machine learning to be applied as a diagnostic aid. The second issue is that the size of the biomedical data for a particular disease may be insufficient to train models that require large amounts of training data, such as deep neural networks. Furthermore, the lack of annotated data for training of the model can be considered as another issue. In fact, patient data accumulate over time, but the ground-truth data are insufficient to train machine learning-based models. In the case of biomedical data, it is very difficult to obtain ground-truth data because the efforts of a domain expert such as a medical doctor are inevitable. Finally, when vast amounts of biomedical data are analyzed, such as the human genome, the input/output (IO) pattern of the analysis tool can act as a bottleneck and affect the overall analysis time.

In this dissertation, we present the proposed approaches in four chapters to address each issue. First, we describe a pyramid gradient-based class activation mapping (PG-CAM) technique that allows users to understand the rationale behind the
prediction results when using a deep learning-based model as a diagnostic aid. Second, we introduce a method to successfully train a deep learning-based model and improve the robustness of the model in the case of lack of training data. To address the third issue, we propose a novel lesion detection method through a weakly supervised approach. Lastly, we perform an in-depth profiling of 23 bioinformatics applications and its IO pattern analysis through hierarchical clustering to discover IO patterns that can accelerate massive biomedical data analysis in storage devices. This dissertation proposes a variety of machine learning-based analysis and acceleration methodologies for an effective analysis of biomedical data.

**keywords**: machine learning, deep learning, weakly supervised learning, convolutional neural network, lesion detection, medical image analysis

**student number**: 2013-20848
Contents

Abstract iii

List of Figures x

List of Tables xi

1 Introduction 1

2 Background 8

2.1 Modalities of biomedical data . . . . . . . . . . . . . . . . . . . 8
2.1.1 Imaging modalities . . . . . . . . . . . . . . . . . . . . . . 8
2.1.2 Sequence modalities . . . . . . . . . . . . . . . . . . . . . 12
2.1.3 Other modalities . . . . . . . . . . . . . . . . . . . . . . . 14
2.2 Convolutional neural networks . . . . . . . . . . . . . . . . . . . . 14
2.2.1 Layers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
2.2.2 Various CNN architectures . . . . . . . . . . . . . . . . . . . 18
2.3 Major tasks in computer vision . . . . . . . . . . . . . . . . . . . . 21
2.3.1 Object detection . . . . . . . . . . . . . . . . . . . . . . . 22
2.3.2 Semantic segmentation . . . . . . . . . . . . . . . . . . . . . 26
2.4 Transfer learning . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
2.5 Data augmentation . . . . . . . . . . . . . . . . . . . . . . . . . . 29

3 Visualization approach for understandable diagnosis 32

3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
3.2 Methods ................................................................. 35
   3.2.1 Architecture and training .................................. 36
3.3 Experimental results ................................................. 40
   3.3.1 Meningioma MR dataset .................................... 40
   3.3.2 Classification results ........................................ 41
   3.3.3 Localization results ........................................... 42
   3.3.4 Additional results ............................................. 44
3.4 Discussions .......................................................... 45
   3.4.1 The effectiveness of DC-FPN .............................. 45
   3.4.2 Visualization of multi-scale features ...................... 45
3.5 Summary .............................................................. 46

4 Complementary fusion approach for lesion detection 50
   4.1 Introduction ......................................................... 50
4.2 Methods ............................................................... 52
   4.2.1 Preprocessing phase (PP) ................................. 52
   4.2.2 Finger extraction phase (FE) ............................... 54
   4.2.3 Joint detection phase (JD) ................................. 54
4.3 Results and discussion .............................................. 60
   4.3.1 Experimental setup .......................................... 60
   4.3.2 Accuracy of joint detection ............................... 60
   4.3.3 Training CNN and AdaBoost ............................... 61
4.4 Summary .............................................................. 62

5 Weakly supervised approach for lesion detection 63
   5.1 Introduction ......................................................... 63
5.2 Methods ............................................................... 66
   5.2.1 Surrogate ground truth ...................................... 66
   5.2.2 ROI extraction and masking .............................. 70
   5.2.3 Objectness scoring .......................................... 72
   5.2.4 Detection network ............................................. 73
List of Figures

2.1 Examples of biomedical data ........................................ 9
2.2 Various CNN architectures ........................................ 19
2.3 CAM extracted from the ILSVRC 2012 validation data ............ 24
2.4 An overview of transfer learning .................................. 28
2.5 Examples of elastic distortion ..................................... 30

3.1 Overview of the proposed approach ............................... 36
3.2 The qualitative results for our method ............................ 43
3.3 The additional results for our method ............................. 48
3.4 Examples of PG-CAM and PG-CAM extracted from each scale. Each row means the case example. Each column is sorteded in order of original input, PG-CAM extracted from four different scales and integrated version of PG-CAM ........................................ 49

4.1 Examples of finger joint detection .................................. 51
4.2 Overview of the proposed methodology ............................ 53
4.3 Details of JD phase .................................................. 55
4.4 Per phalange epiphyseal/metaphyseal ROI detection using peak finding method ........................................ 58
4.5 Accuracy comparison: FingerNet based on CNN versus AdaBoost ........................................ 60

5.1 Examples of hip X-ray dataset ...................................... 64
5.2 Architecture overview of the proposed approach .................. 67
5.3 Examples of extracted surrogate ground truths .................... 68
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Examples of background occlusion</td>
<td>71</td>
</tr>
<tr>
<td>5.5</td>
<td>The qualitative results of STLDN and existing methods</td>
<td>79</td>
</tr>
<tr>
<td>5.6</td>
<td>Additional qualitative results of STLDN and existing methods on the</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>cerebral palsy validation set</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>Additional qualitative results of STLDN and existing methods on the</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>cerebral palsy training set</td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>A performance comparison of lesion detection with different model</td>
<td>88</td>
</tr>
<tr>
<td>6.1</td>
<td>Performance comparison of three short-read aligners</td>
<td>96</td>
</tr>
<tr>
<td>6.2</td>
<td>Speedup of 23 bioinformatics programs by the drop-in replacement of</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>HDDs by SSDs</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>SSD-based acceleration of bioinformatics pipelines</td>
<td>99</td>
</tr>
<tr>
<td>6.4</td>
<td>Storage feature measurements</td>
<td>103</td>
</tr>
<tr>
<td>6.5</td>
<td>Clustering bioinformatics programs based on the eight storage features list in Table 6.2</td>
<td>105</td>
</tr>
<tr>
<td>6.6</td>
<td>Representative IO traces for each of the five patterns shown in Fig. 6.5</td>
<td>106</td>
</tr>
<tr>
<td>6.7</td>
<td>Impact of randomness on speedup</td>
<td>108</td>
</tr>
<tr>
<td>6.8</td>
<td>CPU-intensive programs in P5 that produce a small amount of IO for</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>moderate-size data</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>Effects of main memory size on SSD-based acceleration of the GATK</td>
<td>111</td>
</tr>
<tr>
<td>6.10</td>
<td>Additional measurements of CPU- and memory-related features for the 23 bioinformatics programs</td>
<td>113</td>
</tr>
<tr>
<td>6.11</td>
<td>SSD-based acceleration of the training of the deep learning-based model</td>
<td>115</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Preset values of the window level and window width for CT .................................................. 10

3.1 Classification results for the meningioma MR validation set ................................................... 42
3.2 Localization performance for the meningioma MR detection set .............................................. 43

4.1 Effects of elastic distortion ........................................................................................................ 62

5.1 Models involved in each phase .................................................................................................. 73
5.2 Validation accuracy of classification network ............................................................................. 75
5.3 The quantitative results of the detection models (IOU $\geq 0.4$) .............................................. 77
5.4 The quantitative results of the detection models (IOBB $\geq 0.5$) ............................................. 77
5.5 Performance comparison of cerebral palsy lesion detection models for IOU criterion range from 0.1 to 0.6 with an interval of 0.1 ...................................................... 81
5.6 Performance comparison of cerebral palsy lesion detection models for IOBB criterion range from 0.1 to 0.9 with an interval of 0.25 ......................................................... 82

6.1 List of the twenty three bioinformatics programs profiled and analyzed in this study ................. 93
6.2 List of storage features used ..................................................................................................... 101
6.3 Storage features measured on the training of the deep learning-based model .......................... 117
6.4 Specifications of the SSDs used in this work ............................................................................ 121
6.5 Specifications of the HDDs used in this work .......................................................................... 121
6.6 List of the data used to test the 23 programs listed in Table 6.1 ............................................... 122
Chapter 1

Introduction

With the breakthroughs in data generation, collection, and analysis technologies, we are living in the era of big data [113]. Novel data-driven research and business opportunities are envisioned in many disciplines, and biomedical engineering is not exception. The recent trend toward precision medicine has triggered the accumulation of a great deal of biomedical data from various sources [90, 39], such as magnetic resonance imaging (MRI), computed tomography (CT), radiographs, (epi-/meta-/pharmaco-) genomics, transcriptomics, proteomics, metabolomics, wearable mobile devices, and crowd-sourced scientific games [62].

Biomedical data derived from patients can be broadly divided into image-based data and sequence-based data, depending on their form. The size of accumulated biomedical data is explosively increasing with every year, and preliminary studies have estimated that the size of biomedical data expected to accumulate by 2019 will be approximately 20,000 petabytes [131]. Image-based data include MRI, CT, radiograph, and microscope image data, and sequence-based data include DNA and RNA data obtained by next-generation sequencing (NGS) [88], and electrocardiogram (ECG) [44] and electroencephalogram (EEG) [95] acquired by healthcare de-
vices. As the amount of accumulated data grows, the need for an efficient and effective analysis method that can find meaningful results from this vast amount of data has increased [85].

In recent years, deep learning has shown successful performance in a variety of areas, such as image recognition and time-series data analysis, and has demonstrated the potential to improve the performance of previously unattainable tasks. Deep learning is no longer limited to image-recognition problems with the MNIST [59] or ImageNet [111] benchmark datasets, but can also be used for new applications that meet people’s practical needs, such as surveillance camera analysis, self-driving car, and medical image analysis [74, 110, 114, 146].

Deep learning has also been extended into a variety of applications in biomedical data analysis, such as diagnosing diseases, detecting the location of lesions, and discovering splice junctions [73, 140, 67]. Indeed, the publication trends in the field of biomedical data analysis show that studies related to deep learning have surged since 2015. In the biomedical field, deep learning-based data-driven approaches have opened up the possibility of solving tasks that are difficult to achieve with satisfactory performance with existing rule-based methods [79]. On the other hand, there exist some limitations to be overcome to apply the deep learning approach for the analysis of biomedical data. Although some of these issues were actually mentioned in previous studies, there are only a few studies dealing with these limitations in depth [140, 24].

In this dissertation, we introduce four issues that exist when applying deep learning to biomedical data and provide four methods on how to handle each issue. In Chapter 3, we describe a pyramid gradient-based class activation mapping (PG-CAM) technique, which is a visually interpretable inference model that captures the discriminative regions that affect model decisions in magnetic resonance images. Chapter 4
presents a novel lesion detection method that combines a deep learning-based model with a signal processing-based model. This method improves the detection performance by the complementary fusion with the conventional method when the training data are too small to train the deep learning-based model. In Chapter 5, we address the issue of lack of strongly annotated data using a weakly supervised detection approach. In Chapter 6, we present a study of input/output (IO) pattern analysis for accelerating the processing methods for biological sequence data through storage devices.

For the diagnosis of the disease through deep learning to be helpful in the decision making of a medical doctor, relevant evidence for reasoning should be presented. It is difficult to interpret which characteristics of the input data derive the model output since the architecture of deep learning is complexly connected by multiple operations such as weighted summation, nonlinear activation, and pooling. No matter how high the test accuracy of the model is, it is still not suitable as a diagnostic aid for medical doctors to help them determine the presence or absence of a disease if the deep learning-based model is uninterpretable. It is therefore critical that domain experts who will use the deep learning-based model can receive intuitive feedback from the biomedical data analysis.

Recent studies [140, 24] in the field of medical image analysis have presented a visual interpretation of the deep learning-based model as a “heatmap,” which directly illustrates an area of the input image, which is then referred through a method called class activation map (CAM) [148]. However, the conventional methods use a typical convolutional neural network (CNN) structure in which the size of the feature map decreases as the depth of the layer increases, resulting in a loss of fine information, which is not suitable for practical use.

Chapter 3 introduces a deep learning-based visual interpretation approach that
overcomes the limitations of previous studies. This model is designed as an encoder-decoder type feature pyramid architecture to avoid losing the spatial information of the input image. Then, the dense connection and the concatenation structure are used to directly transfer the feature map derived from each layer of the feature pyramid to the softmax regression layer. As a result, the model thus designed has the effect of using a multi-scale feature when generating a CAM from the input image, thereby enabling the capture of the fine details of the target object. The contents of this chapter are based on the following research:


As mentioned above, the size of the accumulated biomedical data is large; however, the “well-refined data” that can be used as the training data of the deep learning-based model are relatively insufficient. Although various interpretations are possible, “well-refined” here refers to annotated data with class labels. If the size of the training data is small, it is difficult to sufficiently raise the performance of the deep learning-based model.

In Chapter 4, we utilize domain knowledge as additional information to overcome the lack of training data. The reflection of domain knowledge based on the conventional method complements the deep learning-based model and improves the performance of the overall algorithm. In addition, we can effectively improve the performance of the deep learning-based model by applying a data augmentation technique [119] that transforms an input image with random distortion and uses it as the training data. The contents of this chapter are based on the following research:

- Sungmin Lee, Minsuk Choi, Hyun-soo Choi, Moon Seok Park, and Sungroh

As an extension of the previous issue, we should consider that the deep learning-based model requires the same level of ground-truth data to perform certain tasks. Recently, there have been successful cases of applying deep learning to the analysis of biomedical data [110, 97, 35]. However, such studies are often limited to specific diseases for which benchmark data exist, owing to a lack of strongly annotated data often encountered in applying deep learning to real-world settings. In particular, to perform tasks such as object detection and semantic segmentation, we need strong supervision at the human level (bounding box or pixel-wise annotation). Because biomedical data analysis requires expertise in data annotation, unlike for regular image data, it is more difficult to obtain strongly annotated data. To avoid the need for full supervision, researchers have proposed various approaches that require only image-level category data for training (weakly annotated data) [148]. Training with weakly annotated data, which are relatively easy to acquire compared to strongly annotated data, can be considered as a solution to spread the application of deep learning techniques for the analysis of biomedical data.

In Chapter 5 we propose a weakly supervised learning-based model that accurately detects the target lesions in an image by using only image-level annotation as the training data. The proposed method uses a CAM, which is extracted from the ResNet-based network trained for classification purposes [37], as the pseudo ground-truth to train the region proposal network (RPN) [108], which requires the ground-truth in a bounding-box form. This study demonstrates the feasibility of a weakly supervised approach applied to biomedical data analysis. The contents of this chapter are based on the following research:
Finally, some biomedical data are so large that the time required for the analysis can range from days to weeks. As the analysis is time consuming, various studies on the acceleration of analysis algorithm are being carried out [61, 47]. In general, the purpose of accelerating the algorithm is to speed up the execution time of the operation, and there have been attempts to use various acceleration techniques, such as general-purpose graphics processing unit (GPGPU) and field programmable gate array (FPGA) programming, including parallelization on the multi-core CPUs. On the other hand, no attempt has been made to accelerate through storage. When analyzing a large amount of data, we should not underestimated the storage device characteristics, which can be a bottleneck, depending on the IO pattern of the data access. With the above motivation, Chapter 6 analyzes the data access patterns of various biomedical data analysis tools and measures the speedup that occurs when solid state drives (SSDs) replace hard disk drives (HDDs). In addition, we exploit a clustering technique based on machine learning to discover the IO pattern in which acceleration occurs. We show that the acceleration of the algorithm can be achieved not only by the parallelization but also by the design of the IO patterns, and we find new possibilities for in-storage processing. The contents of this chapter are based on the following research:


The remainder of the dissertation is organized as follows. Chapter 5 provide the methodologies for biomedical image data analysis to overcome each of the con-
straints mentioned above. Chapter 6 presents a study of the IO pattern analysis for accelerating 23 bioinformatics applications. Finally, in Chapter 7 we discuss and conclude the results of the proposed methods and their expected effects.
Chapter 2

Background

In this chapter, we describe the various types of biomedical data and the preliminary work applied to them. In addition, techniques to overcome the lack of training data in biomedical image analysis are also mentioned.

2.1 Modalities of biomedical data

Biomedical data have a unique characteristic that depends on the modality, and it can be classified into imaging modality, sequence modality, and other modality. In this section, various modalities belonging to each category and their characteristics are introduced.

2.1.1 Imaging modalities

Radiography (X-ray): An X-ray that has a strong permeability and can penetrate the inside of an object is an electromagnetic radiation having a wavelength of 10 to 0.01 nm and a frequency of $30 \times 10^{15}$ Hz to $30 \times 10^{18}$ Hz [25]. This electromagnetic radiation is applied to various medical imaging fields such as angiographic examination, computed tomography (CT) examination, as well as simple radiogra-
Figure 2.1: Examples of biomedical data. Biomedical data can be divided into image-type data and sequence-type data. Examples of biomedical image data include X-ray, CT/MRI, and microscopic data. Sequence-based biomedical data typically include genome data and wearable device data such as ECG/EEG.

Radiography is often used to examine bone-like structures and is used to detect a variety of diseases owing to its simple imaging method. In general, X-ray reading is essential to check for bone fracture, and, furthermore, hand radiography is performed to diagnose growth-related diseases in children [124]. In addition, a hip X-ray test is performed to determine the severity of cerebral palsy patients [123] [63]. The simplicity and cost advantages of X-ray imaging make it easier to acquire relatively large amounts of data when compared to other imaging modalities. Recently, a large number of patients’ chest X-ray data and benchmark datasets with disease labeling for each image have also been released for diagnostic automation studies related to deep learning [140].
Computed tomography (CT): CT is an imaging technique that uses X-rays to symmetrically arrange the receptive sensor and the source and rotate it circularly to continuously scan multiple layers of the target object. The basic principle of CT is to reconstruct each voxel value of the original image by using sensor measurement and attenuation coefficient after taking an X-ray from various angles [25]. The voxel of the CT represents the intensity of penetration through the Hounsfeld unit (HU), and the HU differs depending on the properties of the target object. Because of this feature, when performing CT reading, it is necessary to adjust the window level and the window width according to which tissue is to be examined in the preprocessing process. Table 2.1 describes a preset window level and window width preprocessed to analyze the target lesion. By using this preset setting, we can adjust the contrast of the lesion to be read to improve the diagnostic accuracy [41]. Since CT shows the patient’s three-dimensional (3D) information, it is possible to express structures that are superimposed and difficult to identify in an X-ray image. CT is used for the diagnosis of infarction, tumors, calcifications, and hemorrhage [25]. In particular, CT is one of the fastest methods for acquiring data among tomography techniques and is also used in the diagnosis of cardiac-related diseases. CT has been developed to shorten the scan time. There are various types of CT such as fan beam CT, spiral CT, and cone beam CT, depending on the data acquisition method. In recent years, development of CT using multiple detectors (multidetector CT or MDCT) has achieved...
Magnetic resonance imaging (MRI): Unlike X-ray and CT, which utilize the permeability of electromagnetic radiation, MRI is a technique for imaging the interior of an object by measuring the intensity of a signal generated when an external electromagnetic field is applied to a hydrogen nucleus [21]. The hydrogen nucleus in the external electromagnetic field is measured by two types of modalities through the T1 phase, where spin-lattice relaxation occurs, and through the T2 phase, where spin-spin relaxation occurs. Although T1 and T2 are images that can be obtained from a single MRI, the features of the tissue that are visualized in the images are different. In a T1 image, structural features such as soft tissues are visualized. In a T2 image, cerebrospinal fluid and abnormal water-containing regions appear [21]. This characteristic of MRI can be applied as an imaging technique to capture tissue functions such as diffusion-weighted imaging (DWI) and diffusion tensor imaging (DTI), which helps to capture the location of cancer or stroke. In addition, MRI is also effective in visualizing areas of air or areas surrounded by bones that are difficult to identify with conventional CT. However, since an MRI scan takes a long time, the resultant image is distorted in a large moving organ such as the heart or the lung. In recent years, an imaging technique that combines CT and MRI has been introduced to overcome this limitation. MRI is especially effective in brain tumor diagnosis and has been actively used in research on the automation of brain cancer diagnosis, such as the Brain Tumor Image Segmentation (BRATS) challenge [87].

Positron emission tomography (PET): PET is a technique for sensing and imaging positron emission when a contrast agent that binds a radioactive isotope is injected into the body and released according to its half-life. As a contrast agent for PET, radioactive isotopes such as C-11, N-13, and F-18 are used [21]. The positron of the
isotope is mainly detected in the area where the metabolism occurs actively and is used for a cancer test or a functional test of the brain \[27\]. In the case of PET images, the functional aspect of the body is highlighted, although the ability to visualize the anatomical information is weak. Therefore a PET/CT scanner has been developed that combines PET with CT, which has excellent anatomical expression. PET/CT is equipped with separate detectors that detect each signal, and, thus, two images of the same region are acquired at a time. The resulting image is shown overlaid with a heatmap PET image on a gray scale CT image. Because PET is more expensive than CT or MRI, few hospitals have this system, and, thus, the number of data is smaller than that of other modalities \[21\].

**Microscopic imaging:** Microscopy imaging is a technique for visualizing tissues or cells that have been magnified through a microscope. Microscopic images are used in various tasks. For example, microscopic images are used to identify abnormal tissues such as cancer through biopsy and also to measure the degree of DNA damage of individual cells through comet assay \[66\]. As data acquisition is relatively easy, a variety of challenges have been held to effectively analyze the data acquired from the microscope \[16\]. U-Net, a deep learning-based segmentation model, was also proposed to analyze microscopic data \[110\].

### 2.1.2 Sequence modalities

**Nucleic acid sequence:** Next generation sequencing (NGS), which has appeared since 2000, is a method for acquiring the nucleotide sequence of a genome at a high speed and is capable of processing a large amount of DNA fragments in parallel, which is advantageous in terms of processing efficiency compared to the conventional method \[88\]. Owing to the increase in processing efficiency, acquiring genome
data has rapidly increased, and the cost for acquiring such data has also decreased. Depending on the vendor of the sequencing machine, there are various types of platforms such as Illumina, Roche 454, and PacBio. The characteristics of the data obtained from these platforms are slightly different owing to the differences in the sequencing methods used by each company. Various sequencing techniques exist depending on the type of nucleotide sequence to be analyzed, such as, RNA-Seq, and Chip-Seq. The data acquired by NGS are post processed according to the task to be analyzed through alignment, assembly, and mapping. Nucleic acid sequence data are used in studies to discover the genotype of the cancer or in gene editing using CRISPR [55]. In recent years, a variety of methods have been applied, from traditional methods to deep learning-based methods, to effectively analyze this large amount of data [89].

**Biomedical signal data:** As the use of healthcare devices is becoming widespread, studies on the analysis of time-series sequence data obtained from the human body are actively being conducted. There are various types of biomedical signal data, including electrocardiogram (ECG) signals from mobile sensors for measuring heartbeat patterns and electroencephalogram (EEG) signals for measuring the brain activity [15, 50]. Since the data obtained from a healthcare device are in the form of a time-series sequence, they can be applied to the diagnosis of diseases by detecting abnormal patterns or patterns that deviate from the usual patterns of the subject. For example, a study on the use of biomedical signal data was performed to analyze EEG data to detect dangerous EEG patterns in drowsiness [14]. Furthermore, ECG-based biometrics can be applied to authentication tasks and it is expected to be practically used for personal security along with fingerprint, retinal, and vein pattern analysis [15].
2.1.3 Other modalities

**Electronic medical record (EMR):** With the rapid development of information technology, the size of EMR data storing clinical information is also increasing. EMR data store information in a variety of forms, such as text-based sequences, images, and tables, which include the patient’s medical history, health status, and clinical records [91]. The use of EMR data not only promotes administrative efficiency but also enhances the clinical efficacy and quality of medical research. In particular, expert opinion about the patient’s symptoms in the EMR data is used as additional information along with image modality data such as X-ray and CT to help improve the diagnostic performance of the machine [52]. Furthermore, if the performance of the EMR data analysis model is improved, we expect to be able to perform the task of describing the diagnosis of the patient and the associated findings when input images are given.

2.2 Convolutional neural networks

A convolutional neural network (CNN), a variant of the artificial neural network architecture with multiple levels of learning layers, comprises convolution layers, subsampling layers, and fully connected layers, taking advantage of the 2D structure, such as image or video data [60]. Several processes underlie the multi-layer information flow: The convolution layer performs convolution on the input image with an arbitrary filter before a sub-sampling layer sub-samples the results. Additional filtering is performed in the next layer, etc. These processes converge into a feature map that best reflects the features of input images [119].

CNN exploits a local connectivity pattern between the units of adjacent layers to limit errors in the spatial units that do not affect the others with back propagation. By
back-propagating errors through each layer, CNN can optimize the weights to form filters that achieve a minimum error rate. In addition, CNN adopts weight sharing, which assigns the same parametrization to each hidden unit in the same feature map. Applying weight sharing allows CNN to capture the relative rather than the absolute distance between features. This property makes CNN robust, protecting it against distortions or shifts [60].

Thanks to these characteristics, CNN has demonstrated breakthrough performance in computer vision object recognition challenges such as ILSVRC, PASCAL VOC, and MS COCO, and has widened the gap in performance from conventional image processing-based techniques that rely on handcrafted features [111][23][78]. It is not surprising that CNN, which has demonstrated outstanding performance in the field of computer vision, is applied to biomedical image analysis. In biomedical imaging, the number of preliminary studies using CNN in various fields such as tumor detection and cell segmentation is increasing exponentially [79]. In this section, the representative models of CNN, which is the basis of biomedical image analysis, and some layer architectures that can improve the training performance of the model are described.

2.2.1 Layers

CNN is a model that was proposed in the early 1990s but has recently been reevaluated. CNN was noticed at ILSVRC 2012 when AlexNet won with a big gap against the existing methods. AlexNet is a CNN-based model, and its performance has been dramatically improved compared to existing methods because of the increased number of layers in the CNN model [57]. In the conventional method, when the number of layers is increased, the backpropagated gradients gradually decrease, resulting in a problem in which the weight update is not effectively performed. To overcome these limitations, researchers have proposed several innovative layers that allow model
learning even when the number of layers increases.

**Fully connected (FC) layer:** The FC layer is the most basic structure of a neural network. This is mathematically expressed in Eq. [2.1] The output node of the neural network, \( y \), can be derived by passing the weighted sum of the \( i \)-th input node \( x_i \) and the weight \( w_i \) through the activation function \( \sigma \). In general, the offset term \( b \) is also added before the activation.

\[
y = \sigma(\sum_i w_i x_i + b)
\]

(2.1)

In the CNN architecture, the FC layer that considers the connectivity between all nodes is used when classification or regression is performed on features embedded by the convolution layer.

**Convolution layer:** As mentioned above, CNN can consider local connectivity because it uses a convolution layer that applies the trainable weight parameter as a filter to the input. The trainable weights applied to the convolution layer are called filters or kernels. In the convolution operation, the kernel is repeatedly applied and the weight sharing effect occurs. As a result, it is more memory efficient than the FC layer. The effect of the convolution depends on the size and stride of the kernel and can be adjusted to suit the user’s purpose. Recently, various modifications to the convolution layer have been proposed, e.g., a 3D convolution that can be applied to 3D data and a dilation convolution that keeps the size of the kernel while increasing the size of the receptive field.

**Pooling layer:** The pooling layer is used to emphasize a specific signal while reducing the size of the output feature map of the convolution layer [33, 76]. As a
representative type of pooling layer, there are maximum pooling, average pooling, and minimum pooling according to the implementation method. The pooling layer only reduces the size of the feature map and does not affect the number of channels.

**Activation function:** An activation function is a method for assigning nonlinearity to a neural network model, including the sigmoid, rectified linear unit (ReLU), and leaky rectified linear unit (Leaky ReLU) [33]. In the early neural network models, a sigmoid function, as in Eq. 2.2, was used as an activation function. Recently, ReLU and Leaky ReLU, as in Eq. 2.3 and Eq. 2.4 respectively, are widely used to solve the vanishing gradient problem caused by increasing the number of layers of the neural network model.

\[
Sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (2.2)
\]

\[
ReLU(x) = \max(0, x) \quad (2.3)
\]

\[
LeakyReLU(x) = \max(ax, x) \quad (2.4)
\]

**Dropout layer:** Several regularization methods have been proposed to prevent overfitting for effective training of the CNN model. Dropout is one of them, and unlike the method of adding L1 or L2 loss to the loss function, regularization is done by directly affecting the model [33]. Dropout is used to prevent overfitting of a model by stochastically eliminating the possibility of dependence on a specific node that can occur in the learning process [125].

**Batch normalization (BN):** With the increase in training data, various techniques have been proposed to make the training of deep networks fast and effective [126][22]. Examples include using a new activation function (e.g., ReLU), carefully initializing
parameters, and using small learning rates. In recent years, BN [45] has emerged as another remedy. When training a neural network, we often see the internal covariance shift [117] (i.e., the difference between input distributions at different network layers and activations). A solution to resolve this problem would be to perform whitening to make the input distribution have a zero mean and a unit variance. However, whitening normally requires high computational cost.

BN was proposed to address the limitations of the existing approaches. The main advantages of BN are as follows: BN allows us to use higher learning rates for faster convergence; BN can learn a more general model. More specifically, BN replaces each training example $x_i$ in a minibatch by

$$\text{BN}(x_i) = \gamma \left( \frac{x_i - E(X_B)}{\sqrt{\text{Var}(X_B)^2 + \epsilon}} \right) + \beta$$

(2.5)

where $X_B$ denotes a mini-batch input, $x_i$ is the $i$-th element of $X_B$, and $\gamma$ and $\beta$ are the scaling and bias parameters for calibration, respectively. Both $\gamma$ and $\beta$ are learnable parameters.

### 2.2.2 Various CNN architectures

As shown in Fig. 2.2, there are various types of CNN models, among which we would like to introduce four breakthrough models that showed good performance in ILSVRC [111].

**AlexNet:** AlexNet [57] was the winner of the ILSVRC 2012, winning a big gap in performance against the existing methods and becoming the starting point for the CNN model to get attention again. The main reason why AlexNet achieved its breakthrough performance is that it greatly increases the number of convolution layers
Figure 2.2: Various CNN architectures. From left to right are AlexNet/VGGNet, GoogLeNet, ResNet, and DenseNet. AlexNet and VGGNet are models composed of five convolution layers and three FC layers. AlexNet achieved a breakthrough in model performance by increasing the number of parameters in the CNN model. VGGNet is useful for feature extraction of an image by implementing various receptive field sizes of the convolution layer as a $3 \times 3$ filter stack, and it is used in many studies because it is a simple model. GoogLeNet is the first model to suggest an inception layer that is useful for multi-scale feature analysis, and many subsequent studies were inspired by this inception layer. ResNet and DenseNet enable deep model learning by using identity mapping and dense connection, respectively, to solve the vanishing gradient problem in a learning process when the number of layers increases.

and FC layers over existing neural networks. With this design, AlexNet has a vast structure of approximately 650,000 neurons, 60 million parameters, and 630 million connections. Two GPUs were used in parallel to train this huge model, and, as a result, model training with ImageNet data containing 1000 classes was also possible in a short time [57]. In addition, AlexNet has greatly influenced future research using ReLU activation, local response normalization (LRN), dropout, and data augmentation techniques to improve the learning efficiency of the model.
**VGGNet:** VGGNet [120], which was proposed in 2014 and has succeeded in the classification task, has a structure that is advantageous for embedding visual information (see VGGNet in Fig. 2.2). Before VGGNet was proposed, various size filters were applied when designing a network; VGGNet, however, implements it as a hierarchy of $3 \times 3$ filter stacks to enhance the model embedding performance by increasing the number of layers. A $224 \times 224$ size image is fed as an input, which is followed by five convolution blocks and five pooling operations to form a $7 \times 7$ feature map. This feature map, which implies the information of the input image, passes through three FC layers and, finally, to the softmax regression layer, which derives the classification score. The implementation of VGGNet is simple, and, therefore, not only is it applied to classification tasks but also is used as a backbone model for detection and segmentation such as in single shot multibox detector (SSD), DecoupledNet, and fully convolutional networks (FCNs) [80, 96, 82]. However, VGGNet has a large number of parameters, which requires a large amount of GPU memory and takes a long time to train.

**GoogLeNet:** The ILSVRC 2014 winning model, GoogLeNet, is well known for applying the inception layer [127]. In the conventional CNN model, one convolution filter is usually used for one convolution layer. However, the inception layer introduces various types of filters or pooling in one layer to analyze the information of the input image on various scales, and it was a great inspiration for future studies (see GoogLeNet in Fig. 2.2). Despite the proposal of a breakthrough architecture, the implementation of the model is complicated and is not used much in comparison with VGGNet, which appeared at a similar time.

**ResNet:** In a deep learning-based model, as the number of layers increases, the values of the gradients generated in the backpropagation process gradually become...
smaller and, therefore, the model cannot be successfully trained. This phenomenon is called the vanishing gradient problem. ResNet is a model designed to overcome the vanishing gradient problem [37]. ResNet applies a residual connection (identity mapping) to the convolution layer so that the layer before the convolution layer can directly affect the feature map passed through the convolution. As can be seen in Fig. 2.2, the residual connection is a structure that computes the element-wise sums of the feature map from the layer before the convolution and the feature map after the convolution. In the model training, the effect of the residual connection is to prevent the gradient from vanishing in the backpropagation process because the information of the previous layer does not disappear even if the layer is deepened. ResNet has 50-layer, 101-layer, and 152-layer models according to the layer depth, and the 152-layer model has achieved the best performance in ILSVRC 2015.

**DenseNet:** DenseNet uses a skip connection architecture that can directly transfer information from the lower layer to the upper layer to successfully train a much deeper model than the conventional models [42]. The way in which skip connection is applied in DenseNet is called dense connection because it connects all layers as depicted in Fig. 2.2. Since the dense connection transfers the feature map from each layer using the concatenation operation rather than the summation as the identity mapping, the gradient generated from the specific layer can be preserved even if the depth of the model is deep. As a result, the DenseNet model proposes deep model architectures of 100 layers, 190 layers, and 250 layers.

### 2.3 Major tasks in computer vision

The main tasks in computer vision are image recognition, object detection, and semantic segmentation. These three tasks are actively studied not only in the general vi-
sion domain but also in the analysis of biomedical data. In biomedical image data, an image recognition task is performed to diagnose the presence of disease, and, furthermore, a detection task is also performed to identify the area where the patient’s lesion is located. In addition, a semantic segmentation task is performed to automatically measure the size of the tumor present in the image. If the segmentation technique is applied to the follow-up data, the growth of the tumor can be predicted. This section introduces object detection and semantic segmentation techniques in computer vision, which is a preliminary study of the proposed methods.

2.3.1 Object detection

Object detection based on deep learning can be categorized into two types. One is the region proposal-based approach [32, 31, 108]. This approach extracts multiple patches from an input image. After that, the model determines whether an object exists in each patch. The other solves the object detection problem by regression [106, 107, 80]. These approaches directly map from image pixels to bounding box coordinates and class probabilities [106]. In terms of accuracy, the region proposal approach shows better performance, while in terms of speed, the regression-based model is better.

Region proposal network (RPN): The Region Proposal Network (RPN) replaces the conventional method using edge- or superpixel-based region proposal methods with neural network [108]. RPN extracts the region patches where the target object is supposed to be present, and transmits them to the next detection network. RPN applies a sliding window to the feature map from the convolution layer of the CNN model, and sets an anchor that adjusts k scales and ratios to detect various types of objects. Because RPN performs all operations on the feature map that encodes
the characteristics of the input image by learning the inner semantic, not only is the processing speed fast, but also a high detection performance can be achieved with fewer region patches than methods based on selective search [136] or EdgeBox [150].

Weakly supervised object detection

Recently, studies on weakly supervised detection methods have increased. Many have evaluated the objectness score for selected patches of those from outcomes of the conventional proposal method, and finalized detection based on the assumption that there is a target object in the patch with the largest objectness score. If the detection network depends simply on the value, the detection accuracy can be poor. Thus, there have been several attempts to improve this. Jie et al. [49] found object candidates based on the objectness score at the early stage of the detection pipeline, and then expressed them as a graph according to their location. Finally, they selected the most densely-connected patches. In addition, many recent papers based on the objectness score have reported improved detection performance by using the difference of the objectness score as detection evidence caused by the region of interest (ROI) mask-out strategy. This strategy uses the phenomenon that the class probability drops significantly when the region of the target object in the input image is occluded [145]. D. Li [69] and L. Bazzani [7] proposed a detection model by masking out each selected proposals and finding a patch where the score drops significantly. However, when there are several objects of the same class in an input image or when the background is noisy, the method above does not work well [7]. In addition, the mask-out strategy of existing methods is inefficient, because it must be applied to an original image and the inference must be repeated by the number of proposals to classify whether

1The objectness score referred in this dissertation is the regression value by the softmax layer.
Figure 2.3: Class activation maps extracted from the ILSVRC 2012 validation data.
the selected regions contain objects. In this dissertation, we present a methodology to address the issues mentioned above.

**Class activation map (CAM):** Class Activation Map (CAM) [148] showed that the deep learning model discriminates images of different classes by visualizing the activation of the discriminative features in the input image which have high value in the region where the object exists (see Fig. 2.3). Through this approach, it was possible to localize object only with image-level annotation. To draw CAM, the model architecture should be designed with global average Pooling (GAP), one fully connected layer, and a softmax layer, by which CAM is then generated using the Eq. 3.1. As CAM is intuitive, easy to implement, and shows good localization performance, recent studies have used it for detection tasks [40].

**Global average pooling (GAP):** Before the pioneering work [76], most of the previous methods employed FC layers with a nonlinearity function placed in the CNN architecture. However, implementing FC layers in the convolutional layer may give rise to the loss of spatial information. Global average pooling (GAP) [76] was designed to preserve spatial information by enforcing the correspondence between feature maps and categories directly. GAP feeds the average of each feature map into the softmax layer. Given the effectiveness of GAP in preserving spatial information, Zhou et al. used an attention based approach [148], which is related to the perception of information incorporated in an image.

In the literature, saliency has been known for its role in rapid scene analysis [46]. Zhou et al. [148] tried to address the image classification and object localization problems by generating a class activation map (CAM), which refers to the weighted summation of saliency maps to highlight class-discriminative regions. This approach was intended to solve classification tasks by generating a single class activation map.
that consists of various regions, each of which plays a role in classifying the input image. As a CAM is the outcome of multiple layers of convolution and pooling operations, it is often inevitable to see the single CAM having a significant loss in details of the input image.

Object detection for biomedical image

Hitherto, many object detection algorithms proposed for biomedical images are based on strong annotation data of per-pixel or bounding box coordinates. In fact, expensive annotated data are inevitable must be made in order to apply them to new diseases or conditions. To overcome the scarcity of annotation data, we need to exploit a transfer learning method that takes the pre-trained weight of the model learned from general images and fine tunes the model with a small number of medical images. Nevertheless, there is still a limit to this approach. Efforts that include a weakly-supervised manner [140] have been carried out to address this situation.

2.3.2 Semantic segmentation

Semantic segmentation is a computer vision task that assigns predefined semantic labels to each pixel of an image and attempts to parcel out an image into semantically meaningful parts. It is deeply associated with our biological recognition system [149, 28]. For several decades, a variety of semantic segmentation methods have been proposed to understand the human vision system. In particular, CNN-based methods have shown a successful performance in semantic segmentation when combined with other machine learning approaches, such as conditional random field (CRF) and recurrent neural networks (RNNs) [96, 147]. CRF is a graphical model that is the most widely used method for segmentation. CRF has the capability to determine pixel-wise labels using the information of adjacent pixels. For this reason, CNN
combined with CRF has achieved remarkable performance in semantic segmentation \[51, 82, 147, 12\].

For the enhancement of the segmentation performance, there are some approaches to propose the novel network architecture such as the FCN and DeepLab \[82, 12\]. The FCN model replaces the FC layer with a convolution layer to enable pixel-wise prediction regardless of the input size. The DeepLab model uses the dilation convolution layer to improve the segmentation performance by extracting the feature of the input image by considering the relation between the adjacent pixels and the distant pixels. The more complex the architecture of the network, the higher the performance of the semantic segmentation.

**Semantic segmentation for biomedical data**

There are various semantic segmentation methods for biomedical data. Among them, there is a method that derives the segmentation result as an output from a single inference on the input image such as U-Net \[110\]. In addition, pipeline-based methodologies through two-phase inference are also proposed to improve the segmentation accuracy. For example, to accurately segment a tumor present in the liver, a prior study first performed a segmentation of the liver and then a segmentation of the tumor in the detected region \[75\]. On the other hand, since it is difficult to obtain strongly annotated data for segmentation model training, most segmentation method studies for biomedical data are now focusing only on a limited number of diseases in which benchmark data exist \[140\].
Figure 2.4: An overview of transfer learning. Model A pretrained with ILSVRC 2012 data has knowledge learned from the hammerhead shark class included in the dataset. Once pretrained knowledge is transferred in the learning process of model B, it becomes possible to form a model for analyzing human hip joint data with a small amount of data.

2.4 Transfer learning

Transfer learning is a widely used technique in the field of machine learning. It is a method that is applied to overcome the lack of training data for a specific task. In particular, transfer learning is often exploited to train deep learning-based models that require a large amount of data for learning. In the case of applying deep learning to biomedical image analysis, the number of biomedical images is less than that of general images, thus transfer learning enables to overcome the lack of training data \cite{118, 79}. As illustrated in Fig. 2.4, transfer learning is a method that uses the weights of the model trained from the dataset of domain A as the initialization weights for data training of a new domain B whose characteristic is similar to that of domain A. Because the pretrained model is only fine-tuned to domain B, training with a small amount of data becomes possible.
amount of data is possible. To perform transfer learning, we start with the premise that the characteristics of the data in the domain A and domain B should be similar. For biomedical images and general images, there is a difference between using a gray scale and using an RGB color space, and the objects mainly described in the images are also different. However, since these two are image data and basically can be analyzed with a certain pattern such as an edge or a patch describing the object, transfer learning is possible for the weight of the lower layer of the deep learning-based model in which such feature is extracted [118].

Fig. 2.4 is an example of transfer learning for biomedical images. When a hip X-ray image is used as input to model A, which is trained with a general image, model A recognizes it as a hammerhead shark. Here, the hammerhead shark is a knowledge of model A. The heatmap is a CAM for the hammerhead, and allows us to observe which part of the image the model is paying attention to. In the figure, we can see that the model predicts the class with a similar shape to the biomedical image and that transfer learning shows that the CAM for the hip joint X-ray image is activated in a form similar to that of the hammerhead. The above example shows that transfer learning can be useful for the training of the model in biomedical imaging.

2.5 Data augmentation

According to previous studies, when training a deep learning-based model, it has been reported that at least 5000 training data per class are required to achieve an acceptable performance [33]. It is difficult to obtain such a large dataset to train the deep learning-based model in a real-world environment. For this reason, data augmentation is a technique that is essential for the successful training of the deep learning-based model. Commonly used data augmentation techniques include affine
Elastic distortion

Figure 2.5: Examples of elastic distortion. The input was a finger joint image cropped from a hand X-ray image. For the training of the deep learning-based detection model, the amount of data of the finger joint was expanded by using elastic distortion. The elastic distortion randomly migrates the location of each pixel by considering the size of the kernel and that of the Gaussian noise. As a result, the input image is transformed to appear in various forms while maintaining the shape of the joint.

transformations such as rotation, size conversion, and warping [33]. However, data augmentation techniques such as warping can damage the information of biomedical images. Instead, adding uniform random noise [10] or elastic distortion [119], which can increase the amount of data while preserving the image information, is used as a data augmentation technique for biomedical images [110, 151].

**Random noise addition:** This is a data augmentation technique that applies uniform random noise to image data. This method increases the size of the training sample without distorting the object shape and improves the performance of the model by helping the model to be robust against noise [10]. Therefore, the augmentation method of injecting random noise into an image can be useful in biomedical data analysis [151]. In fact, even with the same modality, biomedical data may have different noise distributions owing to the differences in the methods used by imaging device manufacturers, which can be a factor in degrading the performance of the
model. The random noise addition method can thus be expected to improve the generalization performance of the model [10].

**Elastic distortion:** Elastic distortion or elastic deformation is a data augmentation technique that can generate new data while preserving the object shape by randomly moving the location of each pixel for a given image, considering the size of the kernel and that of the Gaussian noise. Elastic distortion was proposed to improve the learning performance of a neural network model for MNIST handwritten data recognition [119]. After that, it has been used in data augmentation for biomedical image analysis and has proved its effectiveness.

Fig. 2.5 shows an example of applying elastic distortion to biomedical images. The input image is a finger joint, which is the criterion of growth-related disease in hand radiograph, and it is used for the training of the deep learning-based model that detects it automatically. A detailed description of the detection methodology is given in Chapter 4. As shown in the figure, when elastic distortion is applied to a given image, the finger joints are deformed into various shapes while maintaining the shape and the number of data can be expanded to hundreds of times. Through the application of the data augmentation technique, it is possible to overcome the lack of training data.
Chapter 3

Visualization approach for understandable diagnosis

3.1 Introduction

A meningioma is one of the most common primary brain tumors in adults [92]. Thanks to its benign nature, the majority of meningiomas grow very slowly, and are diagnosed incidentally without any symptoms. Therefore a “wait and see” policy is recommended to asymptomatic meningioma patients, until there is solid evidence of radiological growth of the tumor or it becomes symptomatic. However, it is sometimes difficult to discern the growth of the tumor from sequential images taken each year without any conscious effort. A volumetric analysis of the tumor is thus indispensable for understanding the growth rate of the individual tumor, which is required to reach appropriate clinical decisions. For an accurate volume measurement, it is necessary to detect the exact position of a tumor. However, manual or semi-automatic detection of tumor is not only inconvenient in daily clinical practice, but also error-prone. Thus, the development of a fully-automated tool for meningioma detection is
expected to have a significant impact on the clinical community if it can guarantee fast and reliable results.

In recent years, deep learning has been extended into a variety of applications in medical image analysis, such as diagnosing diseases, detecting the locations of lesions, or segmenting them [79]. However, the most of such studies are difficult to interpret which features of the input image influence the model output. If the rationale for the decision of the model is not clearly understandable, it is impossible to fully trust the machine as a diagnostic aid. To resolve the above issue, various approaches have been proposed [145, 148, 140]. One such technique is localization using a class activation map (CAM)[148].

CAM performs localization based on the following principles: By applying average pooling to the entire feature map of each unit directly after the final convolution layer, which is called global average pooling (GAP), we preserve the spatial information within the feature map, and then visualize the class-relevant discriminative region(s) in the input image. Target objects can be detected by drawing a tight box around the region. This technique is effective, but also has some limitations. A CAM represents the information from the last layer of a convolutional neural network (CNN) (which should pass through many convolution layers), and has a much lower resolution than the original image. We can only detect an approximate location of an object in the CAM, and some details might have been lost. To address this issue, Grad-CAM, which is a generalized version of CAM, have been proposed [115]. However, a performance improvement to capture the shape of the target object is still required.

An important observation is that the information extracted by each layer in a CNN possesses different characteristics. In lower layers, we can extract low-level features such as lines and edges, while when moving to the upper layers we can observe
high-level patterns (such as wheels or windows in an automobile) by combining the features from the lower layers\cite{145}. A feature pyramid network (FPN)\cite{77} is a model in which the CNN architecture is designed using an encoder-decoder type pyramid structure, so that the feature maps of each layer are intended to form the above-mentioned hierarchical characteristics. Feature maps extracted from each layer of an FPN can be combined with a detection or segmentation model to improve performance. Therefore, we consider a multi-scale CAM approach that can exploit these layer-specific properties of CNN. Different instances of the CAM extracted from different layers will be able to complement one another, often representing the discriminative details of objects to be localized.

In a similar approach, a method of extracting a multi-scale CAM using multiple GAPs has been attempted in previous work\cite{24}, but during the learning process each branch (GAP with skip connection) that connects various layers and a classifier causes an interference, degrading the classification and localization performance. We applied a dense connection\cite{42} to effectively transfer multi-scale feature maps generated from an FPN to a classifier, without using multiple GAPs. A dense connection prevents all information in the lower layer from disappearing, even if the depth of the layer is increased by connecting all the layers with a direct connection.

In this paper, we propose a novel approach to tumor localization based on a pyramid gradient-based class activation map (PG-CAM). The proposed PG-CAM approach can be trained using weakly annotated data. Our model consists of densely connected layers, each of which transfers multiple scale information without loss. Consequently, PG-CAM can generate CAMs from various scales. These outputs are aggregated in the final map (i.e., PG-CAM), which can robustly capture tiny details of an object that would otherwise be missed using an original CAM. The proposed PG-CAM model does not require time-consuming processing to generate patches by
using a window-based selective search, significantly expediting the inference procedure.

Our specific contributions include the following:

- We propose a pyramidal fusion of gradient-based class activation maps, with the final results culminating in a single map called the \textit{pyramid gradient-based class activation map} (PG-CAM). The proposed model only needs weakly annotated data for training. In our experiments, this PG-CAM approach delivered superior performance compared to the existing single-step Grad-CAM approaches compared.

- We evaluate the effectiveness of using multi-scale features in a CNN and present results from various ablation experiments. In addition, we show the feasibility of feature pyramid network as a classification model.

- For effective training, the proposed method uses the batch normalization directly after concatenation operation performed, because dense connection and concatenation operation can affect the distributions of activated value from various layers.

3.2 Methods

Here, we describe further details of the proposed PG-CAM approach, which is motivated by the single CAM architecture for object localization tasks\cite{148} and FPN\cite{77} for multi-scale feature extraction. Fig. 3.1 illustrates the P-CAM architecture, which consists of an FPN component and a CAM component. The dense connection of the FPN’s respective layers allows that the feature map, which corresponds to all scales of the feature pyramid, is directly delivered to a classification layer. Therefore, P-
Figure 3.1: Overview of the proposed approach. The blue, yellow, green, and gray arrows indicate convolution, convolution after max pooling, deconvolution, and the fully-connected layer after GAP, respectively. The black dotted arrows represent the direct connection, and the dotted blue line separates the scale of the network.

CAM becomes an activation map in which all scale features of the feature pyramid are integrated.

3.2.1 Architecture and training

Grad-CAM: CNNs are capable of representation learning that can extract significant features from the data. To understand this capability, there have been studies analyzing the visual encoding process of CNNs. The notion of CAM [148] originated from such efforts. CAM can highlight the regions identified by a CNN for performing classification. In traditional CNNs, the fully connected (FC) layers can be used to formulate relationships between elements appearing in feature maps. However, in this process spatial information is preserved until directly after the last convolution layer can be lost.
To resolve this issue, a GAP layer is inserted directly before the fully connected layer [76]. Then, the input to the softmax function for a category $c$ and unit $k$ can be converted to a CAM by using simple distributive law. CAM is described as follows:

$$CAM_c = \sum_k w_c^k f_k(X, Y)$$

(3.1)

where $f_k(X, Y)$ represents the $k$-th feature map in the last convolution layer at the spatial location $(X, Y)$.

Grad-CAM [115] is a generalized version of CAM. The biggest difference between Grad-CAM and CAM is that Grad-CAM uses the gradient $\frac{\partial L^c}{\partial f_k}$ of the feature map $f_k$ with respect to the class loss $L^c$ when calculating the weight for the feature map. The assumption in the Grad-CAM is that the gradient associated with the class can act as a weight. Thus, it is possible to extract the CAM not only from the feature map immediately before the GAP layer, but also from the existing all feature maps of the convolution layer. Grad-CAM is expressed by following equation EQ 3.2. As describe in EQ 3.2, weight values are obtained by taking an average for a gradient existing in each feature map.

$$GradCAM_c = ReLU(\sum_k \sum_i \sum_j \frac{\partial L^c}{\partial f_{kij}} f_k(X, Y))$$

(3.2)

where $i$ and $j$ denote the spatial location of gradient in feature map $f_k$. In Grad-CAM, ReLU activation is applied to the extracted CAM. ReLU activation is used to select information activated with a positive value, because of a Grad-CAM may have many negative activated values. In this study, we propose a method of extracting CAM more precisely than previous methods by utilizing the advantages of Grad-CAM which enables CAM extraction from any layer and the advantages of encoder-
Densely connected feature pyramid network: The proposed densely connected feature pyramid network (DC-FPN) consists of a CNN-based encoder-decoder. This is a network designed to form a hierarchical feature map which is represented in each layer. The encoder and decoder of DC-FPN consists of a symmetrical structure with the same number of channels and stride, and are connected by a direct connection on the same scale so that structural characteristics are preserved on each scale. The overall structure looks similar to a U-net [110], but all layers are connected by a dense connection, and the number of channels in each layer has also been modified accordingly. As shown in Fig. 3.1, at the joint where a dense connection is applied, feature maps are concatenated, and batch normalization (BN) [45] is applied to adjust the distribution of the activated values. The feature maps of the previous layer preserved by the dense connection are transferred to the final layer of the network. Although, U-net, which is the backbone architecture of FPN, is not a suitable structure for classification, the proposed method using dense connection that exploits concatenation and BN can successfully train it. Experimental results related to this effect will be presented in the ablation study of Table. 3.1.

Pyramid Grad-CAM: The principle of obtaining a Pyramid Grad-CAM is expressed mathematically as follows: it is difficult to accurately track the information flow of the multi-scale features because the number of channels for feature maps is reduced during the upsampling process. For clarification, we thus represent the concept of our proposed method with a simple equation. The symbol for the feature maps at the end of each scale is denoted as \( \{ f_{s^p} \}_{p=1}^{n} \). \( p \) and \( n \) are an index of the scale and the number of scales, respectively. Dense connections have the effect that each feature map is concatenated, resulting in the feature maps being passed to the classifier.
as $concat(f^{s_1}, ..., f^{s_n})$. Therefore, PG-CAM can be described as in Eq. 3.3

$$PGCAM_c = \sum_{p=1}^{5} ReLU(\sum_{k_p} \sum_{i} \sum_{j} \frac{\partial L_c}{\partial f_{kp}^{s_p}} f_{kp}^{s_p}(X,Y)) = \sum_{p=1}^{5} GradCAM^{s_p}_c \quad (3.3)$$

where $k_p$ is the unit of the feature map $s_p$, $GradCAM^{s_p}_c$ is a GradCAM generated from the feature maps delivered from each scale $p$, and the PG-CAM aggregating the GradCAMS will consequently contain information on multiple scales of the feature pyramid. The dense connection of PG-CAM reduces the classification performance degradation, as shown in a previous study [24] that tried to extract a CAM for a multi-scale feature map by using multiple GAPs, where the performance boost achieved by adopting PG-CAM was about 8.7 percent point. In addition, it will be shown by our experimental results that using PG-CAM in the tumor localization task is effective for capturing subtle details of the target object, which are missed for a single CAM.

As mentioned in the above paragraph, the PG-CAM merges the information of the lower scale layer during the upsampling process and transfers it to the feature map of the next scale. Therefore, it is difficult to trace exactly which features affect the classification result. In addition, even if the information in the lower layer is sent through a concatenation operation, the effect of the feature at the smallest size can be weakened. (The details about this issues will be discussed in Section [3.4.2]) On the other hand, Grad-CAM has a property to visualize the feature map associated with the class from any convolutional layers of the network using gradient generated by backpropagation. In this chapter, we also propose a complementary integration of multi-scale PG-CAM.
**Loss function and optimization:** To perform localization, P-CAM only requires image-level annotations of whether a tumor exists. Therefore, we train our model with a softmax regression loss for binary tumor classification. P-CAM exploits ADAM optimization \([56]\). At this time, \(\beta_1, \beta_2\) and \(\epsilon\) were 0.9, 0.999 and \(10^{-8}\), respectively. The base learning rate started at 0.001 and was multiplied by 0.1 for every 10,000 iterations.

**Object localization:** A discriminative region in the input image tends to have a high CAM activation score between zero and one. We empirically set the threshold for detecting a tumor in the P-CAM to 0.5 of the CAM activation score and the ResCAM to 0.8 of the CAM activation score. We represent each localized tumor using a bounding box.

### 3.3 Experimental results

We used a brain magnetic resonance (MR) image dataset of 550 patients with meningioma, collected from a hospital to which one of the authors belonged. We performed our presented experiments using a Linux machine running CentOS, equipped with four P100 (GDDR5 15GB) GPU cards. All experiments are implemented on the Caffe\([48]\).

#### 3.3.1 Meningioma MR dataset

This study was approved by the institutional review board of a collaborating hospital, who waived informed consent. Each patient underwent up to 12 follow-up examinations, and the resulting images were composed of slice images of axial, coronal, and sagittal axes with T1 modality. The training data used to train the classification model consists of 171,107 image slices of 500 patients, and the validation data consists of
18,322 image slices of the remaining 50 subjects. All image slices are resized and center-cropped to fit the receptive field size (224 × 224 pixel lengths). Two neurosurgeons read all the image slices, and annotated the presence of tumors. The ratio of the image slices with tumors to without tumors is imbalanced at approximately 85:15. In order to measure the localization performance, regions where a tumor was located were annotated in the form of a bounding box on 188 axial images for eight patients in the validation set.

### 3.3.2 Classification results

As shown in Table. 3.1 to confirm the effect of the feature pyramid and dense connection, an ablation study was conducted on the classification performance. The models compared in this experiment are PG-CAM, ResNet50-based single CAM (ResCAM) [37], ResNet50-based single Grad-CAM (Grad-ResCAM) [115] and Multi-GAP CAM [24]. As a result of the dense connection, the classification accuracies of PG-CAM and Multi-GAP CAM are improved by 12.3 and 2.9 percent points, respectively. In the structure with the dense connection, the accuracy of the CAM model with multiple GAPs was 13.0 percent points lower than that of the PG-CAM model. This is because when the skip connection is used to deliver feature maps located on each scale to a classifier, these interfere with each other during the learning process, thereby hindering a stable weight learning. On the other hand, ResCAM without a feature pyramid and dense connection showed the best performance, with a 96.4 percent accuracy, because this model is 28 layers deeper than other models using an FPN. Although the number of layers is relatively small, the difference in classification performance is only 0.8% due to the effect of dense connection. Since ResCAM and Grad-ResCAM utilize the same ResNet structure, their classification accuracy is equal. Interestingly, PG-CAM (without a dense connection) and Multi-GAP CAM
Table 3.1: Classification results for the meningioma MR validation set. We show the ablation study results to demonstrate the effect of the feature pyramid network (FPN) and dense connection (DC). A comparison of the performance of PG-CAM and Multi-GAP CAM showed the effectiveness of dense connection. ResCAM (Grad-ResCAM) showed the best classification performance due to the number of layers, however the difference is negligibly small. Although the number of layers is relatively small, the difference in classification performance is only 0.8% due to the effect of DC. On the other hand, in the localization experiments, the model using FPN recorded higher performance than ResCAM and proved its effectiveness.

<table>
<thead>
<tr>
<th>Model</th>
<th>FPN</th>
<th>DC</th>
<th># of Layers</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG-CAM (Our method)</td>
<td>O</td>
<td>X</td>
<td>22</td>
<td>86.3%</td>
</tr>
<tr>
<td>PG-CAM (Our method)</td>
<td>O</td>
<td>O</td>
<td>22</td>
<td>95.6%</td>
</tr>
<tr>
<td>Multi-GAP CAM [24]</td>
<td>O</td>
<td>X</td>
<td>22</td>
<td>79.7%</td>
</tr>
<tr>
<td>Multi-GAP CAM [24]</td>
<td>O</td>
<td>O</td>
<td>22</td>
<td>82.6%</td>
</tr>
<tr>
<td>ResCAM [37]</td>
<td>X</td>
<td>X</td>
<td>50</td>
<td>96.4%</td>
</tr>
<tr>
<td>Grad-ResCAM [115]</td>
<td>X</td>
<td>X</td>
<td>50</td>
<td>96.4%</td>
</tr>
</tbody>
</table>

(both the models with and without a dense connection) have been trained to classify most images as without tumors, due to being affected by the class imbalance of the training data, even though the achieved classification accuracy was over 80 percent.

### 3.3.3 Localization results

We analyzed the localization performance of PG-CAM. To this end, we assessed the quality of localization by measuring the IOBB values between an extracted bounding box and the ground truth. If the extracted bounding box had an intersection over the detected bounding box ratio (IOBB) value of over 0.2, then we declared it a success. Although the IOBB value of 0.2 may be considered low, however there are cases where the threshold is adjusted for evaluation in the case of a medical image in which the boundary is ambiguous even when reviewed by humans [43].

As can be seen from Table. 3.2, we compared a PG-CAM model with a dense connection, which achieved the best performance among our models, with ResCAM
Table 3.2: Localization performance for the meningioma MR detection set. PG-CAM means the integrated result of Scale1 to Scale4. PG-CAM:Scale1 and PG-CAM:Scale1&4 represent PG-CAM extracted from only Scale1, and PG-CAM extracted and integrated from Scale1 and Scale2, respectively. Although, ResCAM and Grad-ResCAM share the backbone structure outputs are extracted by using different equation (Eq. [3.1] and Eq. [3.2]).

<table>
<thead>
<tr>
<th>Model</th>
<th>DC</th>
<th>Precision</th>
<th>F1 score</th>
<th>PR-AUC (AP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG-CAM (Our method)</td>
<td>O</td>
<td>0.45</td>
<td>0.50</td>
<td>0.48</td>
</tr>
<tr>
<td>PG-CAM:Scale1 (Our method)</td>
<td>O</td>
<td>0.42</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>PG-CAM:Scale1&amp;4 (Our method)</td>
<td>O</td>
<td>0.47</td>
<td>0.53</td>
<td>0.54</td>
</tr>
<tr>
<td>ResCAM [37]</td>
<td>X</td>
<td>0.13</td>
<td>0.22</td>
<td>0.18</td>
</tr>
<tr>
<td>Grad-ResCAM [115]</td>
<td>X</td>
<td>0.26</td>
<td>0.39</td>
<td>0.40</td>
</tr>
</tbody>
</table>

Figure 3.2: The qualitative results for our method. The left image shows the original input with a bounding box annotation indicating the tumor location (red). The green and blue box represent the predicted tumor locations using PG-CAM and ResCAM respectively. The other two images demonstrate the results extracted by PG-CAM (middle) and ResCAM (right).

(Grad-ResCAM), which achieved the best performance among all models. In our settings, we tested three version of PG-CAM. PG-CAM is the fused version of PG-CAM extracted from every scales, and PG-CAM:Scale1 represents single PG-CAM extracted from Scale 1. PG-CAM:Scale1&4 is the integration of two PG-CAM of Scale 1 and Scale 4. It is possible to combine various scales, but the reason for selectively using only the PG-CAM generated from Scale 1 and Scale 4 is because they exhibit the most different characteristics. Further details about the multi-scale char-
acteristics of PG-CAM are covered in the Section 3.4.2. Unlike in the classification experiment, every version of PG-CAM outperformed ResCAM and Grad-ResCAM in terms of the precision, accuracy and F1 score in the localization task. This is because, as shown in Fig. 3.2, unlike ResCAM, which considers the approximate position of a tumor, PG-CAM considers multiple-scales to robustly detect the detailed position of a tumor. In particular, the PG-CAM:Scale1&4 showed the best localization performance, which was caused by the complementary fusion of two PG-CAMs with different characteristics.

The precision and PR-AUC (AP) values of the models may seem relatively lower than those of methods using strong supervision. However, considering the fact that these two models detect the tumor without learning the bounding box annotation, and that the accuracy of the state-of-the-art weakly supervised method[148] for the localization task for a generic image set is 43.6 percent, the localization performance of PG-CAM for the meningioma MR dataset is acceptable.

3.3.4 Additional results

We provide additional results for qualitative/visual inspection of the localization results performed by PG-CAM and Grad-ResCAM. In Fig. 3.3 we show (from left to right), the original image, the output of the PG-CAM, and the output of Grad-ResCAM. Three colors of bounding boxes in the original image are indicating the tumor location (red), the predicted tumor locations using PG-CAM (green), and the predicted tumor locations using Grad-ResCAM (blue). We observed that Grad-ResCAM only captures approximate location of the tumor. On the other hand, we discovered that the PG-CAM successfully detects sophisticated regions even if the tumor is small (see first row in Fig. 3.3), thus improving localization performance.
3.4 Discussions

3.4.1 The effectiveness of DC-FPN

As can be seen from the experimental results, a very deep model with a large number of layers, such as ResNet, is advantageous for performing classification tasks. However, to visualize or detect a lesion it is advantageous to use the densely connected feature pyramid architecture, even if the depth of the network is shallow, because the details of the image to be analyzed can be robustly captured. For example, in the case of Grad-ResCAM, the size of the CAM extracted from the last feature map is reduced to being 64 times smaller than the input image, through several pooling operations. Therefore only the approximate position is captured, as seen for Grad-ResCAM in Fig. 3.2. On the other hand, the feature maps of each scale for an FPN are upsampled by the deconvolution layer to fit the size of the upper scale, and finally become the same size as the input image. Thus, the size of the extracted PG-CAM is the same as the size of the input image, and it is possible to specify the shape of the tumor in detail. To summarize, compared with the Grad-ResCAM method, our proposed PG-CAM approach shows more robust localization performance with only a negligible sacrifice in classification accuracy.

3.4.2 Visualization of multi-scale features

This section discusses the validity of using multi-scale features by comparing and analyzing PG-CAM extracted from different scale. Furthermore, we consider a combination of multi-scale features that can enhance the shape of the tumor captured by PG-CAM. Fig. 3.4 shows the PG-CAM extracted from each scale of FPN. The outputs are illustrated in the order of the fine to coarse scale of the feature map from the left to the right. Each PG-CAM is represented by the pixel lengths of 224×224,
112×112, 56×56 and 28×28, respectively. To integrate them, we resize each of PG-CAM to the pixel lengths of 224×224 (see the rightmost picture in Fig. 3.4). As shown in Fig. 3.4 PG-CAM extracted from Scale 1 through Scale 3 demonstrate similar activated regions. There is a slight difference between Scale 1 to 3: In PG-CAM Scale 1, the attention is more on the area where the tumor is located, whereas in PG-CAM Scale 2 and 3, the skull and the structural part of the brain are also activated. Unlike the other three scales, in PG-CAM Scale 4, which captures the discriminative region on the most coarse scale, strong activation is observed in a specific part of the tumor location. From these results, we find that the features captured on multiple scales are slightly different for the input image as described in this paper, and we can expect that the target lesion can be characterized in detail by using PG-CAM. For this purpose, it is possible to combine PG-CAM extracted from all scales into one, and further, a method of selectively fusing scales having the most different characteristics can be considered, such as fusion of PG-CAM Scale 1 and PG-CAM Scale 4 (see localization performance of PG-CAM:Scale1&4 in Table. 3.2).

3.5 Summary

In this chapter, we have proposed a robust localization method named pyramid gradient-based class activation mapping (PG-CAM). Through an ablation study, we considered and evaluated the effectiveness of the dense feature pyramid architecture from the viewpoint of tumor localization. In our experiments, the proposed PG-CAM approach delivered outstanding performances in detecting tumors in input images, achieving a 23 percent point precision boost for the meningioma MR dataset compared with the state-of-the-art single Grad-CAM method. We anticipate that the PG-CAM model, which is trainable in a weakly supervised manner, will be applicable
to medical image analysis, which often suffers from a lack of annotated datasets. Furthermore, we expect that it will be possible to utilize the extracted PG-CAM as an effective and efficient preprocessing method for downstream analysis, such as in tumor segmentation.
Figure 3.3: The additional results for our method. The left image shows the original input with a bounding box annotation indicating the tumor location (red). The green and blue box represent the predicted tumor locations using PG-CAM and Grad-ResCAM, respectively. The other two images demonstrate the results extracted by PG-CAM (middle) and Grad-ResCAM (right).
Figure 3.4: Examples of PG-CAM and PG-CAM extracted from each scale. Each row means the case example. Each column is sorted in order of original input, PG-CAM extracted from four different scales and integrated version of PG-CAM.
Chapter 4

Complementary fusion approach for lesion detection

4.1 Introduction

Finger joints play an important role in hand radiograph assessment. Due to the simplicity of the procedure and limited exposure to radiation, hand radiographs are typically used to assess the finger bones and joints to carry out bone age evaluation for children and adolescents, and to diagnose growth disorders, inherited disease \[129\] and joint inflammation, such as rheumatoid arthritis. Because of the large number of joints in the hand and the frequent use of this type of examination, it is beneficial to automate joint detection in hand radiograph images. Fig. 4.1 illustrates an example of finger joint detection.

Since Pal and King proposed the first automated system for detecting joints in hand radiographs \[100\], several advanced methods, such as an automation system by Giordano \[30\] and computer-assisted bone age assessment technique by Pietka \[102\], have been developed. However, medical studies \[138\] indicate that these methods
require improvements in accuracy, reliability and consistency to cultivate full-scale acceptance into clinical fields.

A recent breakthrough, deep learning is gaining popularity for its excellent performance in various recognition problems. For example, deep-learning architectures have achieved the best performance in various competitions in natural language processing and image recognition. Specifically, convolutional neural networks (CNN) [60], a variant of deep-learning architecture, is widely used in computer vision, such as detection or recognition, where systems have produced state-of-the-art results when performing a variety of tasks. The excellent performance of CNN in other applications motivated us to adapt CNN architecture for finger joint detection. In this chapter, we propose a user-interactive CNN-based joint detecting system called FingerNet. The proposed method performs binary classification via CNN to each windowed image sub-block to determine whether it is a joint. The performance of the proposed method is compared with that of AdaBoost [139], which is another state-of-the-art approach to face detection. FingerNet outperformed AdaBoost, achieving 98% accuracy compared to AdaBoost’s 92% in 130 test data sets.

Figure 4.1: Examples of finger joint detection. (a) Input hand radiograph. (b) Extracted fingers and detected finger joint regions. (c) Joint bone maturity.
4.2 Methods

Fig. 4.2 is a flow chart demonstrating the proposed algorithm. FingerNet consists of three stages: preprocessing (PP), finger extraction (FE) and joint detection (JD).

4.2.1 Preprocessing phase (PP)

In the first phase (PP), hand mask segmentation is performed in the radiographic image. The extracted mask acts as a base for the latter FE phase.

In the PP.1 and PP.2 steps, hand radiograph image intensity adjustment and noise reduction are performed respectively. Prior to any further processing, all input images are resized to $1,400 \times 1,200$ pixels, which is a reasonable size for a radiograph. Results from statistical analysis are used for intensity adjustment and a Wiener filter [141] is applied for denoising (PP.2). The Wiener filter suppresses background noise without the loss of information in the original image due to the use of minimum mean squared error as its base.

Step PP.3 and PP.4 are carried out to detect edges in the image and separate the hand mask from the image background. FingerNet uses a gradient-based edge detector and returns an image that represents the intensity gradient of the input image. The resulting grayscale image consists of information regarding the presence of edges. Background is separated using Otsu’s method [99], which divides signal and background by maximizing the separability of the gray level distribution.

In step PP.5, dilation, erosion and fill methods are performed for postprocessing, and labels and noise in the background are removed to return a binary hand mask.
Preprocessing, PP

PP.1: Intensity adjustment
PP.2: Wiener filtering (Denoising)
PP.3: Edge detection
PP.4: Otsu's threshold (Separation)
PP.5: Fill areas (Segmentation)

Finger extraction, FE

FE.1: Watershed clustering
FE.2: User-Interactive mode selection
FE.3: Grouping adjacent clusters
FE.4: Extend each finger regions

Joint detection, JD

JD.1: Sliding window
JD.2: Classification using CNN (Fig.3)
JD.3: Joint peak detection (Fig.4)
JD.4: Merging detected regions (Fig.4(e))

Figure 4.2: Overview of the proposed methodology.
4.2.2 Finger extraction phase (FE)

The FE phase extracts five separate fingers from the hand mask image. This process is effective in enhancing detection performance because the search space of the JD phase can be limited and fingers are always vertically aligned.

The FE.1 step involves manual user interaction to mark the end-point location of the thumb and fifth finger, as well as the forks adjacent to those two fingers. From this information, FingerNet calculates the rotation angles of the fingers, which are used in FE.3 and FE.4 (see Fig. 4.2).

The FE.2 step extracts each finger from a hand mask using the watershed algorithm [137], which is a region-based segmentation method that segments an image into a number of fragments based on a watershed where the pixel values surge.

In FE.3, the adjacent fragments are merged into large clusters based on distance. As a result, a hand is divided into fingers, palm, and others made up of carpal bones and the two long bones within the wrist. The determination of the position of fingers is relatively reliable due to their location in the extremity of the hand. If the hand is rotated excessively or the thumb is overly stretched and located under the palm, manual markings by the user, acquired in FE.1, are used to compensate.

The final step, FE.4, calculates the central axis and the rotation angle of each finger region and extends the regions of fingers to a predetermined distance using the direction of the rotation angle as a seed.

4.2.3 Joint detection phase (JD)

In the JD phase, three joints are detected for each finger image by combining two different approaches: machine learning-based CNN and signal processing-based joint peak detection.
Figure 4.3: Details of JD phase. (a) Applying sliding window. (b) Image patches extracted by (a). (c) CNN architecture. (d) Merging detected regions and joint peaks. (e) User interaction step. User can confirm and adjust the results.
**CNN based joint detection**

Fig. 4.3 shows the procedure for CNN-based joint detection. To extract sub-block images from the finger images, FingerNet applies a sliding window of $32 \times 40$ pixels to the images so that it reflects the ratio of the joint, which is measured empirically (Fig. 4.3(a)). The process is repeated for two additional iterations where window size is scaled 1.2 times. Extracted sub-block images are resized to $28 \times 28$ pixels. FingerNet detects joints by performing binary classification to evaluate whether each resized sub-block image is a joint using a CNN model. Additional training samples were created by elastic distortion [119], which generated approximately 1.3 million joint images (see Section IV.A for further details).

The CNN architecture used in FingerNet is optimized and customized based on LeNet-5 [60]. The classification of grayscale joint images resembles the classification task of the MNIST handwritten digits [60], and the sliding window of this method is similar in size to the input data of LeNet-5.

The architecture of the CNN model consists of five layers: two convolution layers, two sub-sampling layers and a fully connected layer. As shown in Fig. 4.3(c), the convolution operation with a $5 \times 5$ square kernel is applied to the joint images of $28 \times 28$ and then connected to convolution layer C1 with six feature maps. Those six feature maps are connected to upper layer S1 after being sub-sampled to half size using the average pooling method. Another $5 \times 5$ convolution filter is applied and the outputs are weighted summed to generate 12 feature maps for convolution layer C2. Through $2 \times 2$ average pooling, C2 is connected to sub-sampling layer S2. Finally, it is connected to fully connected layer F1, which is composed of 192 output neurons. The F1 layer linearly sums up output neurons to present an output that represents the probability of the input image being a joint. This procedure is the feedforward part of FingerNet. Furthermore, FingerNet transfers classification errors to lower layers.
using a back-propagation algorithm and iterates the above processes to update the weights of layers.

**Joint peak detection**

The joint peak detection method used in FingerNet is an image signal processing approach that scans each finger image from the end of the tip downward (in Fig. 4.4(a), left to right) to find peak locations in which intensity changed dramatically. Fig. 4.4(b) represents the intensity integrated over the finger width (vertical) direction of Fig. 4.4(a). Fig. 4.4(c) indicates the difference of Fig. 4.4(b) to clearly show the location where the integrated intensity changed significantly in the form of peaks. In Fig. 4.4(d), Profile \( (x) \) represents the amplified and denoised intensity curve resulting from absolutization and median filtering.

The positions marked with a red dotted line in Fig. 4.4(d) are the peak locations where joints exist. The location of a peak, \( x_{pd}^* \), is given by

\[
x_{pd}^* = \arg \max_{x \in N} \{ \text{Profile}(x) \}
\]  

(4.1)

where \( x \) is the horizontal coordinate of Fig. 4.4(d) and \( N \) is the window size for peak location search. FingerNet gives local maxima for all examined \( x \) from Eq. 1.

By means of thresholding and hierarchical clustering for postprocessing, it is possible to predict the joint locations \( x_{pd}^* \). In this process, thresholding fulfills the role of denoising through exclusion of locations where the magnitude is less than a predetermined value and the clustering algorithm merges adjacent peaks. Lastly, the first three peaks from joint candidates are chosen because several peaks may exist at the fork where fingers are split. This method enhances the precision of an existing peak detection method [30], which simply estimates joint locations based on thresholding.
Figure 4.4: Per phalange epiphyseal/metaphyseal ROI detection using peak finding method. (a) Original image. (b) Integrated intensity. (c) Intensity gradient. (d) Absolutized and median-filtered intensity gradient. (e) Flow chart of merging detected regions.
Merging detected regions

Fig. 4.4(e) represents the flow chart of merging detected regions. Between the two algorithms, the CNN-based method leads the detection of finger joints. The joint regions detected by CNN, distributed in contiguous regions, are merged into a union, which is a rectangle covering all widths and heights of the elements (see Fig. 4.3(d)). Disconnected patches captured by false positive errors are removed and the third joint region is expanded downward through the adaptation of domain knowledge to include the metacarpal.

The results from the joint peak detection method complement the results of the CNN model. This is especially useful, as the CNN model is prone to yield detection faults when the intensity of sub-regions in the radiographic image is too low. The joint peaks further contribute to the improvement of detection specificity by double checking the outputs of the CNN model. Ultimately, the final peak location $x^*_{fn}$ was defined as

$$
x^*_{fn} = \begin{cases} 
x^*_cnn & \text{if } n(x^*_cnn) = 3 \\
x^*_cnn \cup x^*_pd & \text{if } n(x^*_cnn) < 3 \\
x^*_cnn \cap x^*_pd & \text{if } n(x^*_cnn) > 3
\end{cases} \quad (4.2)
$$

where $x^*_cnn$ denotes the joint regions detected by CNN.

The rule defined in Eq. 2 can be described as follows. If the element number of joint regions detected by the CNN model $n(x^*_cnn)$ is less than three, FingerNet deploys peak detection results to make up for the missing joints. The missing joint is determined by finding $x^*_pd$ farthest away from elements of $x^*_cnn$. In case the number of locations detected by CNN is more than three, FingerNet adopts the coincided regions of $x^*_cnn$ and $x^*_pd$. The results ($x^*_{fn}$) as seen in Fig. 4.3(e), can also be adjusted through user interaction.
Figure 4.5: Accuracy comparison: FingerNet based on CNN versus AdaBoost.

4.3 Results and discussion

In this chapter, aimed at evaluating the performance of CNN-based joint detection, the detection accuracy of CNN-based joint detection is compared with that of AdaBoost, which is well known for breakthroughs in face detection [139].

4.3.1 Experimental setup

The experiment was performed on an INTEL i5-2500 (3.3GHz quad-core), 16GB main memory and MATLAB version 2014A. Joint detection accuracy is known to be affected by subjects’ age due to the change in joint shapes as people age. To test the robustness of the method, the researchers thus deliberately selected three different age groups and prepared 130 left hand radiographs in DiCOM format.

As golden standard, a professional clinician manually extracted 3,000 aligned middle phalange images of second, third and fourth fingers from 1,000 hand radiographs distributed between subject ages of 80 to 180 months. Ten thousand non-joint sample images were also extracted and used as negative samples.

4.3.2 Accuracy of joint detection

Joint detection accuracy in random real-world radiographic images was evaluated under the supervision of an expert clinician. Detection results were evaluated for
every joint in a single hand. Thus, all 15 joints successfully detected in an image would be calculated into 100% accuracy. Fig. 4.5 compares the performance of CNN-based FingerNet and AdaBoost-based FingerNet. The overall average accuracy of CNN (98.02%) is 6% point higher than that of AdaBoost (92.06%). The reliability of HOG features suffered with skewed or tilted images, whereas CNN is appropriate for the task because its robust features guard against such effects.

According to Fig. 4.5, both CNN and AdaBoost show the best average accuracy in the age group under 109 months, at 0.9884 and 0.9362 respectively. For 120 to 142 months and the over 170 months groups, the accuracy of the two models gradually decreased down to 0.9609 and 0.8759 each. However, the CNN model exhibited superior robustness against bone maturity.

4.3.3 Training CNN and AdaBoost

The vastness of the sample dataset is an important factor in training a CNN model, according to reports of improvement in the performance of artificial neural networks [142]. Among the numerous approaches, training data expansion with elastic distortion [119] was reportedly successful in increasing CNN handwritten-digit classification performance. All samples were expanded 100 times using elastic distortion, producing a total of 1.3 million data sets.

Table 4.1 shows the 10-fold cross validation results of the CNN model, tested with increasing training sample sizes of 13,000, 130,000 and 1.3 million. Classification performance increased with expansion of the training sample size. When 1.3 million samples were used, the classifier yielded 0.999 accuracy, with only approximately 1,300 images misclassified. The specificity of 0.996 indicates that the probability of producing false positives was relatively higher than that of producing false negatives.
Table 4.1: Effects of elastic distortion

<table>
<thead>
<tr>
<th># of data</th>
<th>ACC</th>
<th>PPV</th>
<th>NPV</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>13,000</td>
<td>0.977</td>
<td>0.986</td>
<td>0.945</td>
<td>0.984</td>
<td>0.953</td>
<td>0.985</td>
</tr>
<tr>
<td>130,000</td>
<td>0.995</td>
<td>0.998</td>
<td>0.987</td>
<td>0.996</td>
<td>0.992</td>
<td>0.997</td>
</tr>
<tr>
<td>1.3 million</td>
<td>0.999</td>
<td>0.999</td>
<td>0.998</td>
<td>0.999</td>
<td>0.996</td>
<td>0.999</td>
</tr>
</tbody>
</table>

ACC, Accuracy; PPV, Positive predictive value; NPV, Negative predictive value.

To train AdaBoost, which requires a feature extraction step, unlike CNN, HOG features \cite{18} from joint images were used. The number of weak classifiers of AdaBoost used was 144, equaling the length of the HOG vector. In the joint detection phase (JD), optimized sliding window sizes for each classifier was used, and a fixed scaling factor of 1.2 and iteration times of three were set.

### 4.4 Summary

In this chapter, we proposed an advanced method for user-interactive finger joint detection in hand radiographs called FingerNet. Using 130 test data sets (over 1,950 joints) acquired from varying subject ages, average detection accuracy is 98.02%. FingerNet achieves this superb performance by combining two complementary strategies: CNN and peak detection. As a result, FingerNet presents robust, accurate results for joint detection from hand radiographs, and it is expected to increase the efficiency of diagnoses.
Chapter 5

Weakly supervised approach for lesion detection

5.1 Introduction

There are critical limitations in expanding and applying the deep learning model, which has shown good performance thus far, in new fields. Tasks that are particularly better using deep leaning compared to existing algorithms include applications in the field of computer vision such as object detection or object segmentation, which generally require strong supervision for model learning\[36, 103, 80, 106, 31\]. Strong supervision refers to learning annotation data such as bounding boxes or segmentation masks that are manually labeled by a person in order to achieve a given purpose. In the case of medical images, the number of accumulated data increases exponentially, but there are only few annotation data\[140\]. It is unrealistic to expect time-consuming pixel-level annotations from a specialist such as a medical doctor, who spends most of his or her time treating patients during the day. Therefore, it is difficult to apply the existing detection or segmentation methodologies based on strong supervision to
medical image analysis. Even if it is applied, it uses domain adaptation after learning strong annotated dataset in other domains. Nevertheless, a few strong annotation data for fine-tuning are still needed.

Various efforts have been carried out to overcome these limitations. A weakly supervised approach is one of them, which performs detection or segmentation only with image-level annotation, and is relatively easy to obtain\cite{98,148,40,122,8,69,7,49}. However, these approaches focus only on general images that can easily be obtained from social media, such as YouTube, Facebook, and Flickr. However, this adaptation to medical imaging is not appropriate.

To overcome this limitation, we devised a Self-taught Lesion Detection Network (STLDN) that detects the lesion with only image-level annotation as was used in the existing weakly supervised approach. The proposed method trains Region Proposal Network (RPN) with surrogate ground truth, which can be made from following methods: Class Activation Map (CAM) extracted through the classification network or the fusion result of the CAM with Edgebox\cite{150}, and then derives the detection results based on the difference score. We introduced two types of surrogate ground truths to apply an appropriate learning strategy based on the characteristics of the
To evaluate the performance of the model, we used the hip joint X-ray data from patients with CP obtained from participating hospitals. As shown in Fig. 5.1 as severity increases, the hip joints in patients with CP are more likely to develop valgus deformity or hip displacement[123, 63]. It is necessary to accurately detect the lesions on the hip joint because operation methods depend on the condition of the patient, and will affect patient’s quality of life[64]. Currently, orthopedic clinicians measure parameters, such as femoral neck shaft angle (NSA), migration percentage (MP) and head shaft angle (HSA), to accurately measure severity in patients; however, this is very time consuming[63]. Therefore, we developed a detection model that specifies the location of the lesion as the first step in order to automate the severity diagnosis of patients. STLDN achieved 0.627 in terms of average precision, which is a 55.2% improvement over the state-of-the-art mask-out strategy of the weakly supervised detection method[7]. Our contributions can be summarized as follows:

- The STLDN detects the lesion on the hip X-ray image of a patient with CP based on a weakly supervised approach, which uses only image-level annotations.

- The proposed model is time-and memory-efficient because the total weights across the convolutional layers are shared in the classification and the region proposal model. Because the model computes all operations at the feature map, the inference process, which consists of a region proposal, background occlusion, and a difference scoring, can be performed within 161 ms on average for single image of 224×224 pixels on a Titan X GPU.

- We propose a self-taught RPN-based detection method, which learns the surrogate ground truth that contains the discriminative region information that is
the criterion for the discrimination between a patient and a normal person. It is more reliable than the existing method that uses the edge- or superpixel-based proposal method.

- We are able to cover cases where there are multiple lesions that are unstable in existing mask-out-based work.

5.2 Methods

STLDN is a weakly supervised detection model that identifies the lesions resulting from CP by bounding box. Fig. 5.2 shows the architectural overview of the STLDN. The training and inference procedures of STLDN are slightly different. Stage 1 (Fig. 5.2a) and Stage 2 (Fig. 5.2b) architectures are trained sequentially and the inference architecture (Fig. 5.2c) is comprised of the combination of both stages. At Stage 1, a binary classification network is trained to distinguish whether an input image shows CP. In Stage 2, the self-taught RPN is trained with a surrogate ground truth as the output of Stage 1. As represented in Fig. 5.2c, a background occlusion layer and a difference scoring layer are appended to the network, which enables the detection of a lesion from the input image. Details of the model are covered in the following section.

5.2.1 Surrogate ground truth

This section describes how pseudo-label is created in the absence of strong annotation to train the RPN. We created two types of pseudo-labels as a result of an independent scenario and regarded them as surrogate ground truths. Thereafter, we compared the detection performance of the STLDN models trained with each type of surrogate ground truth and verified the more effective type.
Figure 5.2: Architecture overview of the proposed approach.
Figure 5.3: Examples of extracted surrogate ground truths.
**Type 1**  
Type 1 is the result of localization based on the CAM generated from Eq. [3.1]. The CAM-based localization method generates a bounding box that specifies the location of the lesion, by setting pixel values less than 20% of the maximum value on the heatmap at zero, and then drawing a rectangular area surrounding the remaining pixels. Because the remaining pixels are not adjacent but can exist separately, in this case, we create clusters by grouping neighbouring pixels, and then bounding boxes are generated surrounding each cluster. We do not set the bounding box number, but rather adjust it in the above way. The middle columns in Fig. 5.3 are examples of the CAM extracted from the CP dataset, in which the target lesion exists in the highlighted area. The bounding box surrounding this region is used as the surrogate ground truth.

**Type 2**  
For the second type of surrogate ground truth, we combined CAM with EdgeBox[150]. EdgeBox is an edge-based proposal method widely used in vision tasks. It calculates the EdgeBox score \( eb_{\text{score}} \) by considering the distribution of the edges across the bounding box boundaries, and the edges in the box. The EdgeBox sorts the proposal candidates in the descending order of score, and then generates regions according to the predefined numbers of proposals. In the EdgeBox method, it is assumed that the presence or absence of an object is related to the distribution of the edges. Because semantics are not considered, the redesign of the scoring method for use as surrogate ground truth is required. Therefore, we generated the surrogate ground truth to estimate the region where the actual object is likely to exist, by adding the semantic information extracted from the CAM to the edge score. We have thus designed \( cam_{\text{score}} \) (Eq. 5.1) and \( ec_{\text{score}} \) (Eq. 5.2). Those are calculated for each proposal.
\[
\text{cam}_{\text{score}}(b) = \frac{\sum_{(x,y) \in b} p_{\text{cam}}(x, y)}{\# \text{ of pixels in } b} \times \frac{1}{255}
\]

where \( b \) is the element of bounding boxes \( B(b \in B\{b_1, b_2, ..., b_n\}) \), \( x \) and \( y \) are the pixel coordinates in \( b \), and \( p_{\text{cam}} \) is the activation value of pixel in the CAM. \( \text{cam}_{\text{score}} \) is the average CAM value of the pixels in the proposal selected by the EdgeBox method. Generally, a high CAM value can be considered as a region in which an important feature exists in classifying an object. Because the activated CAM value is implemented to have an intensity from 0 to 255, it is divided by 255 so that the maximum value is 1 in the score.

\[
e_{\text{score}} = \alpha \times e_{\text{b score}} + (1 - \alpha) \times \text{cam}_{\text{score}}
\]

By using \( e_{\text{score}} \), our proposed algorithm can generate region proposals considering both semantic and objectness informations. \( \alpha \) is a hyper parameter, which selects the weight of the EdgeBox and CAM when the proposal is taken. \( \alpha \) is chosen to be 0.55 empirically. The images in the last column of Fig. 5.3 are the proposals extracted through \( e_{\text{score}} \), which is used as the surrogate ground truth. To sample plenty of lesion candidates, the number of proposals used as surrogate ground truth is ten for each image.

### 5.2.2 ROI extraction and masking

As shown in Fig. 5.2(c), the ROI extraction and masking parts have Self-taught RPN and a background occlusion layer. The self-taught RPN trained with surrogate ground truth performs region proposal on feature maps, and at the background occlusion
layer, areas outside the selected proposals are occluded. At this point, occlusion is also performed on the feature map. The implementation details are described below.

**Self-taught RPN** The RPN model used in this study is named "self-taught RPN" because it learns the surrogate ground truth generated from Stage 1. The feature extractor in Fig. 5.2(b) share weights with the feature extractor in Stage 1. To apply the RPN that was used in the Faster R-CNN[108], we added a region proposal module to the convolution layer which generates 14 × 14 feature maps. Self-taught RPN takes a sliding window with respect to the feature map, which is the forward result of the feature extractor, and extracts \(k\) anchors with various scales and sizes for each position in order to learn whether the anchor is an object. At this time, the surrogate ground truth becomes the estimation criterion. The detailed learning process of the model using surrogate ground truth is discussed in Section 5.2.5. The loss function of the self-taught RPN consists of softmax loss and smooth L1 regression loss[31, 108]. The Self-taught RPN computes the softmax value for each anchor at the inference and transmits the top 300 proposals, which are expected to contain the target lesion, to the next background occlusion layer.
**Background Occlusion** In the absence of the bounding box annotation, determining whether an object exists in the ROI involves examining the change of the objectness score when the background area is covered and not covered. The background occlusion layer is the preprocessing step for this. Except for the area on the feature map where a region patch created by RPN is located, the background occlusion layer deactivates all feature map values to zero as shown in Fig. 5.4. Background occlusion is applied to tensor $R^i$, where the values in selected region by RPN are set to one and others are set to zero, in every existing channels $c \in \{1, ..., \text{the number of channel}\}$ by the number of proposals ($i \in \{1, ..., \text{the number of proposals}\}$), and we formulate this as Eq. 5.3 The occlusion technique on the feature map was inspired by J. Dai et al.’s occlusion of the background on the feature map for improving segmentation performance.

\[
F_{\text{occ}}^{i \times c} = F^{c} \odot R^{i} \tag{5.3}
\]

where $\odot$ represents element-wise product. $F$ and $F_{\text{occ}}$ are input feature maps and occluded feature maps, respectively.

5.2.3 Objectness scoring

In the objectness scoring step, the model uses two data streams in parallel. Each stream is configured to derive the softmax value through the convolution layer (Res5_x in Fig. 5.2(c)), the global average pooling(GAP) layer, and a fully connected (FC) layer, in that order. We utilize the trained weights of Stage 1 at the Res5_x and FC layers. In one the stream, the remaining inference process is performed on feature maps that have been background occluded from the previous step to pass. The other
stream executes forward operation on features extracted from Res4_x.

5.2.4 Detection network

The detection network verifies whether the region proposal generated from the self-taught RPN contains the object. We assumed that the region with the smallest score drop may contain the lesion when we masked the background region and performed inference.

**Difference score layer** The difference score is derived from the difference between the original feature map and the background occluded feature map. According to M.D. Zeiler *et al.* [145], the lower difference score imply a high probability that the object exists in the ROI. Therefore, this layer selects the top N candidates from calculating the difference score for every proposals. We acquired the detection results by removing the overlapped region by non-maximum suppression (NMS) [31].

5.2.5 Model training

STLDN has two separate training stages(Fig. 5.2(a) and Fig. 5.2(b)). Table 5.1 shows the model involved in each phase. All experiments are implemented on the CAFFE framework [48].

<table>
<thead>
<tr>
<th>Stage</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification net</td>
<td>O</td>
<td>-</td>
<td>O</td>
</tr>
<tr>
<td>Self-taught RPN</td>
<td>-</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Detection net</td>
<td>-</td>
<td>-</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 5.1: Models involved in each phase
**Stage 1: training classification network**

In this stage, we train a binary classifier to identify patients with CP and those without CP. Ideally, a well-converged model can capture latent features observed between normal people and patients, which affects the quality of the surrogate ground truth. In addition, as the weights of the trained model are also used in the convolution layer and the residual layer of Stage 2 (Fig. 5.2(b)) and the inference architecture (Fig. 5.2(c)), a model that achieves the best test accuracy is required. Therefore, we trained and tested the ResNet-50[37] and VGGnet-16[120] using pre-trained weights. Both models have shown outstanding classification performance on the ILSVRC[111]. After comparing the performance of the CP dataset of the two models, we chose the better one.

**Stage 2: training the Self-taught RPN**

Similar to the initialization of the existing method[108], self-taught RPN uses the weights of the convolutional layer and the residual layer of the model trained in the previous stage equivalently existing, and initializes newly-added convolution layers from the Gaussian Weight Filler[48] with zero mean and 0.01 standard deviation. At this stage, to maintain the classification performance, the weights of shared convolution layers are fixed. Training losses consist of classification loss to determine whether an object is included in the corresponding ROI, and L1 regression loss to estimate bounding box coordinates. To train self-taught RPN, the definition of positive and negative patches is needed. The surrogate ground truth is used here. The intersection over union (IOU) between the proposal patches and the surrogate ground truth is calculated, and a patch above a certain threshold is set as a positive patch and a patch less than 0.3 is set as a negative patch. In case of there are multiple surrogate ground truths in a single image, such as type2, IOU values between each patch and
Table 5.2: Validation accuracy of classification network.

<table>
<thead>
<tr>
<th>Model</th>
<th>The number of layers</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGGnet</td>
<td>16 layers</td>
<td>0.733</td>
</tr>
<tr>
<td>ResNet</td>
<td>50 layers</td>
<td>0.989</td>
</tr>
</tbody>
</table>

surrogate ground truths are computed repeatedly as the number of surrogate ground truths in an image. If one of IOU values larger than positive threshold, then we regard it as positive patch. The IOU threshold of the positive patch used in this study is 0.5 and 0.7. Because the surrogate ground truth is more inaccurate when compared with real human annotation, if we set the IOU threshold to a lower value, to determines the positive patch, neighboring regions specified by CAM can be scanned. As a result, detection performance is expected to increase, although false positives can be generated. By creating a positive and a negative patch through this process, training RPN without strong annotation becomes possible.

5.2.6 Inference

Algorithm\textsuperscript{1} represents the detailed process of inference. In contrast to previous stages, where the classifier and the RPN are separately trained, a unified model which contains both architectures, is applied to detect lesion in the inference stage (as illustrated in Fig. 5.2(c)). Detection is completed with only one inference. Detection is performed based on a difference score generated by background occlusion, and top N candidates are derived from the primary results. Patches that are larger than 25% of the original image are removed, as they act as obstacles to the calculation of the difference score, and the final detection result is derived by applying NMS\textsuperscript{1} based on the difference score of the remaining candidates.

\textsuperscript{1}The NMS used in this study is a Python-implemented version by fast R-CNN authors\textsuperscript{31}.\textsuperscript{31}
**Algorithm 1: Inference**

**Input:** Image $I$

**Output:** Bounding box $B$

1. $f_{res4x} = \text{Forward}(I)$ until res4x layer
2. **ROI extraction and masking stage:**
   1. Proposal region $R = \text{Self-taught RPN}(f_{res4x})$
   2. Occluded featuremap $f_{occ} = \text{B.G. occlusion}(R, f_{res4x})$
3. **Objectness scoring stage:**
   1. Objectness score $s_{occ} = \text{Forward}(f_{occ})$ until softmax layer
   2. Objectness score $s_{ori} = \text{Forward}(f_{res4x})$ until softmax layer
4. $\text{diff score} = \text{Calculate difference score}(s_{ori}, s_{occ})$
5. Detection results $B = \text{NMS(\text{diff score}, R)}$

### 5.3 Experimental results

For performance evaluation, we measured the average precision (AP) and accuracy as our performance metrics. We defined the accuracy as the rate of successful lesion detection in the images. The dataset used in our experiments is CP dataset obtained from the cooperative hospital, which contains the hip joint X-ray image of CP patients and normal people. All patients are anonymized. This study was approved by the institutional review board of the cooperative hospital, who waived the informed consent. The data was divided into a 9:1 ratio that represents the number of training to the number of validation data. The training data consist of 180 CP cases, 558 NCP(no CP) cases, and the validation data consists of 22 CP subjects, and 66 NCP subjects. All images are resized and center cropped to $224 \times 224$ pixels and converted to three channels by repeating it for matching input size of VGGnet\[120\] and ResNet\[37\]. To evaluate the accuracy of lesion detection for 22 CP patients, using images as validation data, an orthopedic surgeon with 18 years of experience has manually drawn the bounding boxes of the lesion on the image. We compared our proposed method with the CAM model as a baseline experiment. Furthermore, we compared our model performance with the mask-out strategy used by state-of-the-art methods in weakly-supervised detections. In the following subsections, we first discuss the results of CP
Table 5.3: The quantitative results of the detection models (IOU ≥ 0.4)

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>PR-AUC (AP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STLDN(type1.w.0.5)</td>
<td>ours</td>
<td>0.570</td>
</tr>
<tr>
<td>STLDN(type1.w.0.7)</td>
<td></td>
<td>0.371</td>
</tr>
<tr>
<td>STLDN(type2.w.0.5)</td>
<td></td>
<td>0.629</td>
</tr>
<tr>
<td>STLDN(type2.w.0.7)</td>
<td></td>
<td><strong>0.686</strong></td>
</tr>
<tr>
<td>STLDN(Mask-out.w.type2.0.5)</td>
<td></td>
<td>0.571</td>
</tr>
<tr>
<td>STLDN(Mask-out.w.type2.0.7)</td>
<td></td>
<td>0.514</td>
</tr>
<tr>
<td>CAM Model[148]</td>
<td></td>
<td>0.314</td>
</tr>
</tbody>
</table>

Table 5.4: The quantitative results of the detection models (IOBB ≥ 0.5)

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>PR-AUC (AP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STLDN(type1.w.0.5)</td>
<td></td>
<td>0.686</td>
</tr>
<tr>
<td>STLDN(type1.w.0.7)</td>
<td>ours</td>
<td><strong>0.971</strong></td>
</tr>
<tr>
<td>STLDN(type2.w.0.5)</td>
<td></td>
<td>0.743</td>
</tr>
<tr>
<td>STLDN(type2.w.0.7)</td>
<td></td>
<td>0.829</td>
</tr>
<tr>
<td>STLDN(Mask-out.w.type2.0.5)</td>
<td></td>
<td>0.743</td>
</tr>
<tr>
<td>STLDN(Mask-out.w.type2.0.7)</td>
<td></td>
<td>0.771</td>
</tr>
<tr>
<td>CAM Model[148]</td>
<td></td>
<td>0.229</td>
</tr>
</tbody>
</table>

classification and then compare the performance of lesion detection for CP dataset.

### 5.3.1 Cerebral palsy classification performance

We compared two classification models, VGGnet[120] and ResNet[37]. Each model was trained with CP dataset, where the classification accuracies achieved were 0.989 for ResNet and 0.733 for VGGnet, as described in Table 5.2. CP datasets consists of grayscale images, which have less information than generic datasets composed of RGB images. In such a situation, the ResNet model, which can analyze the input images in more depth, shows better performance. Therefore, we chose ResNet as the framework of the detection network.
5.3.2 Lesion detection performance

We measured the lesion detection performance of STLDN for various settings. In medical images, it is often difficult to determine the extent to which the detection of the lesion has succeeded, because the boundaries of the lesions that are to be detected are often ambiguous. In addition, as this paper does not evaluate the performance of the model that learned the strong annotation, the intersection over union (IOU) threshold 0.5, which is the criterion for the detection for the generic dataset, may be too tight. For this reason, in previous studies on medical images, the IOU threshold was adjusted according to the situation, or the intersection over the detected bounding box ratio (IOBB) as the replacement of IOU threshold is applied to evaluate performance\[140, 43\]. Similarly, we show that the test results of the lesion detection for 0.4 of IOU and 0.5 of IOBB in Tables 5.3 and 5.4, respectively. The qualitative results can be found in Fig. 5.5. For the sake of fairness, in the Section 5.3.3, the performance evaluated by varying IOU and IOBB thresholds are described. In this experiment, we compared the results of STLDN with four different settings, and applied the mask-out strategy for two settings with type 2 as they have better performance. Furthermore, the detection performance of the CAM model used to generate the surrogate ground truth is also described. Type 1 and Type 2 are the types of surrogate ground truth, and 0.5 and 0.7 are the thresholds used as a decision criterion for the positive patch when training RPN. From the results of the two experiments, we could confirm that the detection performance of the proposed STLDN model of Type 2 with 0.7 threshold was the best (0.400 and 0.627 for IOU and IOBB, respectively). The use of the mask-out strategy (the best result in mask strategy, 0.241 AP) was 0.15 AP higher than the best result of STLDN for Type 1 (0.091 AP) as shown in Table. 5.3. However, in Table. 5.4, the best result in mask strategy (0.404 AP) showed 0.16 lower AP than STLDN of Type 1 with 0.7 threshold (0.566 AP). The
performance of the CAM model is also better than that of Type 1 in IOU table, but is worse than that of Type 1 in IOBB table.

5.3.3 Additional results

Additional quantitative results

As mentioned in section 6.2 of the main paper, it is necessary to show the variation in performance according to the adjustment of intersection over union (IOU) and intersection over the detected bounding box (IOBB) criterion threshold for the validity of the quantitative results. Tables 5.5 and 5.6 show the results of lesion detection in terms of accuracy and average precision (AP) with respect to adjustment of the thresholds of IOU and IOBB. IOU and IOBB are the criteria for the decision of whether a selected patch is true positive or not. We applied IOU criterion from 0.1 to 0.6, increasing it in steps of 0.1, and IOBB criterion from 0.1 to 0.9, increasing it in steps of 0.25. We tested four of our models along with three existing models. The
models compared in the tables include four different models of Self-taught Lesion Detection Network (STLDN), two mask-out strategy models, and a class activation map (CAM) model. The two mask-out strategy models, which are used for performance comparison, are based on the region proposal network (RPN) trained by using surrogate ground truth Type 2 with 0.5 and Type 2 with 0.7. We did not attempt a mask-out based on Type 1 as the detection results of Type 2 are better than that of Type 1. In Table 5.5, the model: Type 2 with 0.7 shows the best performance up to an IOU threshold of 0.5, in terms of AP. When the IOU threshold is adjusted to 0.6, the best accuracy of our method is equal to the best accuracy of the mask-out strategies; although our method demonstrates lower AP value by 0.011, this is very small and relatively insignificant that our method is competitive considering other IOU criteria. In Table 5.6, experimental results on IOBB criterion confirm that our method outperforms the existing methods. When the threshold of IOBB criterion is less than 0.75, the model: Type 2 with 0.7 shows the best performance. When the threshold of IOBB criterion is 0.75 and 0.9 for each, the model: Type 1 with 0.7, has the highest accuracy and AP. The qualitative results that support the quantitative results are presented in the next section.

Additional qualitative results

We present additional qualitative results for the methods mentioned in the above section. Figs. 5.6 and 5.7 illustrate the detection results for the validation and the training set, respectively. In Fig. 5.6, it is possible to find more detection results that are not exhibited in the main paper. We used IOBB criterion with a threshold of 0.5 to decide whether a selected patch is true positive.

In Fig. 5.7, it may seem contradictory to provide the detection results of the training set. However, since the proposed method is not trained with human annotation,
Table 5.5: Performance comparison of cerebral palsy lesion detection models for IOU criterion range from 0.1 to 0.6 with an interval of 0.1. For cases where IOU criterion is larger than 0.3, the best performance in each criterion appears in bold type.

<table>
<thead>
<tr>
<th>Model</th>
<th>Type1.w.0.5</th>
<th>Type1.w.0.7</th>
<th>Type2.w.0.5</th>
<th>Type2.w.0.7</th>
<th>Mask-out(Type2.w.0.5)</th>
<th>Mask-out(Type2.w.0.7)</th>
<th>CAM Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>0.886</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.486</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.807</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.800</td>
<td>1.000</td>
<td>0.435</td>
</tr>
<tr>
<td>IOU ≥ 0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.771</td>
<td>0.971</td>
<td>1.000</td>
<td>1.000</td>
<td>0.971</td>
<td>1.000</td>
<td>0.429</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.605</td>
<td>0.566</td>
<td>1.000</td>
<td>1.000</td>
<td>0.643</td>
<td>1.000</td>
<td>0.406</td>
</tr>
<tr>
<td>IOU ≥ 0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.400</td>
<td>0.857</td>
<td>0.771</td>
<td>0.943</td>
<td>0.771</td>
<td>0.857</td>
<td>0.400</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.230</td>
<td>0.347</td>
<td>0.509</td>
<td>0.762</td>
<td>0.494</td>
<td>0.392</td>
<td>0.299</td>
</tr>
<tr>
<td>IOU ≥ 0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.057</td>
<td>0.371</td>
<td>0.629</td>
<td>0.686</td>
<td>0.571</td>
<td>0.514</td>
<td>0.314</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.004</td>
<td>0.091</td>
<td>0.354</td>
<td>0.400</td>
<td>0.241</td>
<td>0.159</td>
<td>0.233</td>
</tr>
<tr>
<td>IOU ≥ 0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0</td>
<td>0.114</td>
<td>0.371</td>
<td>0.314</td>
<td>0.314</td>
<td>0.257</td>
<td>0.143</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0</td>
<td>0.065</td>
<td>0.107</td>
<td>0.161</td>
<td>0.117</td>
<td>0.062</td>
<td>0.040</td>
</tr>
<tr>
<td>IOU ≥ 0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0</td>
<td>0.003</td>
<td>0.200</td>
<td>0.200</td>
<td>0.114</td>
<td>0.200</td>
<td>0.200</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0</td>
<td>0.030</td>
<td>0.042</td>
<td>0.015</td>
<td>0.053</td>
<td>0.047</td>
<td>0.003</td>
</tr>
<tr>
<td>IOU ≥ 0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5.6: Performance comparison of cerebral palsy lesion detection models for IOBB criterion range from 0.1 to 0.9 with an interval of 0.25. For cases where IOBB criterion is larger than 0.5, the best performance in each criterion appears in bold type.

<table>
<thead>
<tr>
<th>IOBB ≥ 0.1</th>
<th>Ours</th>
<th>Existing methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.886 1.000 1.000 1.000 1.000</td>
<td>0.514</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.808 1.000 1.000 1.000 1.000</td>
<td>0.534</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IOBB ≥ 0.25</th>
<th>Ours</th>
<th>Existing methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.886 1.000 1.000 1.000 1.000</td>
<td>0.457</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.807 1.000 1.000 1.000 0.756</td>
<td>0.418</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IOBB ≥ 0.5</th>
<th>Ours</th>
<th>Existing methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.686 0.971 0.743 0.829 0.743</td>
<td>0.229</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.488 0.566 0.490 0.627 0.393</td>
<td>0.115</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IOBB ≥ 0.75</th>
<th>Ours</th>
<th>Existing methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.371 0.743 0.371 0.143 0.343</td>
<td>0.057</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.149 0.293 0.097 0.023 0.12</td>
<td>0.012</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IOBB ≥ 0.9</th>
<th>Ours</th>
<th>Existing methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.143 0.514 0.057 0.029 0.086</td>
<td>0.029</td>
</tr>
<tr>
<td>PR-AUC (AP)</td>
<td>0.022 0.177 0.004 0.001 0.018</td>
<td>0.006</td>
</tr>
</tbody>
</table>
but with the surrogate ground truth, Fig. 5.7 enables us to demonstrate the validity of our methods. It is difficult to accurately determine whether an answer is correct or not without the help of a specialist because there is no ground truth of the training set. Nevertheless, most of the detection results in Fig. 5.7 show that the STLDN correctly detects the hip joint lesion of a patient with cerebral palsy. In addition, we extract and visualize the result of CAM as shown in the last column of Fig. 5.7. The proposed method generates the surrogate ground truth using this extracted CAM, and then trains the RPN.
Figure 5.6: Additional qualitative results of STLDN and existing methods on the cerebral palsy validation set. We used IOBB criterion with a threshold of 0.5 to decide whether a selected patch is true positive. Blue boxes represent the ground truth. Green and red colors indicate true positive and false positive, respectively.
Figure 5.7: Additional qualitative results of STLDN and existing methods on the cerebral palsy training set. Each column shows different methods. The right most column shows the CAM extracted by the classification model. It is used as a surrogate ground truth for training a self-taught RPN. Case 7 shows that CAM failed to predict the lesion of the hip joint by highlighting the outside of the body. Due to this effect, a model: Type 1 with 0.5 cannot find anything on the input image (the first column in case 7). However, the model that learned the Type 2 surrogate ground truth generated by fusion with EdgeBox, has successfully identified the hip area of the patient robustly.
5.4 Discussion

5.4.1 Effect of classification accuracy

The proposed method assumes that the performance of the classification model is highly accurate as in the CP dataset (Table 5.2). First, to train the self-taught RPN, surrogate ground truth should be generated with help from the classification model. Moreover, as this classification model is used in Stage 2 as a feature extractor and is also applied in the inference phase to calculate the difference score, classification performance can be assumed to dominate the performance of the overall detection network. If the size of the dataset is small or the characteristics between the classes are not clearly shown, the model may not be converged successfully. In this case, the quality of the CAM used as the surrogate ground truth and the reliability of the difference score cannot be guaranteed. Therefore, as future work to resolve these problems, we will study the end-to-end training strategy, which learns to minimize the difference score when proposal is generated well or maximize the difference score when proposal is poorly generated, and does not rely on the classification network.

5.4.2 Type comparison of surrogate ground truths

We investigate the pattern and detection performance of the region proposal that arises from the difference between the surrogate ground truth Type 1 and surrogate ground truth Type 2. As in Fig. 5.5, the area of the detection result of Type 2 tends to be larger than that of Type 1, because the $e_{score}$ used in training causes the self-taught RPN to explore a larger area than the specific area by the CAM, therefore, the RPN can also be considered to generate the proposal to include it. As shown in Tables 5.3 and 5.4, the overall performance is higher Type 2. In particular, the AP of Type 2 measured by IOU threshold was found to be higher than that of Type 1 (about
0.3 or more). The detection performance measured by IOBB also showed that Type 2 achieved higher value than Type 1 in terms of AP. However, when we compare the accuracy, 0.7 IOU threshold of Type 1 showed the best performance. This is because the proposal area generated by RPN trained with Type 1 is relatively small and is not canceled from NMS. Therefore, it can find even a small lesion existing in the image because it scans the area in detail, although it generates a false positive and drops the AP value.

Another interesting phenomenon is that the detection result from the surrogate ground truth is higher than that of the CAM model used to construct the surrogate ground truth. The causes of this phenomenon are divided as follows. First, the activation region of the CAM captures only the approximate location of the target lesion, thus it does not specify the exact location of the target lesion. To overcome this problem, we proposed ebscore that combines CAM and EbgeBox complementarily, and shows higher detection performance than surrogate ground-truth using CAM alone. The second reason that the detection performance of STLDN trained by surrogate ground-truth is higher than the detection performance of CAM is as follows. Self-taught RPN is a method that proposes areas where lesions are likely to exist, and it is the difference score layer that examines the objectness for that area. The self-taught RPN can propose the faulty area by learning the surrogate ground-truth formed in the wrong place, but the target lesion will be included in the other proposal region due to the effect of normal surrogate ground-truth. Therefore, by examining the difference score for each proposed region, the false positive is filtered and the detection performance is finally improved. This indicates how effective the Self-taught RPN and the difference score.
Figure 5.8: A performance comparison of lesion detection with different model.
5.4.3 Effectiveness of background masking

In this section, we examine how background occlusion affects the detection results by comparing it with ROI masking (Fig. 5.5). When there are two lesions in the image, the difference score of ROI masking does not drop correctly, while background occlusion’s score finds two lesions in an image accurately. This phenomenon is also discovered in the quantitative result. In Fig. 5.8(a) and (b), the precision value of the mask-out strategy is unstable when the recall value is less than 0.2. This unstability is caused by the inaccuracy of the difference score of that method. Consequently, the background occlusion method achieved 0.223 AP higher performance than the best result of the mask-out strategy in terms of AP measured by IOBB.

5.5 Summary

In this chapter, we proposed a Self-taught Lesion Detection Network (STLDN) and it shown to be effective when it is difficult to obtain strong annotation data such as medical images. STLDN has two training stages and it requires only image-level annotations for lesion detection. In the first stage, the classification network is trained. In the following stage, the results generated from the previous stage is designed as surrogate ground truth to train the self-taught RPN. After all training is completed, detection is performed by combining both classification and RPN models. As a combined model share all convolution layers, STLDN is memory efficient. In addition, the model demonstrated rapid inference time, lower than 161 ms on average due to the execution of background occlusion on the feature map, unlike existing methods. The proposed model shows 22.3 % point higher detection performance in terms of AP compared to state-of-the-art models. We expect that more precise surrogate ground truths can be generated with increasing training data, and hope that the capability
of our model will be able to extend to detect all of lesions simultaneously. STLDN opens the possibility in tackling data annotation scarcity in medical images.
Chapter 6

Acceleration with a storage device

The need for efficient processing of biomedical big data has been partly met by parallel computing that spans from shared-memory machines (e.g., multicore CPUs and GPUs) to distributed systems (e.g., MPI/Hadoop/Spark-based cloud computing). For instance, the Broad Institute and Intel Corporation have been jointly working on parallelizing the Genome Analysis Toolkit (GATK, [86]). Its sequential implementation takes more than 360 hours to genotype a single personal human genome, but this collaboration recently reported that it is possible to gain a more than 10-fold speedup by employing multicore processors. Nevertheless, the time demand of many bioinformatics programs still remains unsatisfactory for large-scale practical uses, due to various reasons that hinder acceleration by parallelization, such as limited parallelism in the algorithm, frequent data transfers among computing units, and high cost (time and resources) of parallelization. Additional methods for acceleration (other than parallel computing) have been sought, including storage-centric approaches that are emerging with the renewed interest in near-data processing [5]. Traditionally, there has been a substantial difference between the pace of improvements in CPUs and storage technologies, also known as the CPU-IO performance gap [53]. With the ad-
vent of NAND flash-based solid-state drives (SSDs), this gap is becoming narrower than ever, along with the gradual transition to fast host interfaces (such as PCI Express). SSDs show substantially higher performance than hard disk drives (HDDs) especially when there are frequent random input-output (IO) requests [116], not to mention their mechanical advantages originating from the lack of moving internal components. In data science and engineering, various workloads with abundant random IOs have been successfully accelerated often by a simple ‘drop-in’ replacement of HDDs by SSDs. Furthermore, traditional data analytics algorithms are being re-designed to fully exploit the new, fast secondary storage [65]. Despite the simplicity (e.g., drop-in replacement without any other modifications) and continuous cost reduction fostering widespread use of SSDs, there has been little review of SSD-based profiling and performance exploration in the bioinformatics community. In this review, we compare the performance of 23 well-known bioinformatics programs (see Table 6.1) using multiple types of SSDs and HDDs. The programs we analyze cover traditional and emerging bioinformatics areas of high importance, such as sequence alignment, genome assembly, read mapping, gene expression analysis, motif finding, variant calling, and metagenomics. We classify these bioinformatics tools into two groups, depending on the effectiveness of SSDs on speedup, and investigate the factors that cause the difference from a storage system perspective.
Table 6.1: List of the twenty three bioinformatics programs profiled and analyzed in this study. $G_+$, programs with 2x or more speedup; $G_0$, programs with negligible improvements; MSA, multiple sequence alignment; NJ, neighbor joining; HMM, hidden Markov model; EM, expectation maximization; † speedup by Intel 520 SSD over Seagate Barracuda HDD [see Tables 6.4 and 6.5 for specifications and Table 6.6 for input data]

<table>
<thead>
<tr>
<th>Name</th>
<th>Task</th>
<th>Main algorithm</th>
<th>Source</th>
<th>Speedup†</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$G_+$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GATK BaseRecal</td>
<td>Base quality recalibration</td>
<td>generates recalibration table based on covariates</td>
<td>86</td>
<td>78.4</td>
</tr>
<tr>
<td>Samtools</td>
<td>Utility tool</td>
<td>sorting, merging, indexing large sequence alignment</td>
<td>71</td>
<td>77.2</td>
</tr>
<tr>
<td>ABySS</td>
<td>NGS assembler</td>
<td>distributed de Bruijn graph, hash table searching</td>
<td>121</td>
<td>51.7</td>
</tr>
<tr>
<td>Cluster3</td>
<td>Microarray analysis</td>
<td>calculating pairwise sequence distance, clustering</td>
<td>19</td>
<td>50.0</td>
</tr>
<tr>
<td>Blat</td>
<td>Sequence alignment</td>
<td>index searching on non-overlapping k-mers</td>
<td>54</td>
<td>23.6</td>
</tr>
<tr>
<td>Reptile</td>
<td>NGS denoising</td>
<td>MSA with Hamming distance, k-spectrum extraction</td>
<td>143</td>
<td>13.7</td>
</tr>
<tr>
<td>GATK Aligner</td>
<td>Sequence realignment</td>
<td>Smith-Waterman local realignment</td>
<td>86</td>
<td>12.6</td>
</tr>
<tr>
<td>Maq</td>
<td>NGS assembler</td>
<td>ungapped sequence alignment, maximizing posterior probability</td>
<td>72</td>
<td>10.1</td>
</tr>
<tr>
<td>TopHat</td>
<td>RNA-seq analysis</td>
<td>segmented sequence alignment using Bowtie</td>
<td>134</td>
<td>3.2</td>
</tr>
<tr>
<td>MC-UPGMA</td>
<td>Microarray analysis</td>
<td>memory-constrained multi-round hierarchical clustering</td>
<td>81</td>
<td>2.7</td>
</tr>
<tr>
<td><strong>$G_0$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BWA</td>
<td>Sequence alignment</td>
<td>Burrows-Wheeler transform, trie traversal</td>
<td>20</td>
<td>1.09</td>
</tr>
<tr>
<td>Blast</td>
<td>Sequence alignment</td>
<td>seed-based local sequence alignment</td>
<td>2</td>
<td>1.08</td>
</tr>
<tr>
<td>ClustalW</td>
<td>Sequence alignment</td>
<td>multiple sequence alignment using NJ guide tree</td>
<td>130</td>
<td>1.06</td>
</tr>
<tr>
<td>GATK Unified</td>
<td>Genome variant calling</td>
<td>Bayesian likelihood modeling</td>
<td>86</td>
<td>1.05</td>
</tr>
<tr>
<td>GATK PrintReads</td>
<td>Utility tool</td>
<td>sorting, and merging sequence alignments</td>
<td>86</td>
<td>1.03</td>
</tr>
<tr>
<td>Scripture</td>
<td>RNA-seq analysis</td>
<td>sequence alignment using TopHat, graph traversal</td>
<td>34</td>
<td>1.02</td>
</tr>
<tr>
<td>IGVtools</td>
<td>Utility tool</td>
<td>sequence alignment indexing, sorting</td>
<td>109</td>
<td>1.02</td>
</tr>
<tr>
<td>Meme</td>
<td>Motif finding</td>
<td>expectation-maximization, greedy search</td>
<td>4</td>
<td>1.00</td>
</tr>
<tr>
<td>Bowtie 2</td>
<td>Sequence alignment</td>
<td>Burrows-Wheeler-based sequence alignment</td>
<td>58</td>
<td>1.00</td>
</tr>
<tr>
<td>Mosdi</td>
<td>Motif finding</td>
<td>HMM-based statistical modeling, suffix tree traversal</td>
<td>84</td>
<td>1.00</td>
</tr>
<tr>
<td>AmpliconNoise</td>
<td>NGS denoising</td>
<td>Needleman-Wunsch, hierarchical clustering, EM</td>
<td>105</td>
<td>1.00</td>
</tr>
<tr>
<td>Weeder</td>
<td>Motif finding</td>
<td>suffix tree-based exhaustive searching</td>
<td>101</td>
<td>1.00</td>
</tr>
<tr>
<td>ErmineJ</td>
<td>Microarray analysis</td>
<td>permutation, rank-based statistics analysis</td>
<td>22</td>
<td>0.97</td>
</tr>
</tbody>
</table>
Based on the insight obtained from the research, we further discuss issues in implementing and selecting bioinformatics algorithms and pipelines with the SSDs under consideration. For instance, we show that acceleration by parallelization can be accompanied by SSDs to yield extra runtime improvements. Examples include ABYSS [121] (a parallel short-read assembler) and the GATK (which uses the MapReduce framework [20]). In our experiments, ABYSS and a variant-calling pipeline using the GATK achieved 51.7 and 35.7 times speedup, respectively, when using SSDs. Another discussion on SSD-based acceleration comes from the short-read aligners for next-generation sequencing (NGS) [88]. We compare Maq [72], Burrows-Wheeler Aligner (BWA, [70]), and Bowtie 2 [58] in terms of runtime and quality metrics before and after using SSDs and analyze the result from storage-system perspectives. Based on this analysis, we further discuss how to assess alternative bioinformatics programs in terms of the viability of SSD-based acceleration. To the best of the authors’ knowledge, this review presents the first in-depth profiling analysis of major bioinformatics programs targeted at revealing opportunities and limitations of using SSDs for acceleration of bioinformatics tools. We hope that this review can provide useful directions and tips that should accompany future bioinformatics algorithm design procedures that properly consider new generations of powerful storage devices.

6.1 Introduction

6.1.1 SSD-leveraged resurrection of hash-based aligners

As a motivating example, we tested how using SSDs could accelerate well-known bioinformatics programs simply by the drop-in replacement of HDDs by SSDs in the same computer without any other modifications in hardware or software. To this end, we used the short-read alignment tools for NGS [88]. Note that the first wave
of such tools, mostly hash-based methods (e.g., Maq), has been gradually replaced by Burrows-Wheeler Transform (BWT) based methods (e.g., Bowtie 2 and BWA), mainly because of their rapid searching capabilities backed by smaller memory footprints, albeit a sacrifice in accuracy [26]. Fig. 6.1 shows the running time and quality of Maq, BWA, and Bowtie 2. The figure caption provides details of the devices and the data set used. As expected, when HDDs are used, the runtime of Maq is significantly higher than that of Bowtie 2 or BWA. Maq is a hash-based method, while Bowtie 2 and BWA are more memory-efficient BWT-based techniques. Consequently, these second-generation methods usually run faster than the first-generation aligners, especially when the data size is large and swaps frequently occur. When SSDs are used, Maq is still the slowest, but the runtime gap becomes dramatically narrower, leveraged by the enhanced IO performance and reduced swap cost of SSDs.

Given this boost in runtime and the increase in quality measured using various metrics as shown in Fig. 6.1(b), it would be possible to use Maq instead of Bowtie 2 or BWA when high values for quality metrics are desired. A simple drop-in replacement of HDDs by SSDs has made the earlier generation of tools compete with the later generation of tools, to some extent.

### 6.1.2 Measuring speedup of bioinformatics programs

To further investigate what kind of bioinformatics tools can be accelerated by using SSDs, we prepared a total of 23 bioinformatics programs listed in Table 6.1 and measured the speedup by the drop-in replacement of HDDs by SSDs. Tables 6.4 and 6.5 in Section 6.4.1 provide more details of the experiments. The result is shown in Fig. 6.2. Using SSDs yielded substantial speedup for certain programs (e.g., GATK BaseRecal), but was not always effective. Regardless of the specific SSD used for measurement, we were able to divide the 23 programs into two groups, namely
Figure 6.1: Performance comparison of three short-read aligners: Maq [72], BWA [70], and Bowtie 2 [58]. (a) Runtime. (b) Quality measured in sensitivity, accuracy, precision, and F-measure. [SSD: Samsung 840 Pro (128GB), HDD: Seagate Barracuda (1TB, 7200rpm), data: Staphylococcus aureus whole genome sequence [112].
G_+ (the programs with two times or more speedup) and G_0 (the programs with negligible or no improvements). The programs in each of these two groups are listed in Table 6.1. To find the root cause of the differences between these two groups, we will further profile and analyze these 23 programs from a storage system perspective in Section 6.2.

Note that the result shown in Fig. 6.2(a) is from using a 120GB Intel 520 SSD in place of a 1TB Seagate Barracuda HDD (3.5 inch). The results from the other five SSDs are shown in Fig. 6.2(b). Using different SSDs and HDDs did not change the group membership of each program but only its speedup ranking within each group. Thus, in what follows, we present the results obtained from using an Intel 520 and a Seagate Barracuda, unless otherwise stated. The results from using the other combinations of SSDs and HDDs are available online at http://best.snu.ac.kr/pub/biossd.

6.1.3 Accelerating bioinformatics pipelines by SSDs

Based on the initial profiling results described in Section 6.1.2, we further tested if there was any performance gain by using SSDs for running a bioinformatics pipeline that consisted of multiple component programs. As shown in Fig. 6.3, we measured the runtime of three bioinformatics pipelines before and after a drop-in replacement of HDDs by SSDs. The pipelines analyzed were for variant calling by the GATK [86], whole-genome sequence assembly and annotation [9], and transcriptome reconstruction [34].

Fig. 6.3(a) illustrates the breakdown of the runtime of the GATK pipeline for variant calling. The pipeline consisted of the component tools for sequence alignment and formatting using BWA [70] and Samtools [71], sequence realignment (GATK Aligner), sequence base-quality recalibration (GATK BaseRecal), result merge (GATK PrintReads), and variant calling (GATK Unified). By a simple drop-in replacement,
Figure 6.2: Speedup of 23 bioinformatics programs by the drop-in replacement of HDDs by SSDs. [G+, programs with two times or more speedup; G0, programs with negligible improvements] (a) SSD: Intel 520 (120GB), HDD: Seagate Barracuda (1TB, 7200rpm). (b) SSD: five different models listed in Table 6.4, HDD: the same as in (a). The order of the programs placed below the x-axis remains the same as in (a). Additional results from comparing a complete set of SSD-HDD pairs is available at http://best.snu.ac.kr/pub/biossd.
Figure 6.3: SSD-based acceleration of bioinformatics pipelines. (a) Variant calling by the GATK [86]. [data: NA12878 human whole genome sequence [11]] (b) Sequence assembly and annotation [9]. [data: Staphylococcus aureus whole genome sequence [112]] (c) Transcriptome reconstruction [34]. [data: Mouse (mm9) reads (see Table 6.6 for a link)]
we could achieve more than a 35 times decrease in the runtime of the whole pipeline. The majority of the speedup was due to the reduced runtime of formatting (Samtools, 77.2 times speedup), sequence realignment (GATK Aligner, 12.6 times speedup), and base-quality recalibration (GATK BaseRecal, 78.4 times speedup). The second pipeline depicted in Fig. 6.3(b) carries out sequence assembly and annotation. The first three steps accounted for most of the improvements and consisted of GATK Baserecal, Reptile [143], and ABySS [121], which were all accelerated significantly by SSDs, according to Table 6.1. Replacing Blast with Blat provided an additional runtime reduction, producing 75.7 times total speedup over HDDs. Of note is that ABySS, a parallel short-read assembler, was boosted more than 50 times by SSDs. This is an example in which a combination of computing parallelization and SSD-based storage can yield a dramatic performance gain. Fig. 6.3(c) shows the third pipeline for transcriptome reconstruction [34] in RNA-seq experiments [94]. The amount of speedup was smaller than the previous two pipelines. Although the most time-consuming step (Reptile) of the pipeline was accelerated significantly by SSDs, the total runtime of the pipeline was relatively shorter, and the effect of runtime reduction in Reptile was eclipsed by the Scripture [34] step. We expect that using a larger data set will reveal the effect of SSD-based runtime reduction. (Related results are presented in Section 6.2.4)

6.2 Methods

This section elaborates how we profiled and analyzed the 23 bioinformatics programs in the study. We first measured important storage features for each program and then clustered the programs with respect to the measured feature values. The measurement and clustering allowed us to discover IO patterns that could not only differentiate
Table 6.2: List of storage features used (see Section 6.4.2 for details)

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
<th>When high (low)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>UTIL</td>
<td>Interface bandwidth utilization large (small) data transfers</td>
</tr>
<tr>
<td>R</td>
<td>RIOPS</td>
<td>Read IO per second (in)frequent reads</td>
</tr>
<tr>
<td>W</td>
<td>WIOPS</td>
<td>Write IO per second (in)frequent writes</td>
</tr>
<tr>
<td>P</td>
<td>PFAULT</td>
<td># page faults per second (in)frequent page swaps</td>
</tr>
<tr>
<td>CAR</td>
<td>Consecutive Access Ratio</td>
<td>sequential (random) access</td>
</tr>
<tr>
<td>R</td>
<td>SIZE</td>
<td>Read size per request sequential (random) reads</td>
</tr>
<tr>
<td>W</td>
<td>SIZE</td>
<td>Write size per request sequential (random) writes</td>
</tr>
<tr>
<td>WB</td>
<td>LEN</td>
<td>Write buffer length (many (few) writes in queue)</td>
</tr>
</tbody>
</table>

G_+ and G_0 but also provide useful insight into when SSDs could be effective for acceleration and when not.

6.2.1 Measuring storage features

For each of the 23 bioinformatics programs, we measured eight features widely used in storage research. Table 6.2 lists more details of these features and their acronyms used in the paper. Using these features, we will consider the randomness and the amount of IO involved in these 23 programs. The amount of IO is measured by BUTIL, RIOPS, WIOPS, and PFAULT, whereas the IO randomness is measured by Consecutive Access Ratio (CAR), RSIZE, WSIZE, and WBLEN. More details can be found in Section 6.4.2.

The measurement results are shown in Figure 4. Overall, we can make the following observations:

O1 The features related to the number of IO operations issued by the host (RIOPS and WIOPS) have higher values for G_+.

O2 The features related to the amount or frequency of transfers between the host memory and the storage (BUTIL and PFAULT) are higher for G_+.
O3 Each of the features related to IO randomness (R\textit{SIZE}, W\textit{SIZE}, and CAR) shows a different pattern: R\textit{SIZE} is higher for G_0 \textit{(i.e., negligible speedup for programs with many sequential reads)}, W\textit{SIZE} is higher for G_+ \textit{(i.e., notable speedup for programs with many random writes)}, and CAR is moderately higher for G_+.

O4 The feature affected by both the amount of data transfers and IO randomness (WB\textit{LEN}) is consistently higher for G_+.

O1 and O2 can be explained by the fact that SSDs normally support higher IO Operations Per Second (IOPS) while incurring fewer overheads for swaps. Thus, the programs with more IO operations and page faults can be more effectively accelerated by SSDs. O3 and O4 are related to the fact that SSDs are superior for handling random IOs, but part of these observations is not completely intuitive at first. For instance, not only SSDs but also HDDs normally have DRAM buffers that can hide latency incurred by random writes, implying that programs with many random writes will not see significant speedup by using SSDs. This implication is seemingly against O3. In addition, according to O3, CAR is higher for G_+, which seems to suggest that the programs in G_+ show less randomness. Given that SSDs are effective for handling random IOs, O3 is seemingly inconsistent with the fact that the programs in G_+ are accelerated more by using SSDs. Section 6.2.3 includes further explanations of O3 and O4 that can answer these riddles.

6.2.2 Pattern discovery by clustering

Observations O1–O4 only reveal overall trends. For a specific program, the prediction of the effectiveness of SSDs simply using individual storage features alone may not be accurate. For example, some programs in G_0 have high B\textit{UTIL}, R\textit{IOPS}, and
Figure 6.4: Storage feature measurements. (a) Bandwidth utilization of host-storage interface (BUTIL). (b) Read IOPS (RIOPS) and write IOPS (WIOPS). (c) The number of page faults per second (PFAULT). (d) Consecutive Access Ratio (CAR). (e) Read size per request (RSIZE) and write size per request (WSIZE). (f) Storage buffer queue length (WBLEN). [see Table 6.2 and Section 6.4.2 for more details of these features]
WIOPS but do not show significant speedup. To see the combinations of features leading to effective speedup and to find patterns that can help grouping bioinformatics programs in terms of IO behavior, we tried clustering the 23 programs based on the eight storage features.

Fig. 6.4(a) shows the dendrogram obtained by agglomerative hierarchical clustering with the average linkage. We use the Euclidean distance metric to measure the distance between two vectors, each of which consists of the eight measurement values normalized and ranged in [0, 1] (see Section 6.4.1). Cutting the dendrogram near the root bifurcation point reveals the two groups $G_+$ and $G_0$. Cutting it at the smaller distance as shown in the plot produces five clusters or patterns. Group $G_+$ consists of three patterns (denoted by $P_1$, $P_2$, and $P_3$), while group $G_0$ contains two (denoted by $P_4$ and $P_5$).

Fig. 6.5(b) shows the radar chart representation of the average feature values for each pattern. Fig. 6.5(c) shows the numerical values depicted in the radar charts. Evidently, the most notable difference between the three patterns in $G_+$ and the two patterns in $G_0$ is the average PFAULT value. However, the effect of PFAULT may not be observed clearly when the main memory is large, and we need to compare different patterns using other storage features.

To facilitate the comparison of the five patterns discovered, we present their representative IO traces in Fig. 6.6. We show two traces (read and write) for each pattern. In each trace, the x-axis and the y-axis represent the IO request time and the logical block address (LBA), respectively. Each vertical line corresponds to an IO request, and its length matches the read/write size.

Using the information presented in Fig. 6.5 and 6.6, we can identify notable characteristics of each pattern. For instance, $P_1$ has a high amount of IOs, frequent random reads and sequential writes. $P_1$ shows the lowest RSIZE (0.01) among all the
Figure 6.5: Clustering bioinformatics programs based on the eight storage features listed in Table 6.2. (a) Dendrogram and pattern definitions. The numbers represent the pairwise distance. (b) Radar chart representations of the average feature values for each pattern. Legend is also shown. (c) The numerical values of the average features depicted on the axes of the radar charts in (b). The names and the speedup amounts are also presented.
Figure 6.6: Representative IO traces for each of the five patterns shown in Fig. 6.5. A vertical bar corresponds to an IO request, and its length represents the read or write size. The quantity of vars in each plot is proportional to the IO amount, while the distribution of accessed LBAs represents the IO randomness. [LBA, logical block address; G+, programs with 2x or more speedup; G0, programs with negligible or no improvements]
five patterns, meaning that the read size per request is very small. Additionally, a CAR of 0.72 suggests that 72% of the IO requests make consecutive access to the LBA. Taken together, we expect small data reads from often consecutive locations. In contrast, WSIZE (0.81) of P1 is the highest among all the patterns. Again with 72% CAR, this implies frequent sequential writes of relatively large data. R1OPS and W1OPS are the highest in P1, implying a high amount of IOs. This is also backed by the high values of BUTIL, WBLEN, and PFAULT. In particular, high W1OPS is responsible for high WBLEN.

In a similar manner, we can also interpret the other patterns.

6.2.3 Impact of IO randomness on speedup

We present how the IO randomness affects the amount of speedup by SSDs. We also show that the randomness alone may not always be a good indicator of speedup and should be accompanied by other storage features for more accurate prediction. In Fig. 6.7 for each of the two plots in this figure, the x-axis represents CAR, while the y-axis corresponds to RSIZE or WSIZE. For each of these features, recall from Section 6.2.1 that approaching 1.0 means that the access becomes more sequential, whereas approaching 0.0 indicates more randomness in IO. Each program is represented by a circle, whose size is proportional to the amount of speedup using SSDs.

For the read case depicted in Fig. 6.7(a), we see that the IO randomness, measured by either RSIZE or CAR, is a reasonable first-order indicator for speedup. That is, either small RSIZE or CAR gives speedup by SSDs. For instance, the two patterns associated with steep speedup (P1 and P2) manifest themselves through different types of randomness: P1 has tiny RSIZE but its CAR is not small, whereas P2 has small CAR but its RSIZE is high. P4 shows a typical sequential read behavior (both RSIZE and CAR are high), and the speedup is limited. Comparing P4 with P1 or
Figure 6.7: Impact of randomness on speedup. Each circle represents one of the 23 bioinformatics programs listed in Table 6.1 and its radius is proportional to the amount of speedup achieved by a drop-in replacement. See Figure 5 for pattern definitions. To fully explain the different levels of speedup of different patterns, we need to consider not only randomness but also the other storage features. See the text for details. Discussions on ABySS, Maq, Bowtie 2, and BWA can be found in Section 4. (a) Read. (b) Write. [\( R_{\text{SIZE}} \), read size per request; \( W_{\text{SIZE}} \), write size per request; \( \text{CAR} \), consecutive access ratio]
**P2** confirms that the read randomness is an important factor.

When both RSIZE and CAR have intermediate values, however, it is less obvious to predict the amount of speedup only by randomness. For instance, if we compare P3 and P5 in Fig. 6.7(a) only by RSIZE and CAR, then P5 should give higher speedup, which is not the case in reality. This is because the amount of IO is small for P5, as indicated in Fig. 6.7(b) and (c), and there is little chance for SSDs to accelerate the IO.

In the write case depicted in Fig. 6.7(b), we also observe that other storage features in addition to randomness need to be considered, although randomness remains an important factor for speedup. **P2** has small CAR and shows large speedup, which confirms that SSDs are effective for handling random writes. For the other patterns, we need to consider the role of write buffers inside storage devices. For writes, even HDDs can hide write latency to some extent using the write buffers. This can explain why **P4** does not show speedup even though it has similar levels of randomness measured in CAR compared to P1 or P3, both of which show noticeable speedup. **P1** and **P3** have higher WSIZE than **P4**, which leads them to have higher WBLEN.

### 6.2.4 Impact of input size on SSD effectiveness

We hypothesized that even tools that generate a small amount of IO may benefit from using SSDs as the input size grows. Feeding large data may cause the main memory to be full generating frequent swaps. In this case, using SSDs may help reduce the runtime.

To verify this theory, we tried feeding increasingly larger data to Amplicon-Noise [105], a program in **P5**. Recall that the programs in **P5** are not very effectively accelerated by using SSDs, mainly because of their CPU-intensive behavior producing only a small amounts of IOs. The baseline data contains 2000 sequences sampled
Figure 6.8: CPU-intensive programs in P5 that produce a small amount of IO for moderate-size data may also benefit significantly by using SSDs for handling very large-scale data. [program: AmpliconNoise [105], baseline data: 2000 reads from 454 Titanium [105]] from the 454 Titanium data [105], and we generated larger data sets by replicating the baseline data. For each data set, we measured the runtime, as shown in Fig. 6.8.

The break-even point appears after replicating the baseline data five times. After that, using SSDs yields a huge speedup. This experiment confirms our theory and suggests that adopting SSDs may or may not be a smart decision, depending on the size of input data, even for the same program. For instance, AmpliconNoise often handles a number of pyrosequenced reads and is likely to benefit from using SSDs, although AmpliconNoise belongs to P5.

6.2.5 Effect of main memory size on SSD-based acceleration

The size of main memory affects the runtime of a workload, and ideally, the effect of using SSDs would be eclipsed in a system equipped with the main memory large enough for storing all the input/intermediate/output data. In reality, however, the
memory footprint of a bioinformatics workload often becomes significantly larger than the main memory size affordable in typical systems, necessitating the use of a speedy secondary storage, such as SSDs. We tested how the size of main memory affects the amount of speedup by SSDs using the GATK program, as shown in Fig. 6.9(a). For an input dataset of 20GB sampled from the NA12878 human whole genome sequence [1], we ran the GATK using three sizes of main memory (4GB, 16GB, and 32GB) and measured the runtime of each of the four subprograms in the GATK for each memory configuration.

Using SSDs was most effective for the sequence base-quality recalibration (GATK BaseRecal) step, which shows high randomness in IO and belongs to P1. For the two
memory sizes smaller than the input size (4GB and 16GB main memory), SSDs delivered a significant amount of speedup (66.32 times and 49.79 times, respectively). Even for the 32GB configuration, we observed more than 30 times speedup, which suggests that the memory footprint of GATK BaseRecal grows substantially during execution and the use of SSDs was effective. For the sequence realignment step (GATK Aligner), the use of SSDs was helpful only for the 4GB memory configuration. For the setups with 16GB and 32GB memory, the amount of speedups was negligible. Although the input file size was 20GB, using SSDs was ineffective for 16GB main memory, which reveals the computing-intensive characteristic of the sequence alignment operation in GATK Aligner and the limited effectiveness of SSDs. For the other two programs (GATK Unified and GATK Printreads), we observed only negligible effects of using SSDs.

6.2.6 Additional experiments

In addition to the eight features listed in Table 6.2, which are mostly related to storage devices, we measured CPU- and memory-related features (e.g., CPU usage and cache hit/miss ratios), as shown in Fig. 6.10. The CPU usage was higher for G₀, and the tools therein can be considered more compute-intensive than those in G₊. The miss ratios for the lower-level caches and the translation lookaside buffer (TLB) tend to be higher for G₊, confirming their memory-intensive behavior. The page fault rate was also higher for G₊, which is compatible with the experimental results presented earlier.
Figure 6.10: Additional measurements of CPU- and memory-related features for the 23 bioinformatics programs. Speedup of each program is also shown. Features were normalized to values between 0 and 1. [G+, programs with 2x or more speedup; G0, programs with negligible improvements]
6.2.7 Summary and guidelines for employing SSDs in bioinformatics pipelines

As seen in Fig. 6.5(b) and 6.5(c), the most notable difference between G+ and G0 comes from the amount of page faults. In other words, when the memory footprint of a program exceeds the capacity of main memory, using SSDs is likely to bring a significant gain over using HDDs. By contrast, the programs with small memory footprints is less likely to be accelerated by using SSDs. Optimizing a program by reducing its memory footprint may bring a similar effect as using SSDs, but such a code optimization would typically require a nontrivial amount of efforts. Adopting SSDs thus becomes a more appealing option especially when the resources for code optimization are limited. Installing more main memory would also be helpful for reducing the runtime of programs, but the cost of DRAM may easily become prohibitively expensive, let alone the limited memory bandwidth issue. Other factors that differentiate G+ and G0 include the randomness of IO requests and the amount of data transfers: the more random and larger read/write requests, the more effective the use of SSDs. As the size of input data grows, even some of the programs in G0 may benefit from using SSDs. When deploying SSDs in a cluster environment, the administrator of the cluster should consider the network constraints before replacing HDDs with SSDs, because the effect of successful local acceleration may become eclipsed by the network latency, resulting in no overall performance gain (see Section 6.3 for more details).

6.2.8 Training of deep learning-based model on SSDs

The training time of a deep learning-based model is accelerating with the development of the GPU. However, the bottleneck due to IO still exists in the learning process of the deep learning-based model. According to a previous study, it is necessary
Figure 6.11: SSD-based acceleration of the training of the deep learning-based model. The datasets used in these experiments were a cerebral palsy (CP) hip X-ray dataset and a meningioma (MNG) MR dataset. (a) Effects of main memory size on SSD-based acceleration of the model training for the CP dataset. (b) Effects of main memory size on SSD-based acceleration of the model training for the MNG dataset. (c) Effects of batch size on SSD-based acceleration of the model training for the CP dataset. (d) Effects of batch size on SSD-based acceleration of the model training for the MNG dataset.
to select the batch randomly for the training of the deep learning-based model, and when the epoch is finished data IO occurs to load the next batch. At this time, the GPU becomes idle while waiting for the next batch of data, and, as a result, the time efficiency decreases as the number of epochs increases [104].

In this section, we discuss the benefits of a simple drop-in replacement for HDDs by SSDs when training the deep learning-based model for biomedical data. The HDDs and SSDs used in the experiments were a 1000-GB Seagate Barracuda and a 512-GB Samsung 850 Pro, respectively, and one Titan Xp GPU (GDDR 12 GB) was utilized for training the deep learning-based model. The data used in the experiments were a cerebral palsy (CP) hip X-ray dataset and a meningioma (MNG) MR dataset. The CP dataset consists of 826 images, and the MNG MR dataset consists of 189,429 images. From each experiment, we can see the impact of dataset size on SSD-based acceleration. All experiments were performed with the Caffe framework. The deep learning model exploited was ResNet50 [37], and the results are shown for 10,000 iterations.

**Effect of main memory size on SSD-based acceleration:** Fig. 6.11(a) and (b) shows how the size of the main memory affects the training time of a deep learning-based model when using SSDs. The training time of the models for the CP and the MNG dataset was measured for main memory sizes of 1 GB, 4 GB, 8 GB, and 16 GB, respectively, and the batch size was equal to 32. Neither the CP nor the MNG model showed any speedup, except when the main memory size was 1 GB. The speedup that occurred when the main memory was 1 GB was also marginal because the GDDR memory of the GPU was efficiently used for batch data loading.

**Effect of batch size on SSD-based acceleration:** To see the effect of batch size on the SSD-based acceleration of the training of the deep learning-based model, we
Table 6.3: Storage features measured on the training of the deep learning-based model. The data used in the training of the deep learning-based model were a cerebral palsy (CP) dataset and a meningioma (MNG) MR dataset.

<table>
<thead>
<tr>
<th>Features</th>
<th>CP dataset</th>
<th>MNG dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUTIL</td>
<td>0.087</td>
<td>0.091</td>
</tr>
<tr>
<td>R1OPS</td>
<td>0.118</td>
<td>0.736</td>
</tr>
<tr>
<td>W1OPS</td>
<td>0.326</td>
<td>0.199</td>
</tr>
<tr>
<td>PFAULT</td>
<td>0.767</td>
<td>0.78</td>
</tr>
<tr>
<td>CAR</td>
<td>0.019</td>
<td>0.018</td>
</tr>
<tr>
<td>RSIZE</td>
<td>0.189</td>
<td>0.526</td>
</tr>
<tr>
<td>WSIZE</td>
<td>0.233</td>
<td>0.244</td>
</tr>
<tr>
<td>WBLEN</td>
<td>0.868</td>
<td>0.860</td>
</tr>
</tbody>
</table>

measure the training time while changing the batch size to 32, 64, and 128, with the main memory size fixed at 1 GB (see Fig. 6.11(c) and (d)). In a CP dataset with a small dataset size, no speedup occurred on the SSDs even if the batch size changed. However, in the case of the MNG data set, it can be seen that, as the batch size increased, the training speed was accelerated on the SSDs. This is because if the size of the dataset is large, the data loader of the model is more likely to randomly access the logical block address (LBA) of the storage device to organize the batch, thus increasing the number of random read operations. The results of Table 6.3 show that the amount of IO was large and frequent random reads occurred when training the deep learning-based model for the MNG dataset. The reason why no performance improvement occurred was largely due to the that the GDDR memory of the GPU was efficiently utilized for batch loading, as mentioned in the previous experiment. However, if the size of the training data is large and the capacity of the GDDR and main memory is exceeded, it is expected that the reduction in training time by the SSDs would be effective.
6.3 Discussion

The 23 programs we profiled represent traditional and emerging areas of importance, such as sequence alignment (including conventional dynamic programming-based, heuristic, and BWT-based algorithms), NGS denoising, assembly and mapping (including RNA-seq tools), gene expression analysis, motif finding, variant calling (including four GATK components), and metagenome analysis. These programs should cover the most frequent usages of bioinformatics data processing and related computation.

Through our experiments, we confirmed that acceleration by parallelization can be combined with the use of SSDs for even more performance increases. For example, using SSDs could accelerate ABySS more than 50 times, even though ABySS is a state-of-the-art parallelized assembler. The compute-intensive nature was mitigated by multicore processing, while the data-intensive nature seems to have been handled by SSDs. The GATK package is another example. The GATK was implemented using the MapReduce framework, which is amenable to parallel processing. In our experiments, SSDs could reduce the time demand of the two time-consuming components of the GATK (BaseRecal and Aligner) by 78.4 and 12.6 times, respectively. When we design load balancing for parallelization, it will be helpful to consider the amount and randomness of IOs so that we can take advantage of SSDs.

In case the analysis pipeline contains a component program that is not accelerated by using SSDs, replacing the program with an alternative that runs faster on SSDs can help reduce the runtime of the overall pipeline. For example, in the sequence assembly and annotation pipeline depicted in Fig. 6.3(b), replacing Blast (only 1.08x speedup) with Blat (23.6x speedup) provided additional speedup to the whole pipeline. When there are multiple options for selecting a component block in
a pipeline, it will thus be beneficial to assess the alternatives in terms of the effectiveness of using SSDs.

To this end, we can consider the three short-read aligners as an example: Maq (hash-based first-generation tool), Bowtie 2, and BWA (BWT-based second-generation tools). These three tools show similar CAR values, although Maq belongs to P3 and Bowtie 2 and BWA both belong to P4. In contrast, there is a difference in the IO size: Maq issues smaller reads but generates larger writes, which are linked to larger values of PFAULT and WBLEN. When HDDs are used, it imposes a critical limitation on the performance of Maq. To overcome this issue, significant efforts were made to invent the new generation of tools (Bowtie 2 and BWA) that have smaller memory footprints. The efforts could have been accompanied by using SSDs for even more improvements, given that the page faults and random IOs can be efficiently handled by SSDs.

There remain other intriguing topics for further research. A hybrid drive contains a spacious (but slow) HDD and a speedy (but small) SSD altogether inside a package. The access patterns are monitored, and frequently accessed “hot” data are cached automatically and dynamically in the SSD while the majority of the data are stored in the HDD. Using such a hybrid drive will be helpful for acceleration, under the conditions that the workload program creates enough IO requests (e.g., the programs in group G+) and the composition of the hot data do not change frequently over time.

Exploiting the redundant array of independent disks (RAID) technology [38] may provide additional advantages in performance and reliability. In particular, RAID level 0, which consists of striping without mirroring or parity, will be helpful for significantly improving data throughput. As long as the bandwidth of the host interface (e.g., SATA, PCIe, and NVM Express) is high enough to maintain the enhanced data throughput, using SSDs in RAID 0 will be helpful for accelerating high-throughput
bioinformatics workloads.

Recently Hadoop-based clusters [128] are popular in large-scale data analytics including bioinformatics. The Hadoop file system (HDFS) provides a distributed storage layer on which various MapReduce-based operations are performed [13]. The randomness inherently occurring in the Map phase can be effectively handled by using SSDs [93], which are far more superior to HDDs in terms of handling random IO requests. Improving the performance of a namenode (the node managing distributed file systems) in a Hadoop system by SSDs may provide another opportunity for SSD-based acceleration. In distributed systems, however, the network latency often eclipses the speedups achieved locally (e.g., shared-memory-based parallelization and SSD-based acceleration) [3], and improving the overall performance globally may require significant efforts. Thus, even if the most frequently used applications in a cluster include the programs in the G group, the administrator of the cluster should carefully examine any network constraints that may exist before replacing HDDs with SSDs.

6.4 Experiments

6.4.1 Experiment setup and measurements

The SSDs and HDDs used in our experiments are listed in Tables 6.4 and 6.5 respectively. We selected these devices because they were the most popular in the market at the time of our experiments. For conservative comparison, the SSDs are low-end models with 128GB or less capacity, whereas the HDD selection includes high-performance WD VelociRaptor.

Many of the bioinformatics tools we used take a long time to process large data especially when HDDs are used (often in the order of days or even weeks). To com-
Table 6.4: Specifications of the SSDs used in this work

<table>
<thead>
<tr>
<th>SSD</th>
<th>Capacity (GB)</th>
<th>Sequential (MB/s)</th>
<th>Random (IOPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read</td>
<td>Write</td>
<td></td>
</tr>
<tr>
<td>Samsung 830</td>
<td>128</td>
<td>520</td>
<td>80,000</td>
</tr>
<tr>
<td>Samsung 840 Pro</td>
<td>128</td>
<td>530</td>
<td>97,000</td>
</tr>
<tr>
<td>OCZ Vertex4</td>
<td>128</td>
<td>560</td>
<td>90,000</td>
</tr>
<tr>
<td>Intel 520</td>
<td>120</td>
<td>550</td>
<td>50,000</td>
</tr>
<tr>
<td>Plextor M5 Pro</td>
<td>128</td>
<td>540</td>
<td>91,000</td>
</tr>
<tr>
<td>Corsair Neutron GTX</td>
<td>120</td>
<td>555</td>
<td>85,000</td>
</tr>
</tbody>
</table>

Table 6.5: Specifications of the HDDs used in this work

<table>
<thead>
<tr>
<th>HDD</th>
<th>Capacity (GB)</th>
<th>RPM</th>
<th>Buffer size (MB)</th>
<th>Read/write (MB/s)</th>
<th>IOPS (Estimated)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>Seagate Barracuda</td>
<td>1,000</td>
<td>7,200</td>
<td>64</td>
<td>156</td>
<td>79.0</td>
</tr>
<tr>
<td>WD Caviar Blue</td>
<td>1,000</td>
<td>7,200</td>
<td>64</td>
<td>150</td>
<td>76.6</td>
</tr>
<tr>
<td>WD VelociRaptor</td>
<td>500</td>
<td>10,000</td>
<td>64</td>
<td>200</td>
<td>151.5</td>
</tr>
</tbody>
</table>

To compare the performance of HDDs and SSDs using the same data sets while keeping experiments manageable, for each program, we selected an input data set of appropriate size that can be processed in a reasonable amount of time (the criterion used: less than 72 hours). Table 6.6 lists details of the data used to profile the 23 bioinformatics programs.

To see the effects of changing secondary storage clearly in this setup, we also adjusted the specifications of the computer used accordingly. Unless otherwise specified, we used a machine equipped with a 3.3GHz Intel Core i3-3220 CPU (four threads, 4MB L3 cache), 1600MHz dual-channel DDR3 memory (4GB for the GATK tools and 1GB for the others), and Ubuntu 12.04 LTS (Precise Pangolin). For the results shown in Figure 9, we used a machine with a Samsung 840 Pro SSD, a 3.4GH Intel Core i7-3770 CPU (eight threads, 8MB L3 cache), and 1600MHz dual-channel DDR3 memory (4GB, 16GB, and 32GB). For performance profiling and measurement, we used time (with option -eUSKFW), System Activity Reporter (SAR, [83]),
Table 6.6: List of the data used to test the 23 programs listed in Table 6.1

<table>
<thead>
<tr>
<th>Program</th>
<th>Data</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATK BaseRecal</td>
<td>NA12878 human</td>
<td>link†</td>
</tr>
<tr>
<td>Samtools</td>
<td>C2</td>
<td></td>
</tr>
<tr>
<td>ABBySS</td>
<td>Staphylococcus aureus</td>
<td>[112]</td>
</tr>
<tr>
<td>Cluster3</td>
<td>Protein structure</td>
<td>[144]</td>
</tr>
<tr>
<td>Blat</td>
<td>NCBI Uniref50 protein</td>
<td>[135]</td>
</tr>
<tr>
<td>Reptile</td>
<td>Human chromosome 14</td>
<td>[112]</td>
</tr>
<tr>
<td>GATK Aligner</td>
<td>NA12878 human</td>
<td>link†</td>
</tr>
<tr>
<td>Maq</td>
<td>Human chromosome 14</td>
<td>[112]</td>
</tr>
<tr>
<td>Tophat</td>
<td>Drosophila melanogaster</td>
<td>link‡</td>
</tr>
<tr>
<td>MC-UPGMA</td>
<td>Protein structure</td>
<td>[144]</td>
</tr>
<tr>
<td>BWA</td>
<td>AT1</td>
<td>[6]</td>
</tr>
<tr>
<td>Blast</td>
<td>NCBI Uniref50 protein</td>
<td>[135]</td>
</tr>
<tr>
<td>ClustalW</td>
<td>NCBI Uniref50 protein</td>
<td>[135]</td>
</tr>
<tr>
<td>GATK Unified</td>
<td>NA12878 human</td>
<td>link†</td>
</tr>
<tr>
<td>GATK PrintReads</td>
<td>NA12878 human</td>
<td>link†</td>
</tr>
<tr>
<td>Scripture</td>
<td>Mouse (mm9) reads</td>
<td>link§</td>
</tr>
<tr>
<td>IGVtools</td>
<td>Mouse (mm9) reads</td>
<td>link§</td>
</tr>
<tr>
<td>Meme</td>
<td>Human sequence hm01</td>
<td>[132]</td>
</tr>
<tr>
<td>Bowtie 2</td>
<td>AT1</td>
<td>[6]</td>
</tr>
<tr>
<td>Mosdi</td>
<td>Human sequence hm01</td>
<td>[132]</td>
</tr>
<tr>
<td>AmpliconNoise</td>
<td>454 Titanium</td>
<td>[105]</td>
</tr>
<tr>
<td>Weeder</td>
<td>Human sequence hm01</td>
<td>[132]</td>
</tr>
<tr>
<td>ErmineJ</td>
<td>Human genome U95 set</td>
<td>link‡</td>
</tr>
</tbody>
</table>

‡ http://trace.ddbj.nig.ac.jp/DRASearch/submission?acc=SRA012173

blktrace [11], and Intel VTune Amplifier XE. To avoid interference between tools, we ran each of these profilers independently. We used time and SAR for measuring CPU usage and virtual-memory related features, blktrace for measuring block-level storage features (e.g., read/write amounts, throughput, and IOPS), and VTune for measuring CPU-internal features (e.g., cache hit/miss, TLB hit/miss, and IPC). When the range of measurements was large, we took the logarithm. We then normalized each of the measurements so that values were ranged in [0, 1]. We repeated all the time measure-
ments three times and used the average value for the analysis.

6.4.2 More details of the storage features used

As mentioned previously, we profile and analyze the 23 programs in terms of eight storage features that can characterize the amount and/or randomness of IOs. To measure the amount of IO we use three measures. **BUTIL** measures how much bandwidth of the interface between the host computer and the storage device is used. If there is a large amount of data transfers between the host and storage, **BUTIL** would be high. **RIOPS** and **WIOPS** measure how many read and write requests are made per second, respectively. A high value of these features implies frequent read/write requests. **PFAULT** represents the number of page swaps per second. High **PFAULT** suggests frequent page swaps, which can be costly for HDDs.

The randomness of IOs can be measured in different ways. In this study, we use two widely used measures: read/write size per request [68] and CAR [133]. Reads or writes that transfer a small amount of data are often considered random, whereas large read/write transfers are considered sequential. **CAR** measures how often consecutive access to the LBA space occurs. The **CAR** value of one (zero) means perfectly sequential (random) IO access patterns. **WBLEN** represents the number of write requests waiting in the write buffer of a storage device. High **WBLEN** normally can be caused by a high amount of write IOs and/or by a large number of small random writes. **WBLEN** is thus related to both the amount and the randomness of IOs.

6.5 Summary

There exist cases in which a simple drop-in replacement of HDDs by SSDs can dramatically expedite bioinformatics programs. For instance, we observed more than
50 times speedup of widely used tools, such as GATK components, Samtools, and ABySS. In the arena of short-read aligners, we observed that Maq (a hash-based first-generation tool) could compete again with Bowtie 2 and BWA (the second-generation tools) leveraged by SSDs. According to our experiments, using SSDs could accelerate the GATK-based variant calling pipeline by more than 30 times.

However, SSDs are not silver bullets and cannot boost every bioinformatics program of one’s interest. Moreover, SSDs are still expensive. Eventually the price of SSDs may become competitive to HDDs, but the price per gigabyte of SSDs is still approximately 15 times more expensive, as of 2015. Researchers handling large-scale biomedical data should thus make a careful and informed decision regarding whether to replace their HDDs (at least partially) with SSDs.

To this end, profiling the bioinformatics tools of interest from system perspectives is critical. According to our experiments, there exist many bioinformatics programs that can benefit immediately by using SSDs, especially when the program causes frequent random IOs or page swaps due to relatively large input compared to system memory. This review reports other patterns indicating the viability of SSD-based acceleration. As the size of input data grows, we expect that the territory of the SSD-acceleratable programs will expand. In any case, as the performance of SSDs is rapidly improving with continuous cost reduction and technology developments, SSDs will eventually become the storage device of choice, phasing out HDDs firstly in performance-critical domains and later in the mainstream. We thus believe that future bioinformatics algorithms should be designed to consider the advantage of using SSDs in addition to the applicability of parallel processing. We hope that the results and insight presented in this review will be a valuable asset to such a journey for inventing efficient and scalable bioinformatics tools.
Chapter 7

Conclusion

In this dissertation, we review the problems that may arise when applying a machine learning method for biomedical data analysis, and describe the approaches to handling them. This chapter summarizes our contributions to overcoming the aforementioned issues and introduces the expected effects and future work.

Chapter 3 addressed the difficulty in interpreting the prediction results of the deep learning-based model from the user’s perspective. We proposed a pyramid Grad-CAM (PG-CAM) that captures the discriminative region of the magnetic resonance image more precisely than the existing methods and facilitates the interpretation. The PG-CAM exploits a feature pyramid network (FPN) with dense connection and outputs an attention map that reflects multi-scale features. Furthermore, by the nature of the encoder-decoder architecture that generates the last feature maps of the same size with the input image, an attention map extracted from the PG-CAM can capture the fine details of the target lesion.

Chapter 4 introduced an approach that makes the performance of the prediction model robust even if the training data are insufficient. A way to compensate for the small amount of training data is to use domain knowledge as additional information.
For an example of how to solve the above problem, we presented a methodology to detect a finger joint, which is a measure of the health of a patient, by inputting a radiographic image of the human hand. To achieve the above objectiveness, we improved the robustness of the detection model by fusing the deep learning-based method and the signal processing technique for finding the specific patterns found in finger joints. Additionally, an elastic distortion-based data augmentation technique is applied to sufficiently converge the test accuracy of the deep learning-based model. FingerNet achieved a 98.02% average detection accuracy for 130 test data sets containing over 1950 joints. Further analysis was performed to verify the system robustness against factors such as epiphysis and metaphysis in different age groups.

In Chapter 5 we proposed a Self-Taught Lesion Detection Network (STLDN) that detects areas in hip X-ray images that are noticeably different from the general population by using deep learning. The proposed method is a weakly supervised approach to find lesions using image-level labels with no box annotation, as it is difficult to obtain strongly annotated data in most cases of medical image analysis. The STLDN uses a shared convolutional layer at the inference stage for a more efficient memory model and performs all operations on the feature map, such that the detection results can be derived from a single inference within 161 ms on average. The proposed method outperformed a state-of-the-art mask-out strategy, with a 55.2% performance improvement in terms of average precision (AP) on the hip X-ray dataset of cerebral palsy (CP) patients obtained from a participating hospital.

Chapter 6 provided an in-depth profiling of important biomedical data analysis workloads from a storage-system perspective and presented guidelines that should be helpful for biomedical data analysis practitioners to optimize their biomedical data analysis tools and pipelines, and for biomedical data analysis researchers to design their next-generation algorithms. The time demand of many biomedical data analysis
programs still remains high for large-scale practical uses, despite various parallelization efforts; additional acceleration schemes should be employed to further reduce the time demand of important biomedical data analysis programs. Depending on the characteristics of biomedical data analysis workloads, we observed that a simple drop-in replacement for HDDs by SSDs could result in orders of magnitude speedup in the same machine and that certain biomedical data analysis programs may be accelerated easily without undergoing time-consuming optimization.

### 7.1 Future work

The studies presented in this dissertation can be expanded in various ways. First, the methodologies described in Chapters 3, 4, and 5 can be applied to various diseases. In addition, PG-CAM can serve as the surrogate ground truth of the STLDN, which is an appropriate attempt to improve the performance of a weakly supervised approach model. With respect to Chapter 4, FingerNet is able to improve the assessment of the severity of rheumatoid arthritis or the assessment of the bone age in a fully automated fashion. In particular, the results of FingerNet demonstrate the validity of the complementary combination of the conventional method and the deep learning method. Thus, a complementary fused approach is expected to be applicable in various fields where domain knowledge is strong in addition to biomedical data analysis. The STLDN presented in Chapter 5 currently has many false positives, and the detection performance of the model will improve if additional post-processing techniques are applied. With respect to Chapter 6, the IO pattern of the biomedical data analysis workloads in which the acceleration occurs can also affect the IO scheduling design, and further studies on acceleration in storage devices for various deep learning-based lesion detection models will be prospective.
Bibliography


[6] Andrea K Bartram et al. Generation of multimillion-sequence 16s rRNA gene libraries from complex microbial communities by assembling paired-end il-


[14] Hyun-Soo Choi, Siwon Kim, Jung Eun Oh, Jee Eun Yoon, Jung Ah Park, Chang-Ho Yun, and Sungroh Yoon. Xgboost-based instantaneous drowsi-


[52] Akash P Kansagra, J Yu John-Paul, Arindam R Chatterjee, Leon Lenchik, Daniel S Chow, Adam B Prater, Jean Yeh, Ankur M Doshi, C Matthew


[67] Taehoon Lee and Sungroh Yoon. Boosted categorical restricted boltzmann


[90] Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. Deep patient: an


Conference on Computer Vision and Pattern Recognition, pages 779–788,
2016.


[108] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: To-
wards real-time object detection with region proposal networks. In Advances

Guttman, Eric S Lander, Gad Getz, and Jill P Mesirov. Integrative genomics

[110] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional
networks for biomedical image segmentation. In International Conference on
Medical Image Computing and Computer-Assisted Intervention, pages 234–

[111] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh,
Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bern-
stein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recog-
nition Challenge. International Journal of Computer Vision, 115(3):211–252,
2015.

[112] Steven L Salzberg, Adam M Phillippy, Aleksey Zimin, Daniela Puiu, Tanja
Magoc, Sergey Koren, Todd J Treangen, Michael C Schatz, Arthur L Delcher,


[127] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabi-


초록

하드웨어의 발전과 방대한 크기의 데이터셋의 공개로 인공지능 분야는 황금기에 접어들었다. 인공지능에 기반한 연구들은 오랜 시간 답보 상태에 있던 영상 인식, 객체 감지, 자연어 처리, 기계 번역 및 자율주행 자동차와 같은 다양한 분야에서 성공적인 결과를 보여주었다. 생체 의료 데이터 분석 분야에서도 방대하게 축적되는 생체 의료 빅데이터를 효과적으로 분석하기 위해 기계 학습을 적용하려는 다양한 연구가 시도되고 있다.

한편, 생체 의료 데이터에 기계학습 기법을 효과적으로 적용하기 위해서는 극복해야만 하는 몇 가지 이슈가 존재한다. 첫번째 이슈는 기계 학습 기법이 의료 현장에서 진단 보조도구로 적용되려면 기계의 예측 결과와 그에 대한 추정 근거가 해석이 가능해야 한다는 것이다. 두번째 이슈로 특정 질환에 대한 생체 데이터의 크기가 딥러닝과 같은 대량의 학습데이터를 요구하는 기계학습 모델의 학습에는 부족할 수 있다는 것이다. 더 나아가, 모델의 학습을 위한 그라운드 트루스 데이터의 부족도 또한 하나의 이슈로 여길 수 있다. 생체 의료 데이터의 경우 그라운드 트루스 데이터를 생성하기 위해서는 의사와 비슷한 전문가의 노력이 불가피하며 이를 확보하기란 매우 어려운 일이기 때문이다. 마지막으로 인간 유전체와 같이 방대한 양의 생체 데이터를 분석해야 하는 경우 분석 도구의 입출력 폭뿐만 아니라 재작용에 따라 전체적인 분석 시간이 영향을 줄 수 있다는 점이다.

본 학위 논문에서는 각각의 이슈들을 해결하기 위해 제안한 접근법들을 4개의 챕터에 걸쳐 제시한다. 첫번째로는 딥러닝 기반의 모델을 진단 보조도구로 사용하였을 때, 사용자가 모델의 판단 근거를 시각적으로 피드백 받을 수 있도록 하는 pyramid Grad-CAM을 제안하였다. 두번째로는 학습 데이터가 부족한 상황에서 딥러닝 모델을 성공적으로 학습시키고, 모델의 강아함을 항상시키기 위한 방법을
소개한다. 학습데이터 부족을 극복하기 위하여 가우시안 노이즈 기반의 왜곡을 활용한 데이터 중강 기법을 사용하였으며, 학습된 모델을 보조할 수 있는 신호 처리 기법 기반의 방법론을 상보적으로 융합하였다. 세번째 이슈를 극복하기 위해서 약한 지도 학습법에 기반한 새로운 병변 검출 기법을 소개한다. 마지막으로는 방대한 생체 의료 데이터 분석 기법을 저장장치 단에서 가속화 할 수 있는 입력력 패턴을 발견하기 위하여, 23개의 생물정보학 어플리케이션에 대한 심층적인 프로파일링과 계층적 군집화 기법을 통한 입력력 패턴 분석을 수행하였다. 본 학위 논문에서는 이와 같이 생체 의료 데이터의 효과적인 분석을 위한 다양한 기계 학습 기반의 분석 기법과 가속화 방안을 제안하였다.

주요어: machine learning, deep learning, weakly supervised learning, convolutional neural network, lesion detection, medical image analysis
학번: 2013-20848

수용대학교