NAIST-IS-DD1821003

Doctoral Dissertation

Decoding the representation of source code categories in the brain of expert programmers

Yoshiharu Ikutani

March 1, 2021

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Yoshiharu Ikutani

Thesis Committee:

Professor Kenichi Matsumoto	(Supervisor)
Professor Kazushi Ikeda	(Co-supervisor)
Associate Professor Takatomi Kubo	(Co-supervisor)
Associate Professor Takashi Ishio	(Co-supervisor)
Assistant Professor Hideaki Hata	(Co-supervisor)
Senior Researcher Shinji Nishimoto	(NICT CiNet)

Decoding the representation of source code categories in the brain of expert programmers*

Yoshiharu Ikutani

Abstract

Expertise enables humans to achieve outstanding performance on domain-specific tasks, and programming is no exception. Many studies have shown that expert programmers exhibit remarkable differences from novices in behavioral performance, knowledge structure, and selective attention. However, the underlying differences in the brain of expert and novice programmers are still unclear. This thesis addresses the issue by associating the cortical representation of source code with individual programming expertise using a data-driven decoding approach. The approach identified multiple distributed brain regions, located in the frontal, parietal, and temporal cortices, that have a tight relationship with programming expertise. In these brain regions, functional categories of source code could be decoded from brain activity and the decoding accuracies were significantly correlated with individual behavioral performances on a source-code categorization task. The results suggest that programming expertise is built upon fine-tuned cortical representations specialized for the domain of programming.

Keywords:

Programming expertise, program comprehension, brain decoding, functional magnetic resonance imaging, the neuroscience of programming

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 1, 2021.

Contents

List of Figures

List of Tables

1. Introduction

Programming expertise is one of the most notable capabilities in the current computerized world. Since human software developers keep playing a central role in software projects and directly impact their success, this relatively new type of expertise is attracting increasing attention from modern industries [1,2] and educational institutes [3,4]. Moreover, huge productivity variations were repeatedly found even between programmers with the same level of experience [5,6]. Previous studies have shown the psychological characteristics of expert programmers in their behaviors [7,8], knowledge structures [9,10], and eye movements [11–13]. Although these studies clearly illustrate the behavioral specificity of expert programmers, it remains unclear what neural bases differentiate expert programmers from novices.

Recent studies have investigated the brain activity of programmers using functional magnetic resonance imaging (fMRI). Siegmund *et al.* contrasted brain activity during program output estimations against syntax error searches and showed that the processes of program output estimations activated left-lateralized brain regions; including the middle frontal gyrus, inferior frontal gyrus, inferior parietal lobule, middle temporal gyrus [14,15]. Their results suggested that program comprehension is associated with natural language processing, division of attention, and verbal/numerical working memory. Peitek *et al.* reanalyzed the same data as [14] to investigate the correlation between the BOLD activation strength and individual programming experience, which was determined by subject's self-estimation, but did not find any significant trend [16]. An exploratory study argued that a correlation exists between activity pattern discriminability and subjects' grade point average (GPA) scores counting only courses from the Computer Science department as a proxy for programming expertise [17]. However, the GPA scores would reflect a mixture of diverse factors (IQ, memory ability, calculation skills, etc.) and the assumed relationship of the score to programming expertise was difficult to be empirically validated. Further, the main limitation of these prior studies is the use of a homogeneous subject group that only covered a small range of programming expertise. Recruitment of more diverse subjects in terms of their programming expertise may enable the elucidation of the potential differences of brain functions related to the expertise.

This thesis presents an fMRI study to identify the neural bases of programming expertise that contribute to the outstanding performances of expert programmers. To begin the study, two fundamental factors were defined: an objectively determined reference of programming expertise and a laboratory task that exhibits experts' superior performances under the general constraints of fMRI experiments. First, the study adopted the programmers' ratings in competitive programming contests (AtCoder, `https://atcoder.jp/`), which are objectively determined by the relative positions of their actual performances among thousands of programmers. Top- and middle-rated programmers as well as novice controls were recruited to cover a wide range of programming expertise in the fMRI experiment. Second, the program categorization task was designed for the study and was confirmed that behavioral performances of this task were significantly correlated with the adopted reference of programming expertise. This confirmation allows us to expect an association between the outstanding performances of expert programmers and brain activity patterns recorded by fMRI while they performed this laboratory task.

The core hypothesis of the study is that higher programming expertise and experts' outstanding performances relate to specific multi-voxel pattern representations, potentially influenced by their domain-specific knowledge and training experiences. This hypothesis is motivated by prior studies that contrasted multi-voxel activity patterns of experts against novices and demonstrated that domain-specific expertise generally associates with representational changes in the brain [18–20]. For example, Bilalić *et al.* showed that the multi-voxel patterns in expert radiologists' fusiform face area are more sensitive in differentiating X-ray images from control stimuli than novices [21]. Similarly, identifying the multi-voxel pattern representations specific to expert programmers offers a good starting point for understanding the cognitive mechanisms behind programming expertise. From the previous studies on non-expert programmers and expertise in other domains, the high-level visual and left fronto-parietal regions might be inferred as potential neural correlates of programming expertise [14,22]. However, to the best of our knowledge, there is no prior evidence that directly associates programming expertise with specific brain regions. Thus, this thesis employs a whole-brain searchlight analysis [23] to identify the regions related to program-

ming expertise.

This thesis demonstrates that the functional categories of source code can be decoded from programmers' brain activity and decoding accuracies in multiple distributed brain regions are significantly correlated with individual behavioral performances. In addition, the thesis shows that decoding accuracies of subordinate-level categories on two brain regions are significantly correlated with individual behavioral performance, even though such discriminations are not explicitly required by the tasks. These results suggest that expert programmers' outstanding performances depend on fine-tuned cortical representations of source code categories and such cortical representation refinements might be related to the acquisition of advanced-level programming expertise.

1.1 Contributions

This thesis makes the following contributions.

- It presents a novel fMRI experiment to uncover the association between a programmer's brain activity and individual programming expertise.
- It confirms that individual programming expertise is significantly correlated with behavioral performances of source code categorization task.
- It demonstrates that functional categories of source code can be decoded from brain activity and the decoding accuracies on multiple distributed brain regions are significantly correlated with individual behavioral performances.
- It demonstrates that decoding accuracies of subordinate-level categories on two brain regions are significantly correlated with individual behavioral performance.
- It makes available the source code and de-identified fMRI dataset for future study and replication.

1.2 Outline

The rest of the thesis is organized as follows: Section 2 views related studies on expert programmers and programming expertise. Section 3 gives a detailed description of materials and methods used in the fMRI experiment. Section 4 describes the results of the experiment and shows an association between experts' outstanding performances and their domain-specific cortical representations. Section 5 provides the interpretation and implication of the findings as well as the potential limitations of the study. Section 6 presents the conclusion of this thesis and promising future work.

2. Related work

2.1 Expert programmers and programming expertise

K. Anders Ericsson defines experts as "people who produce clearly above average performances on a regular basis" and expertise as "the characteristics, skills, and knowledge that distinguish experts from novices and less experienced people" [24]. In the domain of software engineering, many expertise studies have been performed with different terminologies, such as programming skills [25], programming experience [26], developer fluency [27]. For example, Baltes and Diehl conducted an online survey on 355 professional software developers and illustrated the cognitive and behavioral characteristics in expert programmers toward building the theory of software development expertise [2]. In addition, multiple experiments have investigated expert programmers and observed their performances clearly higher than novices or non-expert programmers. The pioneering study conducted by Sackman *et al.* found huge variations in individual programming productivity: Their experiment, in particular, showed the ratio of initial coding time between best and worst subjects was 20:1 and debugging time was 25:1 [28]. Later studies have also demonstrated that there is a significant productivity variation between programmers, which can not be explained by simple years of experience [5, 6]. This line of studies confirmed the general finding that there are order-of-magnitude differences among programmers and drove many researchers to investigate how expert programmers differ from novices and what cognitive characteristics make experts as experts [29].

From the 1980s to the 1990s, researchers conducted several observational studies and investigated behavioral differences between expert and novice programmers. For example, Vessey *et al.* collected episodic data from 16 programmers using a think-aloud protocol, which required subjects to say whatever comes into their mind while they were performing the given task, and suggested several hypotheses on experts' debugging processes [7]. Fix *et al.* recruited 20 novices and 20 expert programmers and asked them to answer a series of questions designed to show cognitive characteristics in their program comprehension processes [9]. The results obtained in these observational studies contributed to building classical mental models of program comprehension processes (see [10], for a review). How-

ever, the results were often inconsistent across the studies and many researchers suspected the validity and replicability of self-reports [30, 31].

In the early 2000s, several researchers have begun to employ eye-tracking tools for quantifying experts' program comprehension processes more objectively (for review, see [32], [33]). Uwano *et al.* measured the eye movement of five programmers while they reviewed six source code written in C language and identified an eye movement pattern associated with high performance in code reviews [12]. Busjahn *et al.* focused on the linearity of eye movement during program comprehension processes and demonstrated that expert programmers read source code less linearly than novices [13]. Eye-tracking methods provide an objective measure of program comprehension processes and strengthen the replicability of observed results. For example, the results obtained by Uwano *et al.* [12] and Busjahn *et al.* [13] were replicated in later studies conducted by different researchers [34,35]. Although eye-tracking methods successfully quantify where the attention of an expert programmer is focused on, it has difficulty to explain the cognitive characteristics underlying experts' superior performances because eye movement data ignores lots of cognitive computations in the brain.

From the early 2010s, many researchers have started to measure a programmer's brain activity using an electroencephalogram [36, 37], near-infrared spectroscopy [38, 39], and fMRI [14, 15, 40, 41]. The original and replication studies conducted by Siegmund *et al.* measured brain activity using fMRI while subjects performed Java program output estimations and demonstrated that the processes activated a set of left-lateralized brain regions [14, 15]. The studies have confirmed that brain activity measurements can be used as an objective indicator of program comprehension processes with valid replicability. A recent fMRI study has measured brain activity of programmers understanding Python programs, suggesting the association between program comprehension and fronto-parietal network that is functionally related to formal logical inference [40]. However, the underlying differences in the brain between expert and novice programmers are still unclear. Peitek *et al.* reanalyzed the same data as [14] to investigate the correlation between the BOLD activation strength and individual programming experience but did not find any significant trend [16]. Floyd *et al.* argued an association between the pattern discriminability in fMRI signals and subjects' GPA

scores [17] but the assumed relationship of the score to programming expertise was not empirically validated. This thesis, therefore, aims to identify the neural bases of programming expertise and provide evidence that associates individual programming expertise with specific brain regions.

2.2 Neuroscience of expertise

Expertise enables humans to achieve outstanding performance on domain-specific tasks. Many neuroscientists have been investigating how experts accommodate such expertise in the brain [22]. Guida *et al.* reviewed neuroimaging studies on experts and suggested the association between expertise acquisition and functional cortical reorganization, which is seen as the recruitment of new activation areas and a shift in cognitive process underlying expert's superior performance [42]. For example, Wan *et al.* scanned the brain activity of professional and amateur players in a board game named shogi and found two professional-specific activations in the precuneus and caudate nucleus during quick generation of the best next move [43]. Amalric and Dehaene demonstrated that reading mathematical statements activated a reproducible set of bilateral frontal, intra-parietal, and ventrolateral temporal regions only in professional mathematicians [44]. These previous studies imply that expertise acquisition might be accompanied by the recruitment of new activation areas. The evidence additionally suggests that programming expertise might be associated with the recruitment of additional activation areas observed only in the brain of expert programmers.

More recently, several studies demonstrated that expertise can alter the cortical representations of domain-specific information. Bilalić *et al.* showed that the multi-voxel patterns in expert radiologists' Fusiform Face Area were more sensitive in differentiating X-ray images from control stimuli than novices [21]. Brants *et al.* scanned the brain activity of twelve human subjects using fMRI before and after training of object categorization and differentiation [45]. Their results showed that training to categorize or individuate specific objects strengthens preexisting cortical representations in the human object-selective cortex. Another study applied multi-voxel pattern analysis on the brain activity of professional cinematographers and sound designers [18]. As a result, they found the distinct multi-voxel patterns specific to the modality of subjects' expertise, suggesting that modality-specific expertise can alter the representation of domain-specific information. This thesis follows the line of expertise studies and hypothesize that higher programming expertise and experts' outstanding performances are associated with specific multi-voxel pattern representations in the brain.

3. Materials and Methods

3.1 Subjects

For this study, three recruiting criteria were defined: *Expert*, top 20% rankers in AtCoder who had an AtCoder rate equal to or higher than 1,200; *Middle*, 21-50% rankers who had an AtCoder rate between 500 and 1,199; *Novice*, subjects who had four years or less programming experience and no experience in competitive programming. The recruiting messages were sent via mailing lists and messaging applications with diverse graduate or undergraduate student communities in Japan. Through this procedure, 95 programmers from 28 universities and three companies completed the entry questionnaire to be registered as candidate subjects. The list of candidate subjects consisted of 19 experts (all male), 43 middles (one female), and 33 novices (nine females). Nine left-handed subjects and 20 subjects with less than half a year experience in Java programming were excluded from the list. Five subjects aged under 20 years old were also excluded to avoid additional bureaucratic processes. The remaining candidate subjects were asked to participate the experiment basically on first-in-first-out strategy. Note that setting *Novice* as programmers who had an AtCoder rate under 500 was another potential recruiting criterion; but this study did not adopt the criterion because low values in the rate reflects two indistinguishable factors: low programming expertise or not enough contest participation. In addition, possession of AtCoder rate itself could imply possession of moderate programming expertise. Thus, the recruiting criteria set *Novice* as a programmer with shorter experience in programming and no experience in competitive programming.

Thirty healthy subjects (two females, aged between 20 and 24 years) with normal or corrected-to-normal vision participated in the experiment; see Table.1 for the demographic information of recruited subjects. All were right-handed (assessed by the Edinburgh Handedness Inventory [46], laterality quotient = 83.6 $\pm$ 24.0, ranged between +5.9 and +100) and understood basic Java grammars with at least half a year experience in Java programming. The averaged AtCoder rates (1,967 in *Expert* and 894 in *Middle*) were equivalent to the top 6.5% and 34.1% positions among 7,671 registered players based on the ranking on July 1 2017, respectively. Higher AtCoder rate indicates higher expertise

Table 1. **Demographic information of recruited subjects.** The population of middle and novice classes included one female subject each. Numerics from 3rd (Age) to last columns denote 'MEAN ± SD'. Abbreviations: PE, programming experience; JE, Java experience; CPE, competitive programming experience. Age, PE, JE, CPE are written in a year unit. Significant differences were observed between PE of Expert - Novice, Middle - Novice; CPE of Expert - Middle (two-sample t-test, $p < 0.05$ FDR-corrected).

Class	N	Age	AtCoder rate	PE (year)	JE (year)	CPE (year)
Expert	10	22.6 ± 1.1	1969 ± 467	6.9 ± 2.8	2.8 ± 2.4	4.1 ± 2.6
Middle	10	22.5 ± 0.8	894 ± 175	4.8 ± 1.7	1.1 ± 0.8	1.3 ± 0.8
Novice	10	21.7 ± 1.2	NA	2.8 ± 0.6	1.4 ± 1.0	NA

to win high scores in competitive programming contests and implies that high-rated programmers possess greater skills in writing/reading source code and richer domain-specific knowledge of efficient computer algorithms and data structures. Seven additional subjects were scanned but not included in the analysis because one (novice) showed neurological abnormality in MRI images, three (one expert and two middles) retired from the experiment without full completion, three (one expert and two novices) showed strongly-biased behavioral responses judged when the behavioral performance of one or more choices did not reach chance-level in the training experiments, signaling a strong response bias of sticking to a specific choice. This study was approved by the Ethics Committees of NAIST and CiNet and subjects gave written informed consent for participation. The sample size was chosen to match previous fMRI studies on human expertise with similar behavioral protocols [18, 21, 44].

3.2 Stimuli

For this study 72 code snippets written in Java were collected from an open codeset provided by AIZU ONLINE JUDGE (`http://judge.u-aizu.ac.jp/onlinejudge/`); an online judge system where many programming problems are listed and everyone can submit their own source code to answer those problems online. Four functional categories (*category*) and eleven subordinate concrete algorithms (*subcategory*) were selected based on two popular textbooks about computer algorithms [47, 48]; see Fig.1a for all *category* and *subcategory* classes. By searching in the open codeset, 1251 candidates were found as Java code snippets implementing one of the selected algorithms. To meet the screen size constraint in the MRI scanner, code snippets with a number of lines of more than 30 and a max number of characters per line of more than 120 were excluded. From all remaining snippets, a set of 72 code snippets was created with minimum deviations of these numbers of lines and characters to minimize visual variation as experimental stimuli; the mean and standard deviation of the number of lines and max characters per line were 26.4 ± 2.4 and 59.3 ± 17.1, respectively (see Table 2 and Table 3 for detailed statistics on each *category* and *subcategory* class). In the codeset, 18 snippets each belonged to one of the *category* classes and six snippets each belonged to one of the *subcategory* classes except for the linear search class with twelve snippets. The indentation styles of code snippets were normalized by replacing a tab-space with two white-spaces and user-defined functions were renamed to neutral such as "function1" because some of the functions indicated their algorithms explicitly (see Fig.2 for example snippets with normalized indentation styles and function names). All code snippets were verified to have no syntax error and run correctly without run-time error. The reasons why this study focused on Java were because the language has been one of the most famous programming languages and prior fMRI studies on programmers also used Java code snippets as experimental stimuli [14–16].

3.3 Experimental design

The fMRI experiment consisted of six separate runs (9 min 52 sec for each run). Each run contained 36 trials of the program categorization task (Fig.1b) plus one dummy trial to avoid the undesirable effects of MRI signal instability. Seventy-two code snippets were used as stimuli and each snippet was presented three times through the whole experiment (216 trials in total), but the same snippet appeared only once in a run. The experiment employed PsychoPy [49] (version 1.85.1) to display the code snippets in white text and a gray background without syntax highlighting to minimize visual variations. In each trial of the program categorization tasks, a Java code snippet was displayed for ten seconds after a fixation-cross presentation for two seconds. Subjects then responded within four seconds by pressing buttons placed under the right hand to indicate which *category* class was most plausible for the code snippet and all response data were automatically collected for the calculation of individual behavioral performance. To clarify classification criteria, a brief explanation about each *category* class was provided before the experiment started (see Table 4 for the provided description). The presentation order of the code snippets was pseudo-randomized under balancing the number of exemplars for each *category* class across runs. The corresponding buttons for each answer choice were also randomized across trials to avoid linking a specific answer choice with a specific finger movement. Subjects were allowed to take a break between runs and to quit the fMRI experiment at any time.

All subjects took two additional sessions, named "Training" and "Post-MRI", outside of the MRI scanner using a laptop computer and PsychoPy to display source code stimuli. The training session was performed within ten days before the fMRI experiment to mitigate potential confounds caused by task unfamiliarity. The session consisted of three separate runs with the same program categorization task as the fMRI experiment. A different set of 72 Java code snippets from those used in the MRI experiment, which covered the same algorithms, was used as stimuli in the training session; each snippet was presented once or twice in the entire session but the same snippet did not appear twice in a run. The post-MRI session was performed within ten days after the fMRI experiment for assessment of individual ability in *subcategory* categorizations and was consisted

of two separate runs using the same codeset as the fMRI experiment. Before the post-MRI session started, the existence of *subcategory* was explained to the subjects (see Table 5 for the provided description) and assessed whether they recognized subcategory classes during the fMRI experiment using a questionnaire. Program categorization tasks in the post-MRI session followed the same procedure as the fMRI session. In each trial, a Java code snippet was displayed for ten seconds after a fixation-cross presentation for two seconds. Then, within four seconds, the subjects were asked to classify the given code snippet from two or three choices of *subcategory* classes according to its superordinate category, e.g., "bubble sort", "insertion sort", and "selection sort" were displayed when the snippet in "sort" category was presented.

Behavioral performance was calculated as the ratio of correct-answer-trials to all-trials; unanswered trials, i.e. no button input within the response phase, were regarded as "incorrect" for this calculation. Chance-level behavioral performance was 25% in the training sessions and fMRI experiments and 37.25% in the post-MRI sessions adjusted for imbalanced numbers of answer choices. Note that the program categorization task was designed to quantify the ability to semantically categorize source code snippets. Although the ability to understand a word, line, and chunk (a set of multiple lines) in the given code snippet was required to perform the task, these abilities were out of the experiment's scope and were not directly evaluated in this study.

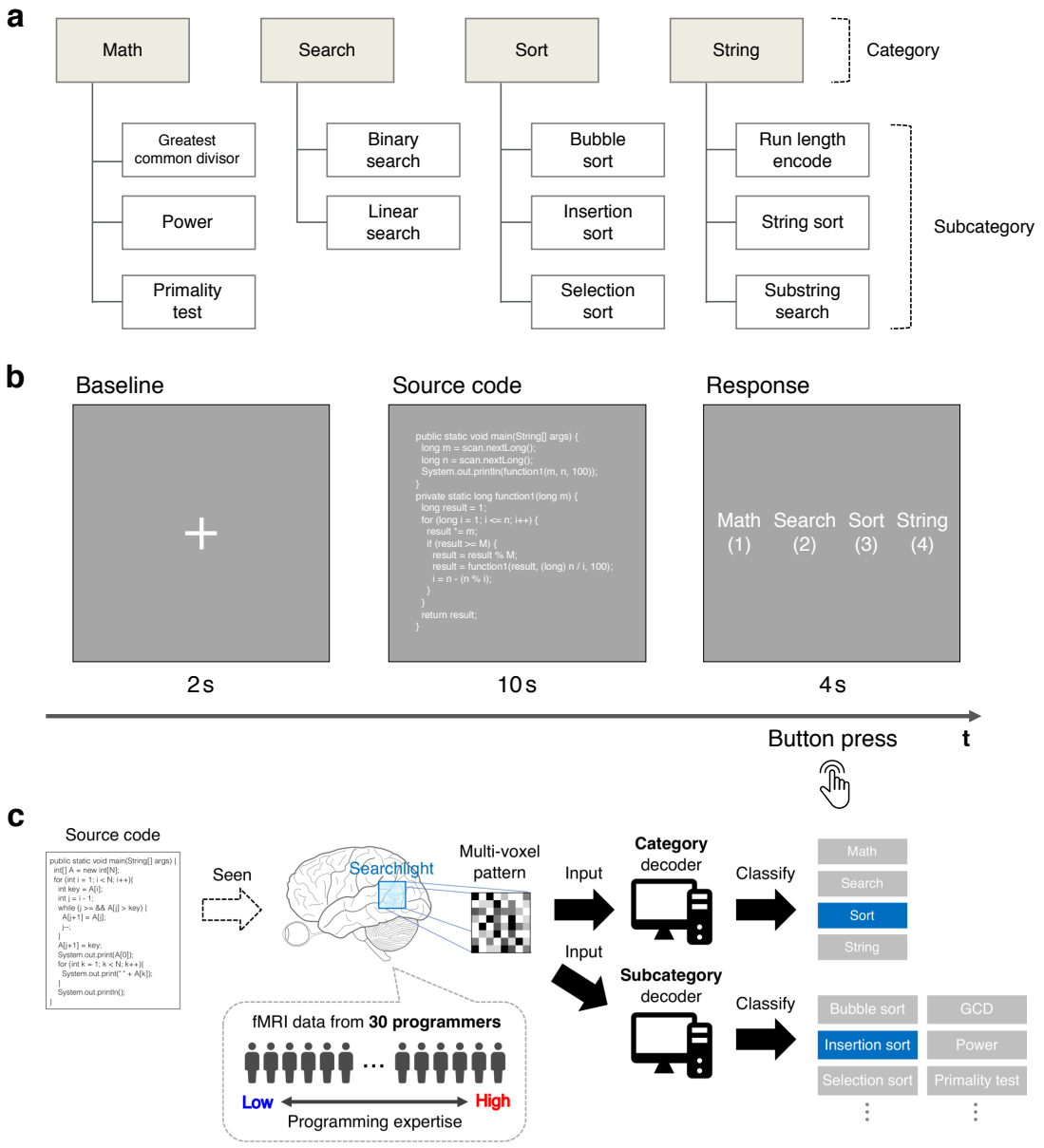

Figure 1. **Experimental design.** (a) Hierarchy of categories used in this study. Category and Subcategory represent abstract functionality and concrete algorithms, respectively, based on two popular textbooks of programming [47, 48]. (b) Program categorization task. (c) Overview of the decoding framework.

Greatest common divisor

```
public static void main(String[] arg) {
  if (a >= b) {
    function1(a, b);
  } else {
    function1(b, a);
  }
}
static void function1(int n, int m) {
  if ((n % m) == 0) {
    System.out.println(m);
  } else if (m == 1) {
    System.out.println(1);
  } else {
    int m1 = (n % m);
    function1(m, m1);
  }
}
```

Power

```
public static void main(String[] args) {
  long m = scan.nextLong();
  long n = scan.nextLong();
  System.out.println(function1(m, n, 100));
}
private static long function1(long m) {
  long result = 1;
  for (long i = 1; i <= n; i++) {
    result *= m;
    if (result >= M) {
      result = result % M;
      result = function1(result, (long) n / i, 100);
      i = n - (n % i);
    }
  }
  return result;
}
```

Primality test

```
public static void main(String[] args) {
  int n = input.nextInt();
  int res = 0;
  for (int i = 0; i < n; ++i) {
    int x = input.nextInt();
    if (function1(x))
      ++res;
  }
}
static boolean function1(int x) {
  if (x < 2)
    return false;
  for (int i = 2; i <= Math.sqrt(x); ++i) {
    if (x % i == 0)
      return false;
  }
  return true;
}
```

Binary search

```
public static void main(String[] args){
    Scanner input = new Scanner(System.in);
    int n, q;
    n = input.nextInt();
    ArrayList s = new ArrayList();
    for (int i = 0; i < n; ++i) {
      int x = input.nextInt();
      if (s.size() > 0)
        continue;
      s.add(x);
    }
    s.retainAll(t);
    System.out.println(s.size());
}
```

Linear search

```
public static void main(String args[]) {
  int n, q, T, cnt, ans = 0;
  int[] S = new int[10001];
  for (int i = 0; i < n; i++) {
    S[i] = sc.nextInt();
  }
  for (int i = 0; i < q; i++) {
    T = sc.nextInt();
    S[n] = T;
    cnt = 0;
    while (S[cnt] != T) {
      cnt++;
    }
    if (cnt < n) {
      ans++;
    }
  }
}
```

Bubble sort

```
public static void main(String args[]) {
  Scanner sc = new Scanner(System.in);
  ArrayList A = new ArrayList();
  n = sc.nextInt();
  for (int i = 0; i < n; i++) {
    A.add(sc.nextInt());
  }
  for (int i = 0; i < n - 1; i++) {
    for (int j = n - 1; j > i; j--) {
      if (A.get(j - 1) > A.get(j)) {
        temp = A.get(j - 1);
        A.set(j - 1, A.get(j));
        A.set(j, temp);
        cnt++;
      }
    }
  }
```

Insertion sort

```
public static void main(String[] args) {
  int[] A = new int[N];
  for (int i = 1; i < N; i++) {
    int key = A[i];
    int j = i - 1;
    while (j >= 0 && A[j] > key) {
      A[j + 1] = A[j];
      j--;
    }
    A[j + 1] = key;
    System.out.print(A[0]);
    for (int k = 1; k < N; k++) {
      System.out.print(" " + A[k]);
    }
    System.out.println();
  }
}
```

Selection sort

```
public static void main(String[] args) {
  for (int i = 0; i < n; i++)
    a[i] = In.nextInt();
  int count = 0;
  for (int i = 0; i < n - 1; i++) {
    int minj = i;
    for (int j = i; j < n; j++) {
      if (a[j] < a[minj]) {
        minj = j;
      }
    }
    if (minj != i) {
      int tmp = a[i];
      a[i] = a[minj];
      a[minj] = tmp;
    }
  }
}
```

Run length encode

```
public static void main(String[] args) {
  while (stdIn.hasNext()) {
    for (int i = 0; i < t.length;) {
      if (t[i] != '@') {
        i++;
      } else {
        i += 2;
        int f = t[i - 1] - '0';
        for (int j = 0; j < f; j++) {
          System.out.print(t[i]);
        }
        i++;
      }
    }
  }
}
```

String sort

```
public static void main(String[] args) {
  int x, i;
  String stock;
  if (x != 0) {
    String[] data = new String[x];
    for (i = 0; i < x; i++) {
      data[i] = scan.next();
    }
    for (i = 1; i < x; i++) {
      if (data[0].compareTo(data[i]) > 0) {
        stock = data[0];
        data[0] = data[i];
        data[i] = stock;
      }
    }
  }
}
```

Substring search

```
public static void main(String[] args){
  String w, t;
  String[] strArray;
  int n = 0;
  w = br.readLine().toLowerCase();
  while (!(t = br.readLine()).equals("EOF")) {
    strArray = t.split(" ");
    for (int i = 0; i < strArray.length; i++) {
      if (strArray[i].toLowerCase().equals(w)) {
        n++;
      }
    }
  }
  System.out.println(n);
}
```

Figure 2. **Example Java code snippets.** Each belonged to one subcategory and its corresponding category shown in Figure.1a.

Table 2. **Statistics of Java code snippets for each category class.** Numerics from 3rd (LOC) to last columns denote 'MEAN±SD'. Abbreviations: LOC: Lines of code, CPL: Max number of characters per line. One-way ANOVA revealed significant differences in mean values of LOC ($F(3,68) = 10.33$, $p = 0.00004$) and number of total characters ($F(3,68) = 8.14$, $p = 0.0002$) across the categories; but no significant difference in CPL ($F(3,68) = 0.26$, $p = 0.85$).

Category	N	LOC	CPL	Total characters
Math	18	25.5 ± 2.5	60.5 ± 16.0	373 ± 58.6
Search	18	26.1 ± 1.8	59.8 ± 20.9	446 ± 109.2
Sort	18	28.3 ± 1.5	60.6 ± 15.4	479 ± 76.0
String	18	25.6 ± 2.5	56.4 ± 16.6	386 ± 84.0
All	72	26.4 ± 2.4	59.3 ± 17.1	421 ± 93.0

3.4 MRI data acquisition

MRI data were collected using a 3-Tesla Siemens MAGNETOM Prisma scanner with a 64-channel head coil located at CiNet. T2*-weighted multiband gradient echo-EPI sequences were performed to acquire functional images covering the entire brain (repetition time (TR) = 2000 ms, echo time (TE) = 30 ms, flip angle = 75°, field of view (FOV) = 192 × 192 mm^2, slice thickness = 2 mm, slice gap = 0 mm, voxel size = 2 × 2 × 2.01 mm^3, multi-band factor = 3). A T1-weighted magnetization-prepared rapid acquisition with a gradient-echo sequence was also performed to acquire fine-structural images of the entire head (TR = 2530 ms, TE = 3.26 ms, flip angle = 9°, FOV = 256 × 256 mm^2, slice thickness = 1 mm, slice gap = 0 mm, voxel size = 1 × 1 × 1 mm^3).

Table 3. **Statistics of Java code snippets for each subcategory class.** Numerics from 3rd (LOC) to last columns denote 'MEAN±SD'. Abbreviations: GCD: Greatest common divisor. One-way ANOVA revealed significant differences in mean values of LOC ($F(10,61) = 5.44$, $p = 0.0004$) and number of total characters ($F(10,61) = 2.99$, $p = 0.014$) across the subcategories; but no significant difference in CPL ($F(10,61) = 0.44$, $p = 0.91$).

Subcategory	N	LOC	CPL	Total characters
GCD	6	24.8 ± 2.2	59.7 ± 19.7	339 ± 51.7
Power	6	25.2 ± 3.4	57.2 ± 13.9	391 ± 73.4
Primality test	6	26.5 ± 1.5	64.8 ± 15.8	391 ± 37.9
Binary search	6	27.2 ± 2.4	63.5 ± 18.6	509 ± 102.7
Linear search	12	25.6 ± 1.2	57.9 ± 22.5	414 ± 101.5
Bubble sort	6	28.7 ± 1.8	62.0 ± 17.8	503 ± 89.9
Insertion sort	6	27.2 ± 0.8	65.8 ± 15.0	471 ± 66.9
Selection sort	6	29.2 ± 1.0	54.0 ± 13.0	463 ± 77.3
Run length encode	6	26.5 ± 3.2	55.0 ± 12.8	393 ± 74.5
String sort	6	25.0 ± 2.2	52.2 ± 14.4	344 ± 77.6
Substring search	6	25.2 ± 1.9	62.0 ± 22.4	422 ± 93.2

3.5 MRI data preprocessing

The Statistical Parametric Mapping toolbox (SPM12, `http://www.fil.ion.ucl.ac.uk/spm/`) was used for preprocessing. The first eight scans in dummy trials for each run were discarded to avoid MRI signal instability. The functional scans were aligned to the first volume in the fourth run to remove movement artifacts. They were then slice-time corrected and co-registered to the whole-head T1 structural image. Both anatomical and functional images were spatially normalized into the standard Montreal Neurological Institute 152-brain average template space and resampled to a voxel size of $2 \times 2 \times 2$ mm^3. MRI signals at each voxel were high-pass–filtered with a cutoff period of 128 seconds to remove low-frequency drifts. A thick gray matter mask was obtained from the normalized anatomical images of all subjects to select the voxels within neuronal tissue

Table 4. **Description of category classes provided to subjects.**

Category	Description
Math	Applying number theory processing on given inputs.
Search	Identifying a specific item in a list of given inputs.
Sort	Arranging given inputs into a certain order.
String	Applying a specific operation on string inputs.

using the SPM Masking Toolbox [50]. For each subject independently, a general linear model (GLM) was then fitted to estimate voxel-level parameters (β) linking recorded MRI signals and conditions of source code presentations in each trial. The fixation and response phases in each trial were not explicitly modeled. The model also included motion realignment parameters to regress-out signal variations due to head motion. Finally, 216 beta estimate maps (36 trials $\times$ 6 runs) per subject were yielded and used as input for the following multivariate pattern analysis.

Table 5. **Description of subcategory classes provided to subjects.**

Subcategory	Description
GCD	Finding the greatest common divisor of two given natural numbers.
Power	Calculating the powers of two given integers m and n, i.e. n-th power of m.
Primality test	Judging whether the given natural number is a prime number or not.
Binary search	A process to search for a value from the given input sequence that is equal to the target value. This process first compares the target value with the middle value of the given sequence and specify the half of the sequence that potentially contains the target value. This process iterates the comparison and sequence division until the target value is found.
Linear search	A process to search for a value from the given input sequence that is equal to the target value. This process examines in order from the first element of the given series.
Bubble sort	Arranging given inputs in a certain order by exchanging adjacent elements if they are in wrong order.
Insertion sort	Arranging given inputs in a certain order by iteratively picking up an element from the given input sequence and insert it to the correct location.
Selection sort	A process to arrange given inputs in a certain order. This process first identify the smallest value of the given inputs and exchange it with the value in the first order. Then, identify the second smallest value and exchange it with the value in the second order. This process iterates this operation until the given inputs are correctly sorted.
Run length encode	A process to compress the given string sequence by replacing a sequence of the same character with the character and the number of repetition. For example, the string sequence 'AAAABBBAABBBB' will be compressed as '4A3B2A4B'.
String sort	Sorting the given words in lexicographic order.
Substring search	Detecting where the given pattern appears in the given string sequence.

3.6 Multi-voxel pattern analysis

This study used whole-brain searchlight analysis [23] to examine where significant decoding accuracies exist using the Decoding Toolbox [51] (version 3.99) and LIBSVM [52] (version 3.17). A four-voxel-radius sphered searchlight, covering 251 voxels at once, was systematically shifted throughout the brain and decoding accuracy was quantified on each searchlight location (see Fig.1c for overview of the entire framework). A linear-kernel support vector machine (SVM) classifier was trained and evaluated using a leave-one-run-out cross-validation procedure, which iteratively treated data in a single run for testing and the others for training. In each fold, training data was first scaled to zero-mean and unit variance by z-transform and test data was scaled using the estimated scaling parameters. Outlier reduction using [-3, +3] as cut-off values was then applied and all scaled signals larger than the upper cut-off or smaller than the lower cut-off were set to the closest value of these limits. The SVM classifier was trained with three cost parameter candidates [0.1, 1, 10], which control the tradeoff between margin maximization and the tolerance of misclassification rate in the training step, and the best parameter was chosen by a grid search in nested cross-validations. The outlier boundary and cost parameter candidates were selected based on the estimated computational load and the documents of tools employed. Specifically, this study adopted a relatively small set of parameter candidates due to the constraint of the high computational load of searchlight analysis. Finally, the trained classifier predicted *category* or *subcategory* of seen source code from the leave-out test data and decoding accuracy was calculated as a ratio of correct-classifications out of all-classifications. Note that corrected misclassification cost weights were used in *subcategory* decoding to compensate for the imbalanced number of exemplars across *subcategory* classes.cite

The training and evaluation procedures were performed independently for each subject and a whole-brain decoding accuracy map was obtained per subject. Second-level analyses was then conducted to examine the significance of decoding accuracies and the correlations between individual decoding accuracies and behavioral performances. For this purpose, the decoding accuracy maps were spatially smoothed using a Gaussian kernel of 6 mm full-width at half maximum (FWHM) and submitted to random effects analysis as implemented in SPM12.

The analysis tested the significance of group-level decoding accuracy and Pearson's correlation coefficient between individual decoding accuracies and behavioral performances. A relatively strict statistical threshold of voxel-level $p < 0.05$ FWE-corrected was used for decoding accuracy tests and a standard threshold of voxel-level $p < 0.001$ uncorrected and cluster-level $p < 0.05$ FWE-corrected was used for correlation tests. The chance-level accuracy (25% in *category* decoding and 9.72% in *subcategory* decoding; adjusted for imbalanced numbers of exemplar) and zero correlation were adopted as null hypotheses. Finally, the resultant significant searchlight maps, i.e. decoding accuracy map and correlation map to behavioral performance, were superimposed on a single cortical surface of the ICBM152 template brain using BrainNet viewer [53]. This overlapping analysis was performed to identify the searchlight centers that had both sufficient information to represent functional categories of source code and significant correlation to individual behavioral performance.

3.7 Balancing visual confounding in decoding analysis

A fundamental limitation of decoding analyses is the lack of direct measure for interpreting which source of information drives decoding accuracy and it becomes problematic when the target variable is confounded by variables that are not of primary interest [54]. In the experiment, the primitive visual features in code stimuli showed systematic difference: the mean values of LOC ($F(3,68) = 10.33$, $p = 0.00004$) and number of total characters ($F(3,68) = 8.14$, $p = 0.0002$) across *category* classes (see Table 2). From the view of *subcategory*, the similar tendency of significant differences was found in mean values of LOC ($F(10,61) = 5.44$, $p = 0.0004$) and number of total characters ($F(10,61) = 2.99$, $p = 0.014$) shown in Table 3. The observed systematic difference is problematic because the target variables (i.e. *category* and *subcategory*) in the decoding framework are confounded by the visual features that are not of our primary interest.

To isolate the potential effect of the visual confounding, this study created another codeset in which the samples of each *category* class were balanced to eliminate the systematic difference in the visual features. Specifically, nine snippets in *Sort* category and one in *Search* category were excluded from the original codeset. Table 6 shows the detailed statistics of the balanced codeset and one-way ANOVA revealed that the dataset had no significant difference in the visual features: the mean values of LOC ($F(3,58) = 1.92$, $p = 0.137$), CPL ($F(3,58) = 0.27$, $p = 0.850$) and number of total characters ($F(3,58) = 2.64$, $p = 0.058$) across *category* classes. For the difference across *subcategory* classes, the balanced codeset showed no significant difference: the mean values of LOC ($F(10,51) = 1.20$, $p = 0.31$), CPL ($F(10,51) = 0.47$, $p = 0.90$) and number of total characters ($F(10,51) = 1.66$, $p = 0.116$).

This study built a balanced MRI dataset for each subject by collecting only MRI data associated with the snippets in the balanced codeset. The balanced dataset consisted of 186 samples (62 snippets $\times$ 3 trials) and searchlight analyses were performed using the dataset on the same procedure as the original dataset. The chance-level decoding accuracy on the balanced dataset was 26.48% adjusted for imbalanced numbers of exemplar. If the visual confounding has little or limited effect on decoding accuracies, similar results (i.e. decoding accuracy maps) will be obtained from the searchlight analyses regardless of using the original or balanced

dataset.

3.8 Data and code availability

The experimental data and code used in the study are available from the repository: `https://github.com/Yoshiharu-Ikutani/DecodingCodeFromTheBrain`.

Table 6. **Statistics of the balanced dataset for visual confounding control.** Numerics from 3rd (LOC) to last columns denote 'MEAN±SD'. No significant difference was observed in mean values of LOC ($F(3,58) = 1.92$, $p = 0.137$), CPL ($F(3,58) = 0.27$, $p = 0.850$) and number of total characters ($F(3,58) = 2.64$, $p = 0.058$) across *category* classes. Numbers differed from original dataset shown in Table 2 are written in **bold**.

Category	N	LOC	CPL	Total characters
Math	18	25.5 ± 2.5	60.5 ± 16.0	373 ± 58.6
Search	**17**	**26.1 ± 1.9**	**58.3 ± 20.6**	**433 ± 100.1**
Sort	**9**	**27.4 ± 1.1**	**56.1 ± 14.7**	**437 ± 34.3**
String	18	25.6 ± 2.5	56.4 ± 16.6	386 ± 84.0
All	**62**	**26.4 ± 2.4**	**59.3 ± 17.1**	**421 ± 93.0**

4. Results

4.1 Behavioral data

The relationship between the adopted reference of programming expertise and behavioral performance on the program categorization task was first evaluated. A significant correlation was observed between AtCoder rate (M = 954.3, SD = 864.6) and behavioral performance in the fMRI experiments (M = 76.0, SD = 13.5 [%]), r = 0.593, p = 0.0059, n = 20 (Fig.3a). The correlation remained significant if the behavioral performances of non-rate-holders (i.e. novices) were included as zero-rated subjects; r = 0.722, p = 0.000007, n = 30. A positive correlation was found between AtCoder rate and behavioral performance on subcategory categorization in the post-MRI experiments (M = 65.9, SD = 17.0 [%]), r = 0.688, p = 0.0008, n = 20 (Fig.3b). The significant correlation also remained significant if non-rate-holder subjects were included; r = 0.735, p = 0.000004, n = 30. From all behavioral data, this study concluded that behavioral performances on the program categorization task significantly correlated with expertise of competitive programming. The behavioral evidence allowed us to study the potential association between experts' outstanding performances and brain activity patterns measured using fMRI while subjects performed this task.

Difference in cognitive demands induced by code snippets of different *category* classes was potential confounding variable for the decoding framework. Although this study did not have a direct indicator of cognitive demands across *category* classes, the difference in behavioral performances for each *category* can be a clue to assess the extent of cognitive demand. Table.7 shows the group-wise behavioral performances for each individual *category* class. One-way ANOVA found no significant difference in behavioral performances between individual classes of *category* for any potential grouping (Expert, $F(3,36) = 1.38$, $p = 0.27$; Middle, $F(3,36) = 2.99$, $p = 0.06$; Novice, $F(3,36) = 2.81$, $p = 0.07$; All, $F(3,116) = 2.02$, $p = 0.12$). Thus, this study considered that the difference in cognitive demands induced by code snippets of different *category* classes did not have a significant effect on the resulted decoding accuracies.

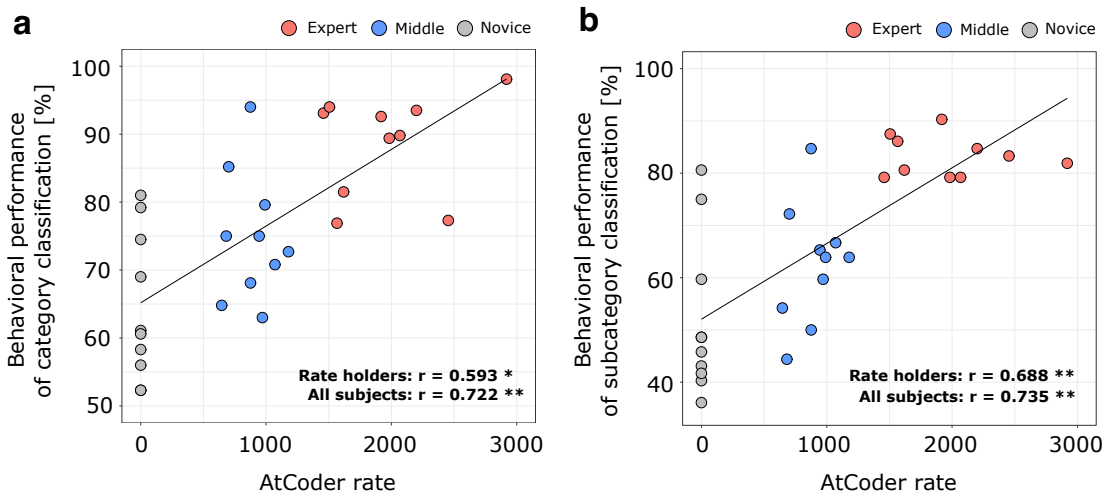

Figure 3. **Correlations between behavioral performance and programming expertise indicator.** (a) Scatter plot of behavioral performances of category classifications against the values of adopted expertise reference (i.e. AtCoder rate). (b) Scatter plot of behavioral performances of subcategory classifications against the values of the same expertise reference. Each dot represents an individual subject. Significance of the correlation coefficients (r) was denoted as *, $p < 0.05$ and **, $p < 0.005$. The solid lines indicate a fitted regression line estimated from all subject data.

Table 7. **Behavioral performance of each category in the fMRI experiment.** Numerics from 3rd (Math) to last columns denote 'MEAN±SD'. One-way ANOVA found no significant difference in behavioral performances between categories for any groupings (Expert, $F(3,36) = 1.38$, $p = 0.27$; Middle, $F(3,36) = 2.99$, $p = 0.06$; Novice, $F(3,36) = 2.81$, $p = 0.07$; All, $F(3,116) = 2.02$, $p = 0.12$).

	N	Math	Search	Sort	String
Expert	10	93.8 ± 11.4	89.4 ± 11.8	84.8 ± 9.2	86.3 ± 8.4
Middle	10	68.1 ± 15.0	77.8 ± 16.2	70.2 ± 15.6	83.1 ± 9.5
Novice	10	72.2 ± 15.2	53.0 ± 20.1	60.4 ± 19.1	72.2 ± 12.5
All	30	78.1 ± 17.7	73.4 ± 22.1	71.8 ± 17.9	80.6 ± 11.6

4.2 Mutli-voxel activity patterns associated with programming expertise

This study next examined where the functional categories of source code can be decoded from programmers' brain activity. Fig.4 visualizes the searchlight centers that showed significantly higher decoding accuracy than chance as estimated from all subject data using a relatively strict whole-brain statistical threshold (voxel-level $p < 0.05$ FWE-corrected). The figure shows that significant decoding accuracies were observed in the broad areas of the bilateral occipital cortices, parietal cortices, posterior and ventral temporal cortices, as well as the bilateral frontal cortices around inferior frontal gyri. The variances of peak decoding accuracies across individual subjects on six widely distributed brain regions were depicted in Figure 5. The figure ensured that the decoding accuracies of all or almost all subjects were higher than the chance-level accuracy with a reasonable margin. Given these results, this study confirmed that functional categories of source code were represented in the widely distributed brain areas and the cortical representations of each *category* class were linearly separable by a simple SVM classifier.

To associate the cortical representation of source code with individual programming expertise, a linear correlation was investigated between behavioral performances and decoding accuracies for each searchlight location. Fig.6a visualizes the searchlight centers that showed significantly high correlation coefficients using thresholds of voxel-level $p < 0.001$ uncorrected and cluster-level $p < 0.05$ FWE-corrected. Significant correlations were observed in the areas of bilateral inferior frontal gyri pars triangularis (IFG Tri), right superior frontal gyrus (SFG), left inferior parietal lobule (IPL), left middle and inferior temporal gyrus (MTG / IT); see the slice-width visualization shown as Fig.6b and Table 2 for the list of significant clusters. In this correlation analysis, the right IFG Tri showed the highest peak correlation coefficient. These results provided evidence that cortical representations in the distinct brain areas mainly located in frontal, parietal, and temporal cortices were significantly associated with experts' outstanding behavioral performances on the program categorization task. In contrast, cortical representations in the bilateral occipital cortices including early visual areas did not show a significant correlation to individual behavioral performances, while

Table 8. **Clusters showing significant correlations between behavioral performance and category decoding accuracy** (voxel-level $p < 0.001$ and cluster-level $p < 0.05$, FWE-corrected). Region names were identified using Automated anatomical labelling atlas 2 [55].

	MNI coordinates					
Region name	X	Y	Z	Corr. (r)	T-value	Extent
R IFG (p. Triangularis)	46	22	8	0.789	6.81	369
L Posterior-Medial Frontal	-12	0	66	0.711	5.36	298
R Superior Medial Gyrus	6	52	42	0.699	5.17	587
L Inferior Parietal Lobule	-56	-28	50	0.698	5.16	649
R Superior Frontal Gyrus	24	4	60	0.675	4.84	428
L IFG (p. Triangularis)	-52	30	24	0.671	4.79	346
L Inferior Temporal Gyrus	-50	-54	0	0.635	4.35	347

significant decoding accuracies were broadly observed in the cortices (see Fig.4).

The previous analyses separately showed where significant decoding accuracies exist and whether the decoding accuracies significantly correlate with behavioral performances. To achieve more validated evidence for the cortical representations associated with programming expertise, these two analyses were integrated to find the searchlight centers that had sufficient information to represent functional categories of source code and their decoding accuracies significantly correlated with individual behavioral performance. Specifically, the two significant searchlight maps, i.e. decoding accuracy map and correlation map to behavioral performance, were superimposed on a single cortical surface to investigate the overlap between them. As a result, 1205 searchlight centers (equal to 0.79%) were survived from both statistical thresholds of decoding accuracy and correlation to behavioral performances; shown as red-colored dots in Fig.7a. The survived searchlight centers were mainly observed in the bilateral IFG Tri, left IPL, left supramarginal gyrus (SMG), left MTG/IT, and right middle frontal gyrus (MFG) as shown in Fig.7b. These results revealed a tight association between superior behavioral performances of expert programmers and improvement of decoding accuracy in these distributed brain regions.

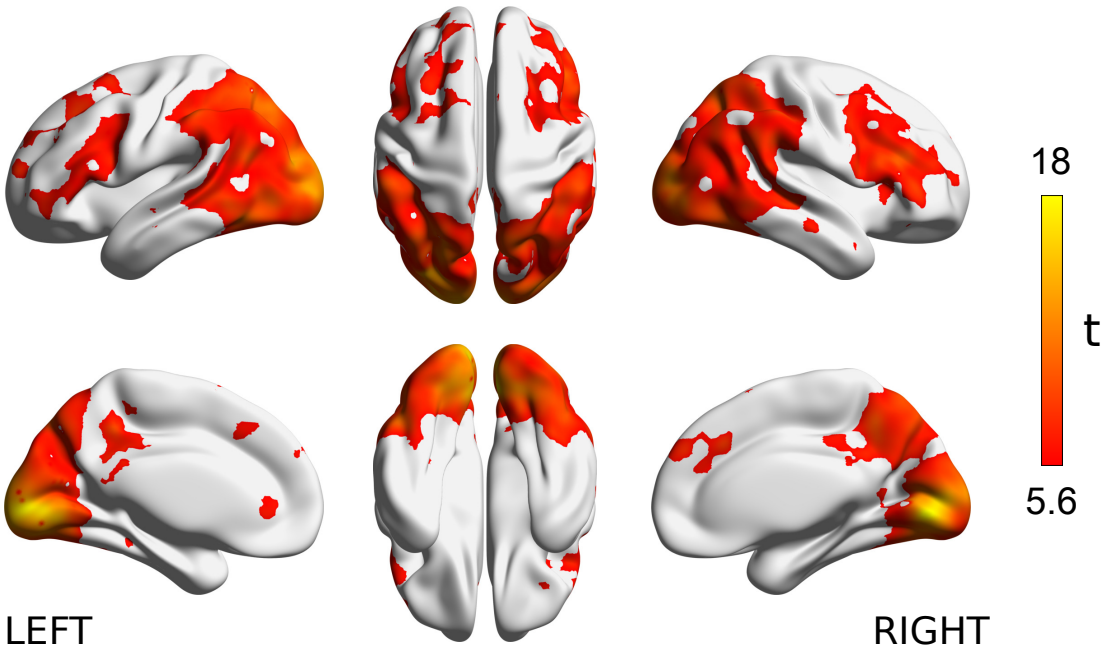

Figure 4. **Decoding accuracy for functional category of source code.** Significant searchlight locations estimated from all subject data (N = 30). Heat colored voxels denote the centers of searchlights with significant decoding accuracy (voxel-level $p < 0.05$, FWE corrected). The brain surface visualizations were performed using BrainNet viewer, version 1.61 [53].

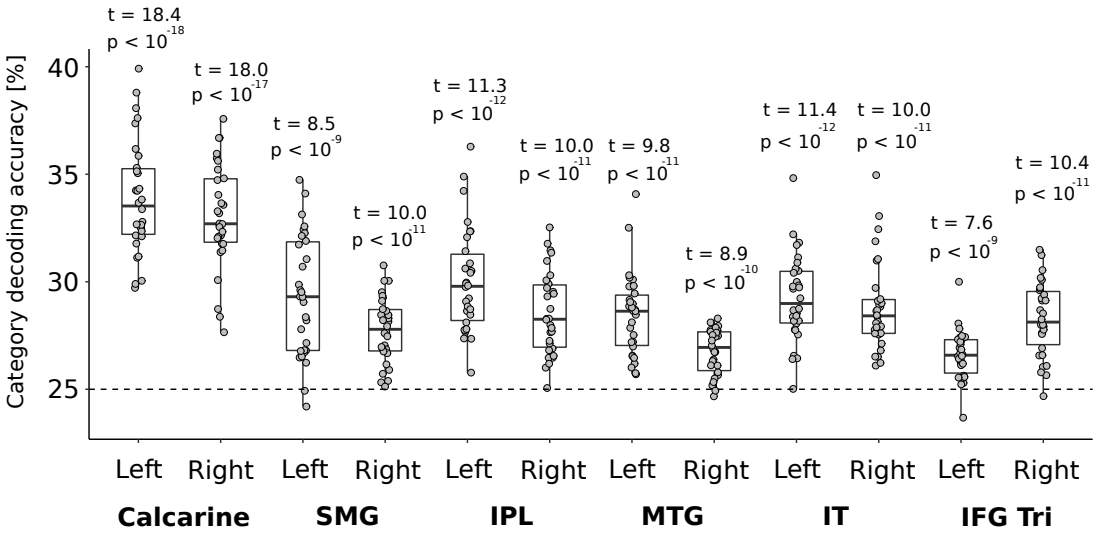

Figure 5. **Box plots of the voxel-level peak category decoding accuracies.** Each dot represents decoding accuracy of individual subject. The dashed line indicates chance-level accuracy (25%). Abbreviations: SMG, Supramarginal gyrus; IPL, Inferior parietal lobule; MTG, Middle temporal gyrus; IT, Inferior temporal gyrus; IFG Tri, Inferior frontal gyrus pars triangularis.

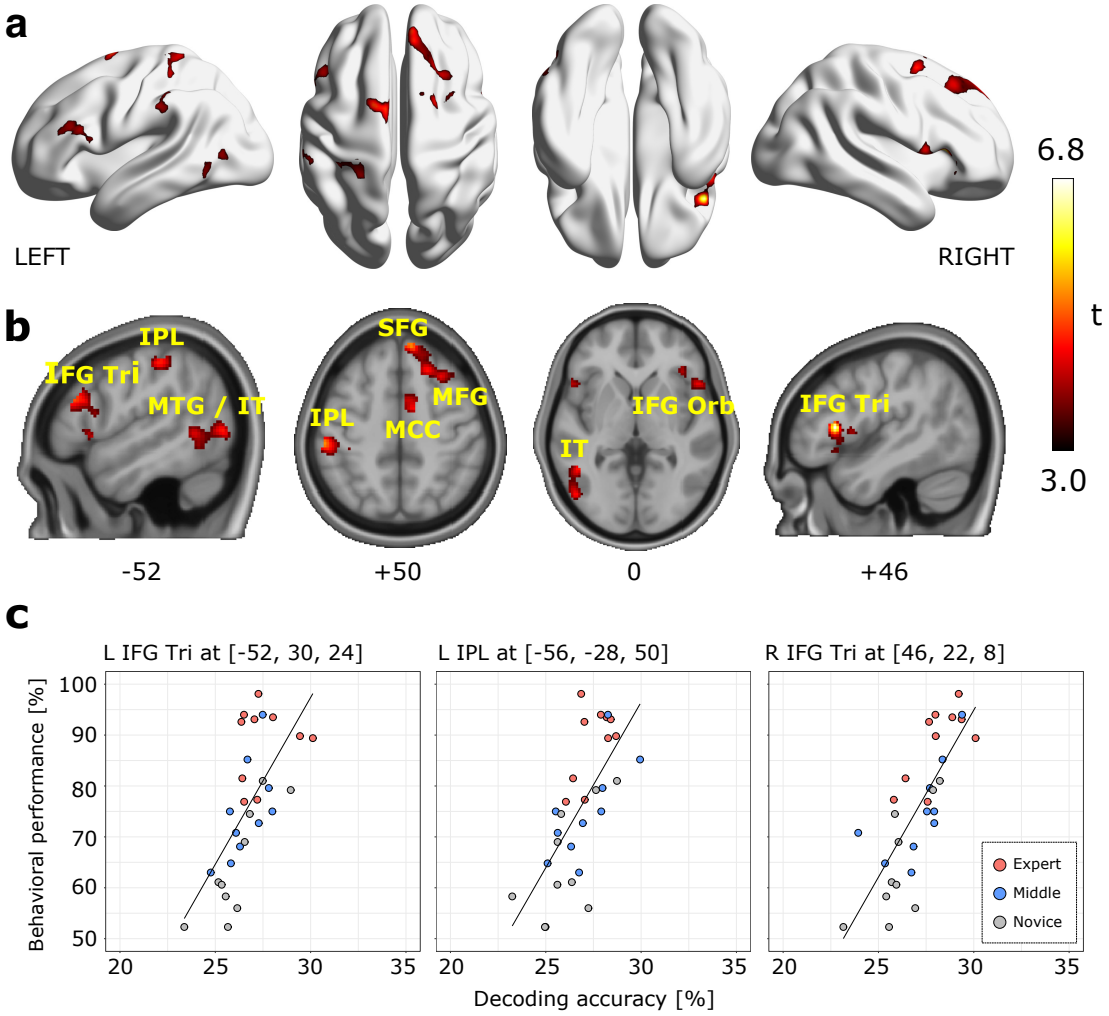

Figure 6. **Searchlight-based correlation analysis between behavioral performances and decoding accuracies.** (a) Locations of searchlight showing significant correlations. Significance was determined by a threshold of voxel-level $p < 0.001$ and cluster-level $p < 0.05$, FWE corrected for the whole brain. (b) Slice-wise visualizations of the significant clusters using bspmview. (c) Correlation between behavioral performance and decoding accuracy. Each dot represents an individual subject data. Abbreviations: SMG, Supramarginal gyrus; IPL, Inferior parietal lobule; MTG, Middle temporal gyrus; IT, Inferior temporal gyrus; SFG, Superior frontal gyrus; MFG, middle frontal gyrus; IFG Tri, Inferior frontal gyrus pars triangularis; IFG Orb, Inferior frontal gyrus pars orbitalis; MCC, medial cingulate cortex.

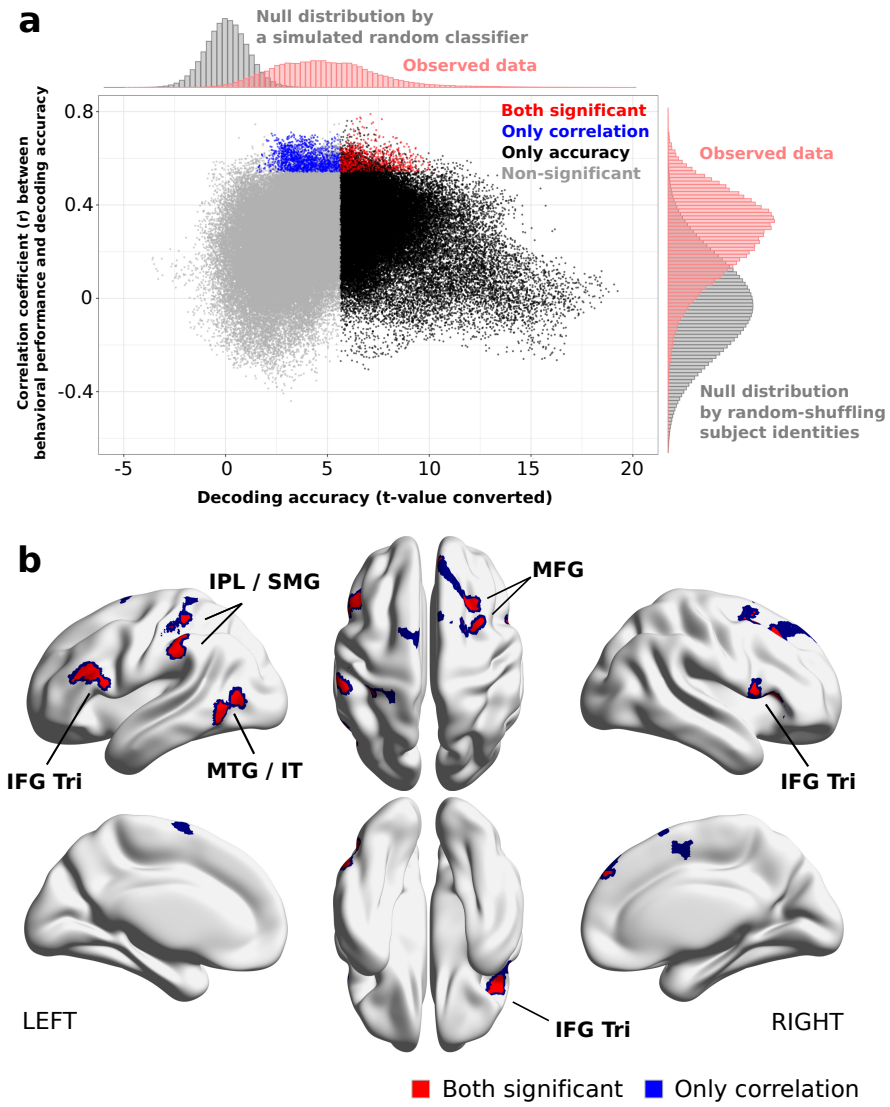

Figure 7. **Identifying searchlight centers that showed both significant decoding accuracy and significant correlation to individual behavioral performances.** (a) Scatter plot of searchlight results. The observed distributions of decoding accuracies and correlations are respectively shown on top- and right-sides of the figure accompanied with null distributions calculated by randomized simulations. (b) Locations of searchlight centers that showed both significant decoding accuracy and significant correlations to individual behavioral performances.

4.3 Cortical representations of subcategory information

This study next investigated where the *subcategory* of source code can be decoded from programmers' brain activity to examine finer-level cortical representations. In the experiment, subjects responded 'sort' when they had been presented with the code snippets implementing one of three different sorting algorithms; i.e. bubble, insertion, and selection sorts (see Fig.1a). This cognitive process could be considered as a generalization process that incorporates different but similar algorithms (*subcategory*) into a more general functionality class (*category*). Additionally, several psychologists indicated that experts specifically show high behavioral performances in subordinate-level categorizations as well as basic-level categorizations [56]. In fact, the behavioral evidence demonstrated that the ability to differentiate *subcategory* classes significantly correlated to programming expertise in competitive programming (see Fig.3b). This observation implies that the detailed difference of source code functionalities might be represented in programmers' brain activity patterns. The decoding accuracy of *subcategory* may be correlated with programming expertise, even though they classified only *category* classes, not *subcategory*, of given code snippets and the existence of *subcategory* classes had never been revealed until the end of the fMRI experiment.

Searchlight analysis was employed with the same setting as used in the previous analysis to reveal the spatial distribution of significant *subcategory* decoding accuracies and significant correlations to behavioral performances. Fig.8 illustrates the searchlight centers that showed significantly higher *subcategory* decoding accuracy than chance (9.72%; corrected for imbalanced exemplars) using a threshold of voxel-level $p < 0.05$ FWE-corrected. The variances of peak *subcategory* decoding accuracies across individual subjects on six widely distributed brain regions were depicted in Figure 9. The linear correlation between *subcategory* decoding accuracies and individual behavioral performances was then assessed using thresholds of voxel-level $p < 0.001$ uncorrected and cluster-level $p < 0.05$ FWE-corrected. Fig.10 visualizes the result and indicates that only a cluster on the left SMG and superior temporal gyrus (STG) showed a significant correlation; the peak correlation coefficient was observed in the left STG. Finally, the results from decoding and correlation analysis of *subcategory* were integrated and demonstrated that 120 searchlight centers (equal to 0.08%) on the left SMG

and STG survived from both statistical thresholds of decoding accuracy and correlation to behavioral performances; shown as red-colored dots in Fig.11a. These results suggest that cortical representations of fine functional categories on the left SMG and STG may play an important role in achieving advanced-level programming expertise, even though the representations are not explicitly required by the tasks.

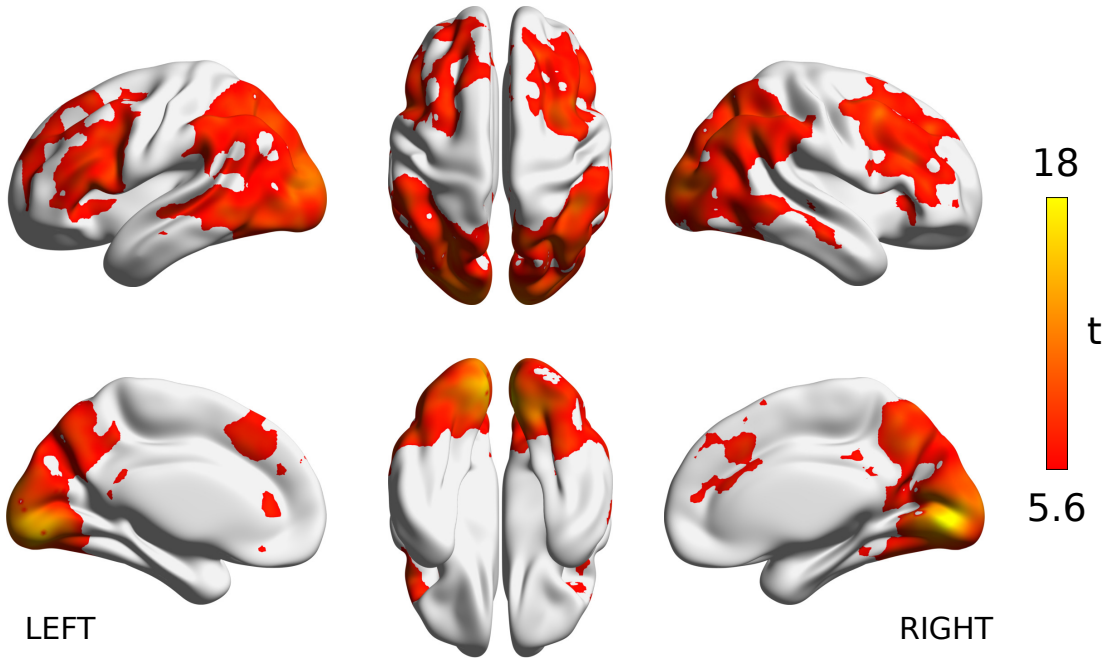

Figure 8. **Decoding accuracy for subcategory of source code.** Searchlight locations showing significant subcategory decoding accuracy than chance estimated from all subject data (N = 30). Heat colored voxels denote the centers of searchlights with significant subcategory decoding accuracy (voxel-level $p < 0.05$, FWE corrected).

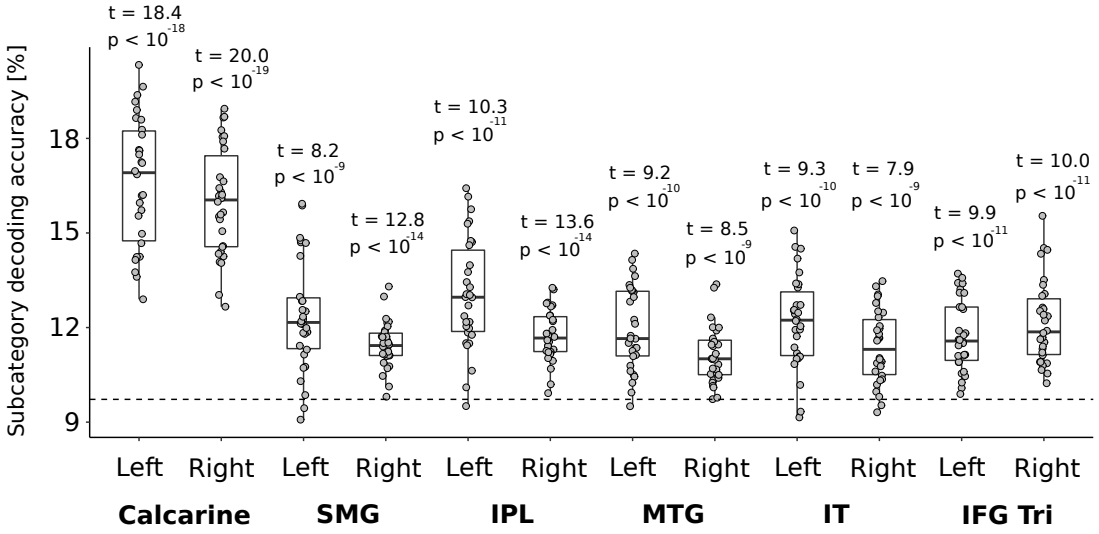

Figure 9. **Box plots of the voxel-level peak subcategory decoding accuracies.** Each dot represents decoding accuracy of individual subject. The dashed line indicates chance-level accuracy (9.72%). Abbreviations are same as Fig.5.

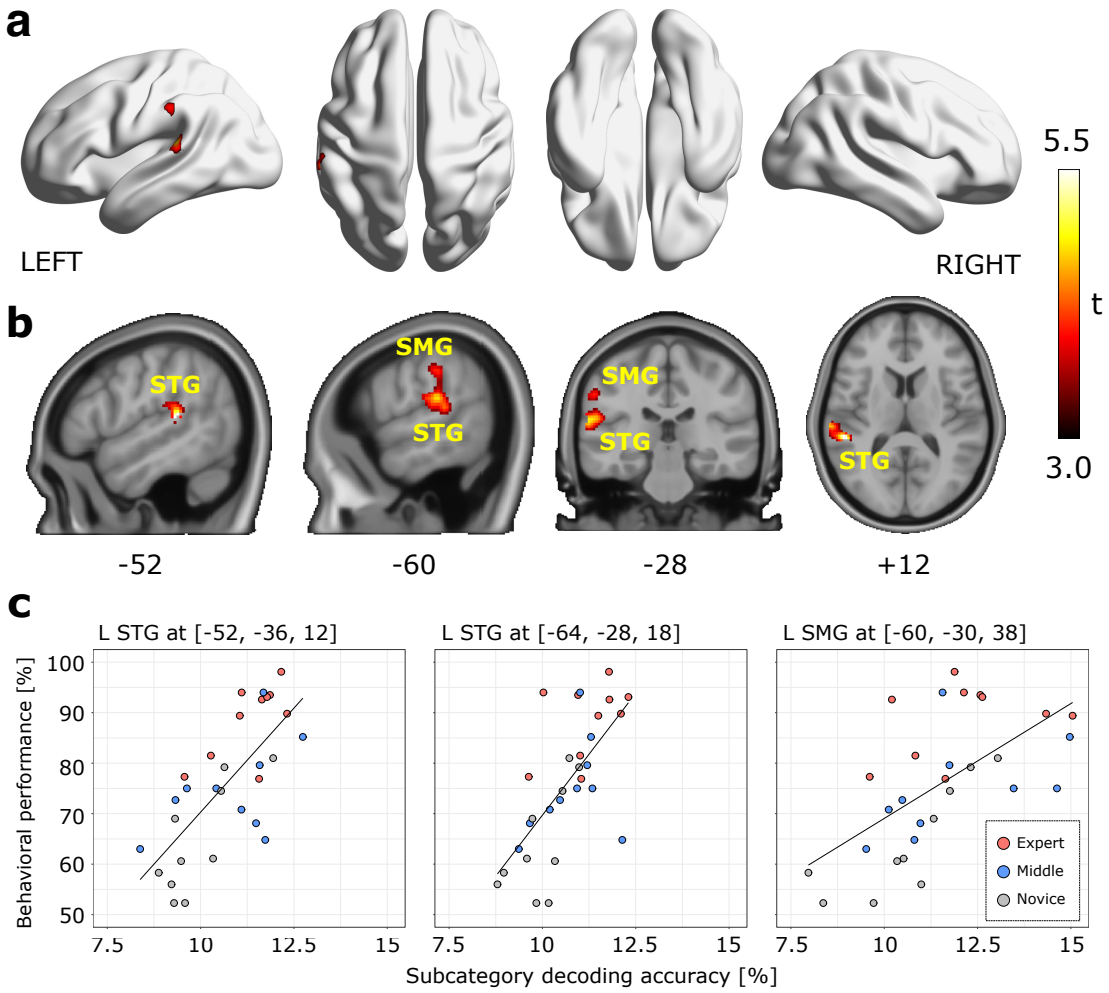

Figure 10. **Searchlight-based correlation analysis between behavioral performances and subcategory decoding accuracies.** (a) Locations of searchlight showing significant correlations. Significance was determined by a threshold of voxel-level $p < 0.001$ and cluster-level $p < 0.05$, FWE corrected for the whole brain. (b) Slice-wise visualizations of the significant clusters. (c) Correlation between behavioral performance and decoding accuracy. Each dot represents an individual subject data. Only one cluster (extent = 501 voxels) had significant correlation in this analysis and three peak correlations in the cluster were shown here. Abbreviations: STG, Superior temporal gyrus.

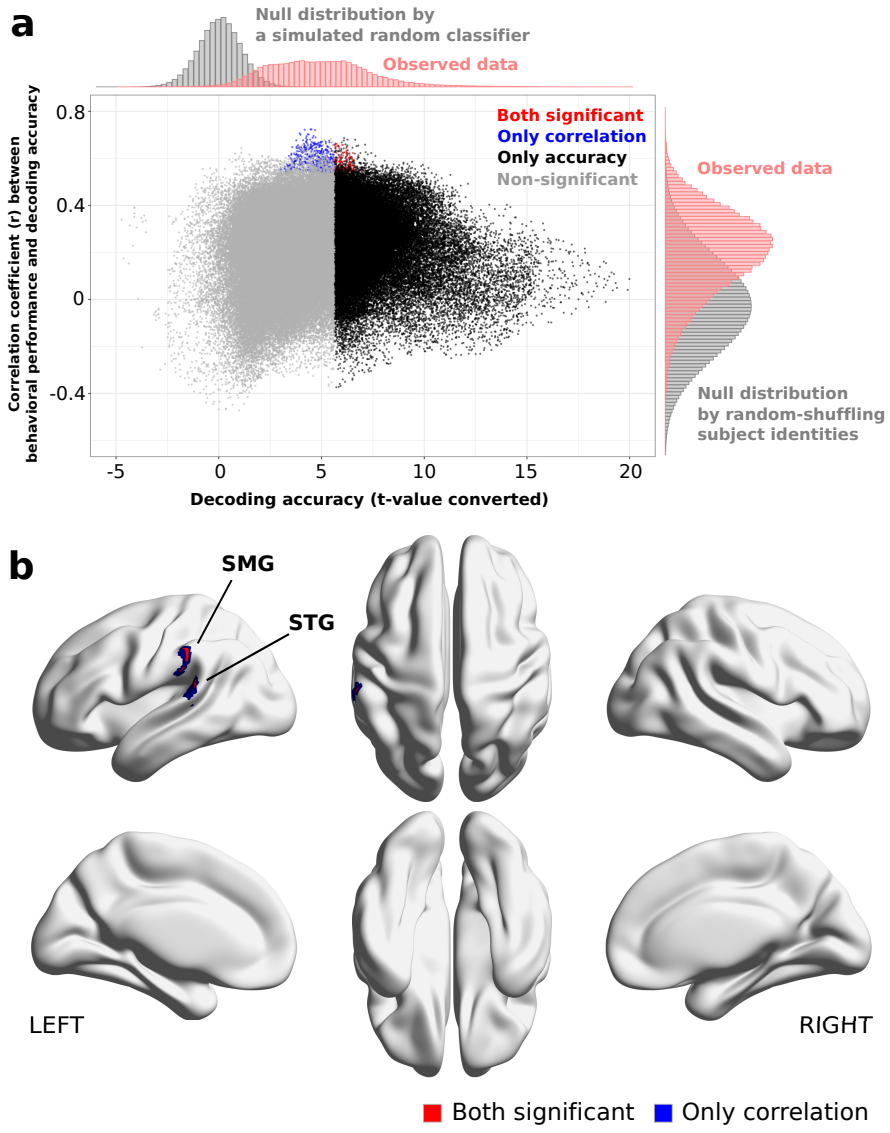

Figure 11. **Identifying searchlight centers that showed both significant subcategory decoding accuracy and significant correlation to individual behavioral performances.** (a) Scatter plot of searchlight results. The observed distributions of subcategory decoding accuracies and correlations are respectively shown on top- and right-sides of the figure accompanied with null distributions calculated by randomized simulations. (b) Locations of searchlight centers that showed both significant subcategory decoding accuracy and significant correlations to individual behavioral performances.

4.4 Decoding accuracy on the visual confounding controlled data

This study additionally performed searchlight analysis using the visual confounding balanced dataset to quantify the effects of primitive visual features in the code stimuli (see Section 3.7). Figure 12a visualizes the searchlight centers that showed significantly higher *category* decoding accuracy than chance (26.48%; corrected for imbalanced exemplars) using a threshold of voxel-level $p < 0.05$ FWE-corrected. The result obtained from the balanced dataset seemed to be almost the same as the result from the original dataset shown as Fig.4. The decoding accuracies on the bilateral occipital cortices, parietal cortices, posterior and ventral temporal cortices, as well as the bilateral frontal cortices remained significant if the systematic differences in the primitive visual features were removed. Figure 12b shows the difference in t-value maps obtained using original and balanced datasets. The figure suggests that the operation of visual confounding control mainly decreased the decoding accuracies on the primary visual areas. This observation was not surprising because the representations of primitive visual features in experimental stimuli are typically reflected in the primary visual areas [57]. These results indicate that the decoding accuracies observed in this study were not significantly driven by the primitive visual features including LOC, CPL, and number of total characters. Note that the visualization of difference in t-value maps (Fig.12b) was created only for visualization purpose and was not tested in their significance because a valid statistical test was not found to examine the difference in t-value maps obtained from whole-brain searchlight analyses.

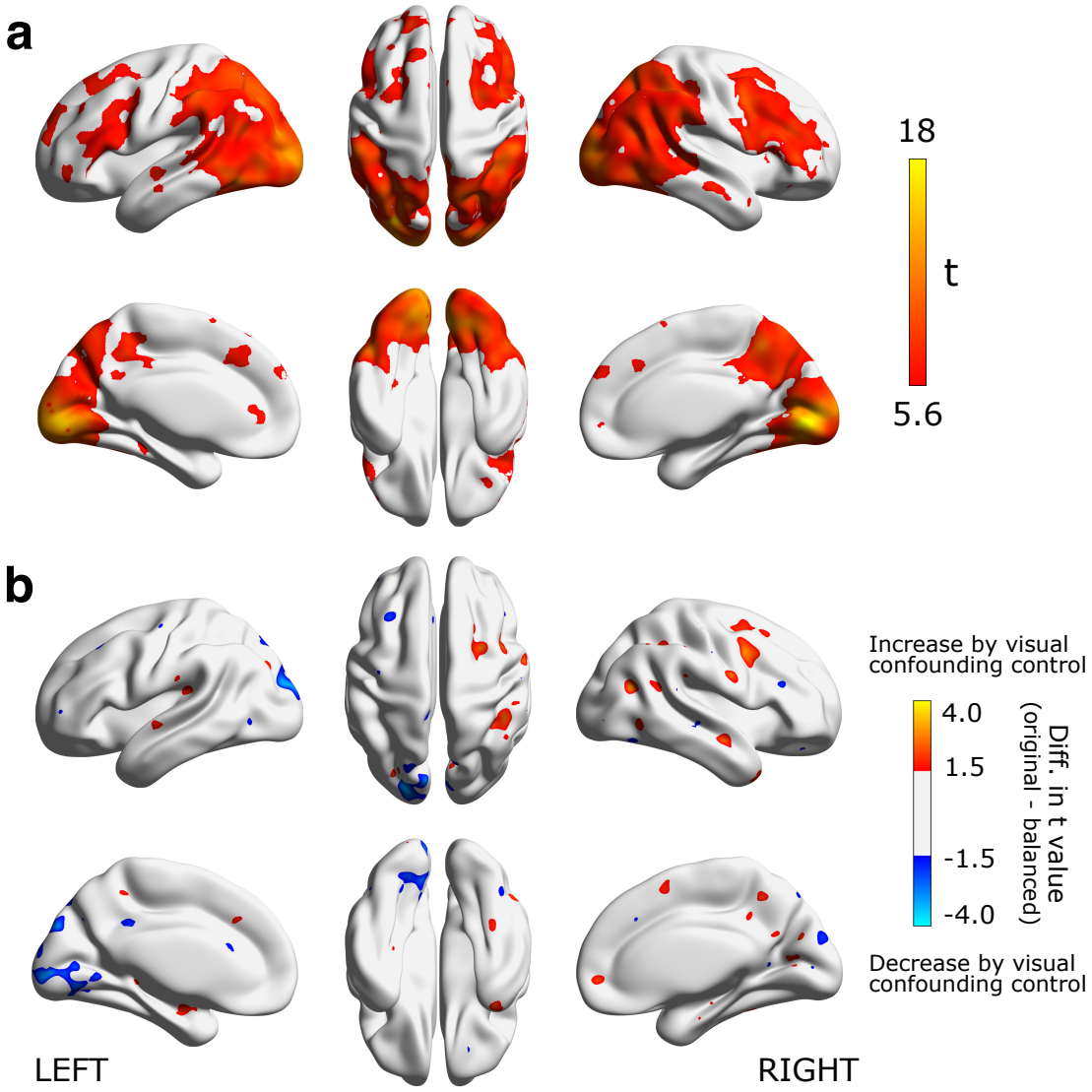

Figure 12. **Decoding accuracy of category using the visual confounding balanced data.** (a) Searchlight locations showing significant category decoding accuracy than chance using balanced data controlling for visual confounding. Heat colored voxels denote the centers of searchlights with significant subcategory decoding accuracy (N = 30; voxel-level $p < 0.05$, FWE corrected). (b) Difference in t value between results using original and balanced datasets. The image of t-value difference is only for visualization purpose.

5. Discussion

5.1 Summary of findings

This thesis demonstrated that functional categories of source code can be decoded from programmers' brain activity measured using fMRI. Decoding accuracies on the bilateral inferior frontal gyrus pars triangularis, left inferior parietal lobule, left supramarginal gyrus, left middle and inferior temporal gyri, and right middle frontal gyrus were significantly correlated with individual behavioral performances on the program categorization task. Furthermore, decoding accuracies of subcategory on the left supramarginal and superior temporal gyri were also strongly correlated with the behavioral performances while the subordinate-level representations were not directly induced by the performing tasks. The results revealed an association between the outstanding performances of expert programmers and domain-specific cortical representations in these brain areas widely distributed in the frontal, parietal, and temporal cortices.

5.2 Cortical representations and programming expertise

Previous fMRI studies on programmers have aimed at characterizing how programming related activities, such as program comprehension and bug detection, take place in the brain [14–17, 58, 59]. Exceptionally, an exploratory study reported that BOLD signal discriminability between code and text comprehension was negatively correlated with participants' GPA scores in a university [17]. However, the relationship between GPA scores and programming expertise was ambiguous and the observed correlation was relatively small ($r = -0.44$, $p = 0.016$, $n = 29$). The aim in the present study was substantially different: This study sought the neural bases of programming expertise that contribute to expert programmers' outstanding performances. To address the goal, the study adopted an objectively-determined reference of programming expertise and recruited a population of subjects covering a wide range of programming expertise. Despite the difference in research aims, a subset of brain regions specified in this study was similar to those specified by prior fMRI studies on programmers [14–16]. In particular, this study associated the left IFG, MTG, IPL, SMG with programming expertise while previous studies related them with program comprehension processes. This commonality may suggest that both program comprehension processes and its related expertise depend on the same set of brain regions.cite

The potential roles of the specified brain regions in this study should be addressed to orient future researches on programming activity and expertise. First, the left IFG Tri and the left posterior MTG are frequently involved in semantic selecting/retrieving tasks [60–63]. Several studies indicated that these two regions are sensitive to cognitive demands for directing semantic knowledge retrieval in a goal-oriented way [64–66]. The involvement of the two regions may be induced by similar demands specialized for the retrieval of program functional categories and suggest that higher programming expertise is related to the abilities of goal-oriented knowledge retrieval. Second, many neuroscientists have shown the left IPL and SMG to be functionally related to visual word reading [67–69] and episodic memory retrieval [70–72]. Both cognitive functions potentially relate to the program categorization task used in the experiment. Visual word reading can be naturally engaged since source code is comprised of many English-like words and subjects may have actively recollected previously-acquired memories to com-

pensate for insufficient clues because they had only ten seconds to categorize the given code snippet. The involvements of the left IPL and SMG suggest that expert programmers might possess different reading strategies and/or depend more on domain-specific memory retrieval than novices.

The set of IFG and IPL has been frequently discussed together as a fronto-parietal network and they often show synchronous activity in a wide range of tasks [73, 74]. Importantly, a recent fMRI study on programmers suggested an association between program comprehension and fronto-parietal network that was functionally related to formal logical inference [40]. The results observed in this study are consistent with these findings, implying that the fronto-parietal network plays a key role in experts' program comprehension processes. In addition, this thesis showed the involvement of the left MTG in programming expertise. Martin *et al.* demonstrated that recall of action words associated with a tool activated a region in the left MTG selectively [75] and several studies suggest that object categories, especially for tools, are potentially represented in the left MTG [76,77]. From this point of view, the finding suggest that the region might represent categorical knowledge that maps the given code snippet onto its corresponding functionality, in a similar way to a mapping from a tool to its functionality.

Other novel findings in the present study included potential involvement of the left IT, right MFG, and right IFG Tri with programming expertise. Importantly, these regions were not specified by previous studies focusing on the relationship between brain activity and program comprehension processes of non-expert subjects [14–17], suggesting that the regions might be more related to expert programmers' program comprehension processes. Because the left IT is well known for the function in high-level visual processing including word recognition and categorical object representations [78–80], the results may suggest that the high-level visual cortex in expert programmers could be fine-tuned by their training experience to realize faster program comprehension process. From another perspective, the observed map involving the left IFG Tri, IPL, and MTG/IT (Fig.6a) could be associated with a semantic system in the brain [81,82]. The results might suggest that an expert programmer's brain recruits a similar language-related network for both natural language processing and program comprehension.

In contrast, the primary visual area showed significant decoding accuracy but

no correlation to programming expertise. Multiple studies demonstrated that visual perceptual learning can occur for task-irrelevant stimulus features and can unconsciously modulate the activity patterns in primary visual area [83,84]. Such visual perceptual learning might have occurred in our study and made us find significantly high decoding accuracies on the primary visual area regardless of individual levels of programming expertise. From another perspective, the primary visual area mainly reflects primitive visual features such as color, contrast, spatial frequency [57] while computations in the high-level visual cortex are characterized by both bottom-up (i.e. how stimuli are visually represented) and top-down (how the representation is used for a cognitive task) effects [85]. Previous studies indicated that fine-tuned representations in the high-level visual cortex, rather than in the primary visual area, could be associated with visual expertise [21] and reading skill [86]. In the experiment, the primary visual area represented a large amount of visual information regardless of programming expertise levels because all subjects were presented with the same set of code snippets inducing similar visual patterns on their retinas. Therefore, the information in the primary visual area was sufficient to decode *category* and *subcategory* classes but the decoding accuracies were not necessarily to be correlated with individual behavioral performances. Meanwhile, the amount of information represented in the high-level visual cortex might be modulated by individual programming expertise. In line with previous expertise studies, this study imply that expertise in program comprehension could be mainly associated with high-level visual perception.

The right MFG and IFG Tri are functionally related to stimulus-driven attention control [87, 88]. The involvement of these two regions suggests that programmers with high-level programming expertise may employ different attention strategies than less-skilled ones. Moreover, additional engagements of right hemisphere regions in experts are common across expertise studies. For example, chess experts [89] and abacus experts [90, 91] showed additional right hemisphere region involvements when performing their domain-specific tasks. Several fMRI studies further suggest that such activation shifts from left to right hemisphere may be related to experts' cognitive strategy changes [89, 92]. Cognitive strategy changes have been observed repeatedly in comparisons between expert and novice programmers: A major characteristic is a transition from bottom-up (or

textual-driven) to top-down (or goal-driven) program comprehension, which becomes feasible by experts' domain-specific knowledge [8–10]. The involvement of the right MFG and IFG Tri observed in this study might be related to such cognitive strategy differences between programmers in the program categorization task. From another perspective, activations in the prefrontal and parietal regions including bilateral IFG/MFG and left IPL have been associated with the extent of cognitive demands [93]. While this study did not have a direct indicator of cognitive demands across categories, the difference in behavioral performances for each *category* can be a clue to assess the extent of cognitive demand across the categories. This study used the one-way ANOVA to test the difference in mean behavioral performances between categories but no significant difference was found for any groupings (see Table 7). Although these results do not provide a direct indication of cognitive demands across categories, there is no positive evidence that the extent of cognitive demands had a significant effect on the observed decoding accuracies.

The results associated programming expertise with decoding accuracies of not only *category* but also *subcategory*, even though the subordinate-level categorizations were not explicitly required by the performing task. This study observed that individual behavioral performances were significantly correlated with *subcategory* decoding accuracies on the left STG and SMG. These two regions are functionally related to pre-lexical and phonological processing in natural language comprehension [60, 94, 95]. Interestingly, a significant correlation was found between behavioral performances and *category* decoding accuracies on the temporal regions (left MTG and IT) associated with more semantical processing [63, 64, 66]. If these functional interpretations could be adaptable to program comprehension processes, it would be intuitive that subordinate concrete concepts (i.e. *subcategory*) of source code are processed in the left STG/SMG and more semantically abstract concepts (i.e. *category*) are represented in the left MTG/IT. Further, Mkrtychian et al. have associated STG, MTG, and IFG with the processing of abstract concepts in their review on concreteness effects [96], implying that representations in these three regions could reflect relative differences in abstractness between the *category* and *subcategory* in this study. These interpretations might suggest a hypothesis that an expert programmer's brain

has a hierarchical semantic processing system to obtain mental representations of source code for multiple levels of abstraction.

5.3 Limitations of the study

The results obtained via the present study were limited to a specific type of programming expertise evaluated by the expertise reference and laboratory task used in the experiment. The study particularly examined the ability to semantically categorize source code that correlated with programming expertise to win high scores in competitive programming contests. Perhaps there is a qualitative gap between expertise in competitive programming and practical/industrial software development. For example, the ability to write efficient SQL programs, for example, may be an explicit indicator of another type of programming expertise; but this study did not cover such type of programming expertise. The program categorization task used in this study primarily evaluated the skill in recognizing algorithms quickly and accurately, which was one aspect of a wide range of cognitive skills that constitute programming expertise. The evaluated skill is related to program comprehension and is also connected to skills in code refactoring and debugging because these processes require a deep understanding of algorithms or how the code works; while its relation to writing code is not assessed in this study. Thus, the results should not be taken to imply the relationship between the neural correlates revealed here and other types of programming expertise that could not be examined by this experiment. However, it is also a fact that nobody can investigate the neural bases of programming expertise without a clear definition of expertise indicator and laboratory task that well fit the general constraints of fMRI experiments. To mitigate the potentially inevitable effects caused by this limitation, this study adopted the objectively-determined reference of programming expertise that directly reflects programmers' actual performances and recruited a population of subjects covering a wide range of programming expertise. This study can be a baseline for future researches to investigate the neural bases of programming expertise and related abilities.

The experiment described in the thesis, which was designed to fit the general constraints of fMRI measurement, might embrace several caveats to external validity. First, this study used the relatively small code snippets with 30 lines at maximum due to the constraint of the MRI screen size. Behavioral performances on system-level source code were not assessed in the study. Thus, generalizing the results to the expertise in systems-level program comprehension was not

guaranteed. Second, only Java code snippets were used as experimental stimuli in this study. The results obtained via the experiments might be biased by the programming language selected; for example, Python has more natural-language-like syntax than Java and might induce more activation in language-related brain regions. While a recent fMRI study has examined brain activities elicited by code written in two programming languages (Python and ScratchJr) [41], it is still unclear whether the choice of a specific programming language can alter an expert's brain activity pattern. The relationship between programming expertise and types of programming languages (e.g. procedural vs. functional languages) is expected to be examined in future work.

Another potential concern of the present study was the unavoidable gender balance in the subject population. While 95 programmers completed the entry questionnaire to be registered as candidate subjects, only one middle-level woman candidate and zero woman expert were found (see Subjects section in Materials and Methods). From this situation, the unavoidable gender bias was recognized in the target population. To properly cover a wide range of programming expertise, this study was forced to give up on maintaining gender balance at each expertise level. However, several fMRI studies have reported possible gender differences in behavior, cognitive function, and neuroimaging data [97, 98]. The results obtained via this study might be biased by the gender imbalance of the subject population. Future work should investigate whether behavioral and cognitive differences would be found between man and woman programmers. In addition, while the sample size in this study was determined in line with previous expertise studies, ten subjects for each expertise level was not a big population and were insufficient to show statistically significant results between different expertise classes. Therefore, making mention of comparison between novice-middle or middle-expert must be with great caution. Larger samples would be desirable in future replication or follow-up studies.

6. Conclusion and Future work

6.1 Conclusion

This thesis reveals an association between programming expertise and cortical representations of program source code in a programmer's brain. The results demonstrate that functional categories of source code can be decoded from programmer's brain activity and the decoding accuracies on the multiple distributed brain regions in the frontal, parietal, and temporal cortices were significantly correlated with individual behavioral performances. The results additionally suggest that cortical representations of fine functional categories (*subcategory*) on the left SMG and STG might be associated with advanced-level programming expertise. Although research on the neural basis of programming expertise is still in its infancy, the findings extends the existing human expertise literature into the domain of programming by demonstrating that top-level programmers have domain-specific cortical representations.

6.2 Future work

This thesis investigated the neural bases of programming expertise and found the association between individual programming expertise and cortical representations of source code. During the work, I encountered promising future research directions as noted below.

Extension with distributed feature vectors of source code.
The decoding framework specialized for the functional category of source code could be extended by the recent advances of decoding/encoding approaches in combination with distributed feature vectors [99]. Several researchers have demonstrated frameworks to decode arbitrary objects using a set of computational visual features representing categories of target objects [100] and to decode perceptual experiences evoked by natural movies using word-based distributed representations [101]. Other studies have also used word-based distributed representations to systematically map semantic selectivity across the cortex [102,103]. Meanwhile, researchers in the program analysis domain have proposed distributed represen-

tations of source code based on abstract syntax tree (AST) [104,105]. Alon et al., for instance, have presented continuous distributed vectors representing the functionality of source code using AST and path-attention neural network [106]. The combination of recent decoding/encoding approaches and distributed representations of source code may enable us to build a computational model of program comprehension that connects semantic features of source code to programmers' perceptual experiences.

Assessment of information flows in the brain of expert programmers. This thesis demonstrated that individual behavioral performances were significantly correlated with decoding accuracies on the multiple distributed brain regions, located in the frontal, parietal, and temporal cortices. The finding consequently offers a new question about the information flows across these brain regions. In other words, we should ask how the cortical representation of source code is constructed in the network of the spatially distributed brain regions. For answering these questions, combining fMRI data with a high time resolution modality such as magnetoencephalography could be a promising solution [107]. The methods to identify networks of regions with synchronous responses, such as functional or informational connectivity [108,109], might be another potential candidate. Understanding both of activation and information flow in the brain of expert programmers will provide a comprehensive understanding of programming expertise.

Holistic processing of source code underlying programming expertise. A subject with the highest AtCoder rate in the fMRI experiment said that only two seconds were enough to recognize the algorithm implemented in Java source code. It means that the subject could recognize the implemented algorithm based on 6-8 gaze points since a human produces 3-4 gaze fixations on average every second during visual search [110]. Although this is just anecdotal evidence with no valid observational data, it might suggest that top-level players in competitive programming have superiority in visual perception of source code. This perspective is also supported by the finding that individual behavioral performances were significantly correlated with decoding accuracies on the high-level visual cortex.

A potential explanation could be built on holistic processing, which refers to the ability to process complex stimuli as a semantic chunk or a whole unit [22], in a similar way as expertise in other domains [111, 112]. Future work should investigate the association between programming expertise and skills in domain-specific holistic processing.

Biases and differences in program comprehension associated with age, gender, mother tongue.
Cognitive abilities (e.g. working memory capacity [113], number sense [114]) are different between individuals and biased by age, gender, mother tongue [97]. However, this thesis does not cover such potential biases and differences in program comprehension associated with the biological profiles across individuals. The fMRI study presented in this thesis only recruited the subjects who were native Japanese speakers of similar ages ranging from 20 to 24 years old (see Section 3.1 for more details). Future work is expected to clarify the effects of age, gender, mother tongue on program comprehension activities.

Utilization of human programming expertise for artificial agents.
This study, together with previous studies on expert programmers, has demonstrated that biometric data obtained from human programmers can be used as a biomarker that indicates domain-expertise in software development tasks. A next step will be injecting these insights into an autonomous agent to efficiently perform software development tasks. Recent studies used human brain activity to improve the generalization ability of feature representations in machine-learning models [115, 116]. I and colleagues also proposed a conceptual framework of neural autonomous agents based on the visual attention of an expert programmer (see Appendix A for details). Future work should evaluate to what extent utilization of human programming expertise improves performances of artificial agents on software development tasks.

Acknowledgements

I would like to thank all people participating the fMRI experiment. I am grateful to professor Takatomi Kubo and Hideaki Hata for their persistent supports to the entire study. I am also grateful to professor Shinji Nishimoto and Satoshi Nishida for critical advices and great supports on the entire study. This research would never be realized without collaboration with professor Shinji Nishimoto and Satoshi Nishida. I would like to express my gratitude to professor Takashi Ishio, Hidetake Uwano, Takao Nakagawa, and Nishanth Koganti for constructive advices regarding the improvement of the entire study. Discussions with Toshiki Hirao, Takeshi D Itoh, Yuki Ueda, Kiyoka Ikeda, Toyomi Ishida, Keisuke Murai gave me lots of insight and improved the entire work. Finally, I greatly appreciate the valuable feedbacks and immense supports offered by professor Kenichi Matsumoto and Kazushi Ikeda.

References

[1] Paul Luo Li, Andrew J Ko, and Jiamin Zhu. What makes a great software engineer? *Proceedings of the IEEE/ACM 37th International Conference on Software Engineering*, pages 700–710, 2015.

[2] Sebastian Baltes and Stephan Diehl. Towards a theory of software development expertise. *Proceedings of the ACM 26th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 187–200, 2018.

[3] Fredrik Heintz, Linda Mannila, and Tommy Färnqvist. A review of models for introducing computational thinking, computer science and computing in K-12 education. *Proceedings of the IEEE 46th Frontiers in Education conference*, pages 1–9, 2016.

[4] Sofia Papavlasopoulou, Kshitij Sharma, and Michail N Giannakos. How do you feel about learning to code? investigating the effect of children's attitudes towards coding using eye-tracking. *International Journal of Child-Computer Interaction*, 17:50–60, 2018.

[5] Barry W Boehm and Philip N. Papaccio. Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, 14(10):1462–1477, 1988.

[6] Tom DeMarco and Tim Lister. *Peopleware: Productive Projects and Teams, Third Edition*. Addison-Wesley Professional, 2013.

[7] Iris Vessey. Expertise in debugging computer programs: A process analysis. *International Journal of Man-Machine Studies*, 23(5):459–494, 1985.

[8] Jürgen Koenemann and Scott P Robertson. Expert problem solving strategies for program comprehension. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 125–130, 1991.

[9] Vikki Fix, Susan Wiedenbeck, and Jean Scholtz. Mental representations of programs by novices and experts. *Proceedings of the INTERACT'93 and*

CHI'93 conference on Human factors in computing systems, pages 74–79, 1993.

[10] Anneliese Von Mayrhauser and A Marie Vans. Program comprehension during software maintenance and evolution. *Computer*, 28(8):44–55, 1995.

[11] Martha E Crosby, Jean Scholtz, and Susan Wiedenbeck. The roles beacons play in comprehension for novice and expert programmers. *Proceedings of the 14th Workshop of the Psychology of Programming Interest Group*, pages 58–73, 2002.

[12] Hidetake Uwano, Masahide Nakamura, Akito Monden, and Kenichi Matsumoto. Analyzing individual performance of source code review using reviewers' eye movement. *Proceedings of the Eye tracking Research and Applications Symposium*, pages 133–140, 2006.

[13] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. Eye movements in code reading: Relaxing the linear order. *Proceedings of the IEEE 23rd International Conference on Program Comprehension*, pages 255–265, 2015.

[14] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. Understanding understanding source code with functional magnetic resonance imaging. *Proceedings of the IEEE/ACM 36th International Conference on Software Engineering*, pages 378–389, 2014.

[15] Janet Siegmund, Norman Peitek, Chris Parnin, Sven Apel, Johannes Hofmeister, Christian Kästner, Andrew Begel, Anja Bethmann, and André Brechmann. Measuring neural efficiency of program comprehension. *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, pages 140–150, 2017.

[16] Norman Peitek, Janet Siegmund, Sven Apel, Christian Kästner, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brech-

mann. A look into programmers' heads. *IEEE Transactions on Software Engineering*, 46(4):442–462, 2018.

[17] Benjamin Floyd, Tyler Santander, and Westley Weimer. Decoding the representation of code in the brain: An fMRI study of code review and expertise. *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering*, pages 175–186, 2017.

[18] Aline W de Borst, Giancarlo Valente, Iiro P Jääskeläinen, and Pia Tikka. Brain-based decoding of mentally imagined film clips and sounds reveals experience-based information patterns in film professionals. *NeuroImage*, 129:428–438, 2016.

[19] Farah Martens, Jessica Bulthé, Christine van Vliet, and Hans Op de Beeck. Domain-general and domain-specific neural changes underlying visual expertise. *NeuroImage*, 169:80–93, 2018.

[20] Jesse Gomez, Michael Barnett, and Kalanit Grill-Spector. Extensive childhood experience with pokémon suggests eccentricity drives organization of visual cortex. *Nature Human Behaviour*, 3(6):611, 2019.

[21] Merim Bilalić, Thomas Grottenthaler, Thomas Nägele, and Tobias Lindig. The faces in radiological images: fusiform face area supports radiological expertise. *Cerebral Cortex*, 26(3):1004–1014, 2016.

[22] Merim Bilalić. *The Neuroscience of Expertise*. Cambridge Fundamentals of Neuroscience in Psychology. Cambridge University Press, 2017.

[23] Nikolaus Kriegeskorte, Rainer Goebel, and Peter Bandettini. Information-based functional brain mapping. *Proceedings of the National Academy of Sciences*, 103(10):3863–3868, 2006.

[24] K Anders Ericsson, Robert R Hoffman, and Aaron Kozbelt. *The Cambridge handbook of expertise and expert performance*. Cambridge University Press, 2006.

[25] G. R. Bergersen, D. I. K. Sjøberg, and T. Dybå. Construction and validation of an instrument for measuring programming skill. *IEEE Transactions on Software Engineering*, 40(12):1163–1184, 2014.

[26] Janet Siegmund, Christian Kastner, Jorg Liebig, Sven Apel, and Stefan Hanenberg. Measuring and modeling programming experience. *Empirical Software Engineering*, 19(5):1299–1334, 2014.

[27] Minghui Zhou and Audris Mockus. Developer fluency: Achieving true mastery in software projects. pages 137–146, 2010.

[28] Harold Sackman, Warren J Erikson, and E Eugene Grant. Exploratory experimental studies comparing online and offline programing performance. Technical report, SYSTEM DEVELOPMENT CORP SANTA MONICA CA, 1966.

[29] Andy Oram and Greg Wilson. *Making software: What really works, and why we believe it*. O'Reilly, 2010.

[30] Paul A Mabe and Stephen G West. Validity of self-evaluation of ability: A review and meta-analysis. *Journal of applied Psychology*, 67(3):280, 1982.

[31] Philip M Podsakoff and Dennis W Organ. Self-reports in organizational research: Problems and prospects. *Journal of management*, 12(4):531–544, 1986.

[32] Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. A systematic literature review on the usage of eye-tracking in software engineering. *Information and Software Technology*, 67:79–107, 2015.

[33] Unaizah Obaidellah, Mohammed Al Haek, and Peter C-H Cheng. A survey on the usage of eye-tracking in computer programming. *ACM Computing Surveys*, 51(1):1–58, 2018.

[34] Bonita Sharif, Michael Falcone, and Jonathan I Maletic. An eye-tracking study on the role of scan time in finding source code defects. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 381–384. ACM, 2012.

[35] Norman Peitek, Janet Siegmund, and Sven Apel. What drives the reading order of programmers? an eye tracking study. *Proceedings of the IEEE 28th International Conference on Program Comprehension*, 2020.

[36] SeolHwa Lee, Andrew Matteson, Danial Hooshyar, SongHyun Kim, Jae-Bum Jung, GiChun Nam, and HeuiSeok Lim. Comparing programming language comprehension between novice and expert programmers using eeg analysis. *Proceedings of the IEEE 16th International Conference on Bioinformatics and Bioengineering*, pages 350–355, 2016.

[37] Seolhwa Lee, Danial Hooshyar, Hyesung Ji, Kichun Nam, and Heuiseok Lim. Mining biometric data to predict programmer expertise and task difficulty. *Cluster Computing*, 21(1):1097–1107, 2018.

[38] Yoshiharu Ikutani and Hidetake Uwano. Brain activity measurement during program comprehension with NIRS. *Proceedings of the 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 1–6, 2014.

[39] Takao Nakagawa, Yasutaka Kamei, Hidetake Uwano, Akito Monden, Kenichi Matsumoto, and Daniel M German. Quantifying programmers' mental workload during program comprehension based on cerebral blood flow measurement: a controlled experiment. *Proceedings of the IEEE/ACM 36th International Conference on Software Engineering*, pages 448–451, 2014.

[40] Yun-Fei Liu, Judy Kim, Colin Wilson, and Marina Bedny. Computer code comprehension shares neural resources with formal logical inference in the fronto-parietal network. *eLife*, 9:e59340, 2020.

[41] Anna A Ivanova, Shashank Srikant, Yotaro Sueoka, Hope H Kean, Riva Dhamala, Una-May O'Reilly, Marina U Bers, and Evelina Fedorenko. Comprehension of computer code relies primarily on domain-general executive brain regions. *eLife*, 9:e58906, 2020.

[42] Alessandro Guida, Fernand Gobet, Hubert Tardieu, and Serge Nicolas. How chunks, long-term working memory and templates offer a cognitive explana-

tion for neuroimaging data on expertise acquisition: a two-stage framework. *Brain and cognition*, 79(3):221–244, 2012.

[43] Xiaohong Wan, Hironori Nakatani, Kenichi Ueno, Takeshi Asamizuya, Kang Cheng, and Keiji Tanaka. The neural basis of intuitive best next-move generation in board game experts. *Science*, 331(6015):341–346, 2011.

[44] Marie Amalric and Stanislas Dehaene. Origins of the brain networks for advanced mathematics in expert mathematicians. *Proceedings of the National Academy of Sciences*, 113(18):4909–4917, 2016.

[45] Marijke Brants, Jessica Bulthé, Nicky Daniels, Johan Wagemans, and Hans P Op de Beeck. How learning might strengthen existing visual object representations in human object-selective cortex. *NeuroImage*, 127:74–85, 2016.

[46] R. C Oldfield. The assessment and analysis of handedness: The Edinburgh inventory. *Neuropsychologia*, 9:97–113, 1971.

[47] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 2009.

[48] Robert Sedgewick and Kevin Wayne. *Algorithms, Forth Edition*. Addison-Wesley Professional, 2011.

[49] Jonathan W Peirce. Psychopy—psychophysics software in python. *Journal of Neuroscience Methods*, 162:8–13, 2007.

[50] Gerard R Ridgway, Rohani Omar, Sébastien Ourselin, Derek LG Hill, Jason D Warren, and Nick C Fox. Issues with threshold masking in voxel-based morphometry of atrophied brains. *NeuroImage*, 44(1):99–111, 2009.

[51] Martin N Hebart, Kai Görgen, and John-Dylan Haynes. The decoding toolbox (TDT): a versatile software package for multivariate analyses of functional imaging data. *Frontiers in Neuroinformatics*, 8:88, 2015.

[52] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), 2011.

[53] Mingrui Xia, Jinhui Wang, and Yong He. BrainNet Viewer: a network visualization tool for human brain connectomics. *PLoS One*, 8(7):e68910, 2013.

[54] Lukas Snoek, Steven Miletić, and H Steven Scholte. How to control for confounds in decoding analyses of neuroimaging data. *NeuroImage*, 184:741–760, 2019.

[55] Edmund T Rolls, Marc Joliot, and Nathalie Tzourio-Mazoyer. Implementation of a new parcellation of the orbitofrontal cortex in the automated anatomical labeling atlas. *Neuroimage*, 122:1–5, 2015.

[56] James W Tanaka and Marjorie Taylor. Object categories and expertise: Is the basic level in the eye of the beholder? *Cognitive Psychology*, 23(3):457–482, 1991.

[57] Frank Tong. Primary visual cortex and visual awareness. *Nature Reviews Neuroscience*, 4(3):219–229, 2003.

[58] Joao Castelhano, Isabel C Duarte, Carlos Ferreira, Joao Duraes, Henrique Madeira, and Miguel Castelo-Branco. The role of the insula in intuitive expert bug detection in computer code: an fMRI study. *Brain Imaging and Behavior*, 13(3):623–637, 2019.

[59] Norman Peitek, Janet Siegmund, Chris Parnin, Sven Apel, Johannes C. Hofmeister, and André Brechmann. Simultaneous measurement of program comprehension with fMRI and eye tracking: A case study. *Proceedings of the ACM/IEEE 12th International Symposium on Empirical Software Engineering and Measurement*, 2018.

[60] Jean-François Demonet, Francois Chollet, Stuart Ramsay, Dominique Cardebat, Jean-Luc Nespoulous, Richard Wise, André Rascol, and Richard Frackowiak. The anatomy of phonological and semantic processing in normal subjects. *Brain*, 115(6):1753–1768, 1992.

[61] Sharon L Thompson-Schill, Mark D'Esposito, Geoffrey K Aguirre, and Martha J Farah. Role of left inferior prefrontal cortex in retrieval of se-

mantic knowledge: a reevaluation. *Proceedings of the National Academy of Sciences*, 94(26):14792–14797, 1997.

[62] Alan Simmons, Daniel Miller, Justin S Feinstein, Terry E Goldberg, and Martin P Paulus. Left inferior prefrontal cortex activation during a semantic decision-making task predicts the degree of semantic organization. *NeuroImage*, 28:30–38, 2005.

[63] Cathy J Price. A review and synthesis of the first 20 years of PET and fMRI studies of heard speech, spoken language and reading. *NeuroImage*, 62(2):816–847, 2012.

[64] Jennifer M Rodd, Matthew H Davis, and Ingrid S Johnsrude. The neural mechanisms of speech comprehension: fMRI studies of semantic ambiguity. *Cerebral Cortex*, 15(8):1261–1269, 2005.

[65] Brice A Kuhl, Nicole M Dudukovic, Itamar Kahn, and Anthony D Wagner. Decreased demands on cognitive control reveal the neural processing benefits of forgetting. *Nature Neuroscience*, 10(7):908, 2007.

[66] Carin Whitney, Elizabeth Jefferies, and Tilo Kircher. Heterogeneity of the left temporal lobe in semantic representation and control: priming multiple versus single meanings of ambiguous words. *Cerebral Cortex*, 21(4):831–844, 2010.

[67] Susan Y Bookheimer, Thomas A Zeffiro, Teresa Blaxton, William Gaillard, and William Theodore. Regional cerebral blood flow during object naming and word reading. *Human Brain Mapping*, 3(2):93–106, 1995.

[68] Lisa E Philipose, Rebecca F Gottesman, Melissa Newhart, Jonathan T Kleinman, Edward H Herskovits, Mikolaj A Pawlak, Elisabeth B Marsh, Cameron Davis, Jennifer Heidler-Gary, and Argye E Hillis. Neural regions essential for reading and spelling of words and pseudowords. *Annals of Neurology : Official Journal of the American Neurological Association and the Child Neurology Society*, 62(5):481–492, 2007.

[69] Cornelia Stoeckel, Patricia M Gough, Kate E Watkins, and Joseph T Devlin. Supramarginal gyrus involvement in visual word recognition. *Cortex*, 45(9):1091–1096, 2009.

[70] Anthony D Wagner, Benjamin J Shannon, Itamar Kahn, and Randy L Buckner. Parietal lobe contributions to episodic memory retrieval. *Trends in Cognitive Sciences*, 9(9):445–453, 2005.

[71] Kaia L Vilberg and Michael D Rugg. Memory retrieval and the parietal cortex: a review of evidence from a dual-process perspective. *Neuropsychologia*, 46(7):1787–1799, 2008.

[72] Akira R O'Connor, Sanghoon Han, and Ian G Dobbins. The inferior parietal lobule and recognition memory: expectancy violation or successful retrieval? *Journal of Neuroscience*, 30(8):2924–2934, 2010.

[73] Christine E Watson and Anjan Chatterjee. A bilateral frontoparietal network underlies visuospatial analogical reasoning. *Neuroimage*, 59(3):2831–2838, 2012.

[74] Radek Ptak, Armin Schnider, and Julia Fellrath. The dorsal frontoparietal network: a core system for emulated action. *Trends in cognitive sciences*, 21(8):589–599, 2017.

[75] Alex Martin, James V Haxby, Francois M Lalonde, Cheri L Wiggs, and Leslie G Ungerleider. Discrete cortical regions associated with knowledge of color and knowledge of action. *Science*, 270(5233):102–105, 1995.

[76] Michael S Beauchamp, Kathryn E Lee, James V Haxby, and Alex Martin. Fmri responses to video and point-light displays of moving humans and manipulable objects. *Journal of cognitive neuroscience*, 15(7):991–1001, 2003.

[77] Michael S Beauchamp and Alex Martin. Grounding object concepts in perception and action: evidence from fmri studies of tools. *Cortex*, 43(3):461–468, 2007.

[78] Leonardo Chelazzi, Earl K Miller, John Duncan, and Robert Desimone. A neural basis for visual search in inferior temporal cortex. *Nature*, 363(6427):345–347, 1993.

[79] Anna C Nobre, Truett Allison, Gregory McCarthy, et al. Word recognition in the human inferior temporal lobe. *Nature*, 372(6503):260–263, 1994.

[80] Nikolaus Kriegeskorte, Marieke Mur, Douglas A Ruff, Roozbeh Kiani, Jerzy Bodurka, Hossein Esteky, Keiji Tanaka, and Peter A Bandettini. Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, 60(6):1126–1141, 2008.

[81] Karalyn Patterson, Peter J Nestor, and Timothy T Rogers. Where do you know what you know? the representation of semantic knowledge in the human brain. *Nature reviews neuroscience*, 8(12):976–987, 2007.

[82] Jeffrey R Binder, Rutvik H Desai, William W Graves, and Lisa L Conant. Where is the semantic system? a critical review and meta-analysis of 120 functional neuroimaging studies. *Cerebral Cortex*, 19(12):2767–2796, 2009.

[83] Takeo Watanabe, José E Náñez, and Yuka Sasaki. Perceptual learning without perception. *Nature*, 413(6858):844–848, 2001.

[84] Zhiyan Wang, Masako Tamaki, Kazuhisa Shibata, Michael S Worden, Takashi Yamada, Yuka Sasaki, Mitsuo Kawato, and Takeo Watanabe. Visual perceptual learning of a primitive feature in human v1/v2 as a result of unconscious processing, revealed by decoded fmri neurofeedback (decnef). *bioRxiv*, 2020.

[85] Kendrick N Kay and Jason D Yeatman. Bottom-up and top-down computations in word-and face-selective cortex. *Elife*, 6:e22341, 2017.

[86] Emily C Kubota, Sung Jun Joo, Elizabeth Huber, and Jason D Yeatman. Word selectivity in high-level visual cortex and reading skill. *Developmental Cognitive Neuroscience*, 36:100593, 2019.

[87] Maurizio Corbetta, Gaurav Patel, and Gordon L Shulman. The reorienting system of the human brain: from environment to theory of mind. *Neuron*, 58(3):306–324, 2008.

[88] Shruti Japee, Kelsey Holiday, Maureen D Satyshur, Ikuko Mukai, and Leslie G Ungerleider. A role of right middle frontal gyrus in reorienting of attention: a case study. *Frontiers in Systems Neuroscience*, 9:23, 2015.

[89] Merim Bilalić, Andrea Kiesel, Carsten Pohl, Michael Erb, and Wolfgang Grodd. It takes two–skilled recognition of objects engages lateral areas in both hemispheres. *PLoS One*, 6(1):e16202, 2011.

[90] Satoshi Tanaka, Chikashi Michimata, Tatsuro Kaminaga, Manabu Honda, and Norihiro Sadato. Superior digit memory of abacus experts: an event-related functional MRI study. *NeuroReport*, 13(17):2187–2191, 2002.

[91] Takashi Hanakawa, Manabu Honda, Tomohisa Okada, Hidenao Fukuyama, and Hiroshi Shibasaki. Neural correlates underlying mental calculation in abacus experts: a functional magnetic resonance imaging study. *NeuroImage*, 19(2):296–307, 2003.

[92] Satoshi Tanaka, Keiko Seki, Takashi Hanakawa, Madoka Harada, Sho K Sugawara, Norihiro Sadato, Katsumi Watanabe, and Manabu Honda. Abacus in the brain: a longitudinal functional MRI study of a skilled abacus user with a right hemispheric lesion. *Frontiers in Psychology*, 3:315, 2012.

[93] Philippe-Olivier Harvey, Philippe Fossati, Jean-Baptiste Pochon, Richard Levy, Guillaume LeBastard, Stéphane Lehéricy, Jean-François Allilaire, and Bruno Dubois. Cognitive control and brain resources in major depression: an fmri study using the n-back task. *Neuroimage*, 26(3):860–869, 2005.

[94] Caroline J Moore and Cathy J Price. Three distinct ventral occipitotemporal regions for reading and object naming. *NeuroImage*, 10(2):181–192, 1999.

[95] Martha W Burton, Douglas C Noll, and Steven L Small. The anatomy of auditory word processing: individual variability. *Brain and language*, 77(1):119–131, 2001.

[96] Nadezhda Mkrtychian, Evgeny Blagovechtchenski, Diana Kurmakaeva, Daria Gnedykh, Svetlana Kostromina, and Yury Shtyrov. Concrete vs. abstract semantics: from mental representations to functional brain mapping. *Frontiers in human neuroscience*, 13:267, 2019.

[97] Sean P David, Florian Naudet, Jennifer Laude, Joaquim Radua, Paolo Fusar-Poli, Isabella Chu, Marcia L Stefanick, and John PA Ioannidis. Potential reporting bias in neuroimaging studies of sex differences. *Scientific reports*, 8(1):1–8, 2018.

[98] Yu Huang, Kevin Leach, Zohreh Sharafi, Nicholas McKay, Tyler Santander, and Westley Weimer. Biases and differences in code review using medical imaging and eye-tracking: Genders, humans, and machines. *Proceedings of the ACM 28th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020.

[99] Jörn Diedrichsen and Nikolaus Kriegeskorte. Representational models: A common framework for understanding encoding, pattern-component, and representational-similarity analysis. *PLoS computational biology*, 13(4):e1005508, 2017.

[100] Tomoyasu Horikawa and Yukiyasu Kamitani. Generic decoding of seen and imagined objects using hierarchical visual features. *Nature Communications*, 8:15037, 2017.

[101] Satoshi Nishida and Shinji Nishimoto. Decoding naturalistic experiences from human brain activity via distributed representations of words. *NeuroImage*, 180:232–242, 2018.

[102] Alexander G Huth, Wendy A de Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453, 2016.

[103] Francisco Pereira, Bin Lou, Brianna Pritchett, Samuel Ritter, Samuel J Gershman, Nancy Kanwisher, Matthew Botvinick, and Evelina Fedorenko. Toward a universal decoder of linguistic meaning from brain activation. *Nature Communications*, 9(1):963, 2018.

[104] Uri Alon, Omer Levy, and Eran Yahav. code2seq: Generating sequences from structured representations of code. *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[105] Jean Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, Kaixuan Wang, and Xudong Liu. A novel neural source code representation based on abstract syntax tree. *Proceedings of the IEEE/ACM 41th International Conference on Software Engineering*, pages 783–794, 2019.

[106] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning distributed representations of code. *Proceedings of the 46th ACM SIGPLAN Symposium on Principles of Programming Languages*, 3, 2019.

[107] Francesco De Pasquale, Stefania Della Penna, Abraham Z Snyder, Christopher Lewis, Dante Mantini, Laura Marzetti, Paolo Belardinelli, Luca Ciancetta, Vittorio Pizzella, Gian Luca Romani, et al. Temporal dynamics of spontaneous meg activity in brain networks. *Proceedings of the National Academy of Sciences*, 107(13):6040–6045, 2010.

[108] Martijn P Van Den Heuvel and Hilleke E Hulshoff Pol. Exploring the brain network: a review on resting-state fmri functional connectivity. *European neuropsychopharmacology*, 20(8):519–534, 2010.

[109] Marc N Coutanche and Sharon L Thompson-Schill. Informational connectivity: identifying synchronized discriminability of multi-voxel patterns across the brain. *Frontiers in human neuroscience*, 7:15, 2013.

[110] Alasdair DF Clarke, Matthew J Stainer, Benjamin W Tatler, and Amelia R Hunt. The saccadic flow baseline: Accounting for image-independent biases in fixation behavior. *Journal of vision*, 17(11):12–12, 2017.

[111] Harold L Kundel, Calvin F Nodine, Emily F Conant, and Susan P Weinstein. Holistic component of image perception in mammogram interpretation: gaze-tracking study. *Radiology*, 242(2):396–402, 2007.

[112] Merim Bilalić, Robert Langner, Rolf Ulrich, and Wolfgang Grodd. Many faces of expertise: fusiform face area in chess experts and novices. *Journal of Neuroscience*, 31(28):10206–10214, 2011.

[113] Edward K Vogel and Maro G Machizawa. Neural activity predicts individual differences in visual working memory capacity. *Nature*, 428(6984):748–751, 2004.

[114] Justin Halberda, Michèle MM Mazzocco, and Lisa Feigenson. Individual differences in non-verbal number acuity correlate with maths achievement. *Nature*, 455(7213):665–668, 2008.

[115] Dan Schwartz, Mariya Toneva, and Leila Wehbe. Inducing brain-relevant bias in natural language processing models. *Proceedings of the Thirty-third Conference on Neural Information Processing Systems*, pages 14123–14133, 2019.

[116] Satoshi Nishida, Yusuke Nakano, Antoine Blanc, Naoya Maeda, Masataka Kado, and Shinji Nishimoto. Brain-mediated transfer learning of convolutional neural networks. *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 5281–5288, 2020.

[117] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[118] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2891–2897, 2017.

[119] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[120] Nishanth Koganti, Abdul Rahman H. A. G., Yusuke Iwasawa, Kotaro Nakayama, and Yutaka Matsuo. Virtual reality as a user-friendly interface for learning from demonstrations. *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages D310:1–D310:4, 2018.

[121] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Proceedings of the Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[122] Ralf Engbert, Antje Nuthmann, Eike M Richter, and Reinhold Kliegl. Swift: a dynamical model of saccade generation during reading. *Psychological review*, 112(4):777, 2005.

[123] Erik D Reichle, Alexander Pollatsek, and Keith Rayner. E–z reader: A cognitive-control, serial-attention model of eye-movement behavior during reading. *Cognitive Systems Research*, 7(1):4–22, 2006.

[124] Rahul Gupta, Soham Pal, Aditya Kanade, and Shirish Shevade. Deepfix: Fixing common C language errors by deep learning. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1345–1351, 2017.

[125] Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 440–450, 2017.

[126] Hideaki Hata, Emad Shihab, and Graham Neubig. Learning to generate corrective patches using neural machine translation. *arXiv preprint arXiv:1812.07170*, 2018.

[127] Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[128] Paige Rodeghero, Collin McMillan, Paul W McBurney, Nigel Bosch, and Sidney D'Mello. Improving automated source code summarization via an eye-tracking study of programmers. *Proceedings of the IEEE/ACM 36th International Conference on Software Engineering*, pages 390–401, 2014.

[129] Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. How do professional developers comprehend software? *Proceedings of the 34th International Conference on Software Engineering*, pages 255–265, 2012.

[130] Jamie Starke, Chris Luce, and Jonathan Sillito. Searching and skimming: An exploratory study. *Proceedings of the 25th IEEE International Conference on Software Maintenance*, pages 157–166, 2009.

[131] Tianhao Zhang, Zoe McCarthy, Owen Jowl, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–8, 2018.

[132] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813, 2017.

[133] Pannavat Terdchanakul, Hideaki Hata, Phannachitta Passakorn, and Kenichi Matsumoto. Bug or not? bug report classification using n-gram idf. *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*, pages 534–538, 2017.

[134] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[135] Yunzhu Li, Jiaming Song, and Stefano Ermon. InfoGAIL: interpretable imitation learning from visual demonstrations. *Proceedings of Advances in Neural Information Processing Systems*, pages 3812–3822, 2017.

[136] Miltiadis Allamanis, Earl T. Barr, Premkumar Devanbu, and Charles Sutton. A survey of machine learning for big code and naturalness. *ACM Computing Surveys*, 51(4):81:1–81:37, 2018.

[137] W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, 2016.

[138] Kamel Alreshedy, Dhanush Dharmaretnam, Daniel M. German, Venkatesh Srinivasan, and T. Aaron Gulliver. SCC: Automatic Classification of Code Snippets. *Proceedings of the 18th IEEE International Workshop on Source Code Analysis and Manipulation*, pages 203–208, 2018.

[139] Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S. Yu. Improving automatic source code summarization via deep reinforcement learning. *Proceedings of the IEEE 33rd International Conference on Automated Software Engineering*, pages 397–407, 2018.

[140] Siyuan Jiang, Ameer Armaly, and Collin McMillan. Automatically generating commit messages from diffs using neural machine translation. *Proceedings of the IEEE/ACM 32nd International Conference on Automated Software Engineering*, pages 135–146, 2017.

[141] Ricardo Chavarriaga and José del R Millán. Learning from eeg error-related potentials in noninvasive brain-computer interfaces. *IEEE transactions on neural systems and rehabilitation engineering*, 18(4):381–388, 2010.

[142] Marta Kutas and Steven A Hillyard. Reading senseless sentences: Brain potentials reflect semantic incongruity. *Science*, 207(4427):203–205, 1980.

[143] Angela D Friederici and Martin Meyer. The brain knows the difference: Two types of grammatical violations. *Brain research*, 1000(1-2):72–77, 2004.

Appendix

A. Toward Imitating Visual Attention of Expert Programmers

This thesis, together with previous studies on expert programmers, has demonstrated that biometric data obtained from human programmers can be used as a biomarker that indicates domain-expertise in software development tasks. In particular, we have gained a lot of insight in the last three decades by knowing where a programmer is allocating visual attention, which can be inferred from eye movement data [32, 33]. A next step will be injecting these insights into an autonomous agent to efficiently perform software development tasks. As a potential research idea, this appendix presents a conceptual framework of neural autonomous agents based on imitation learning, which enables an agent to mimic the visual attention of an expert via his/her eye movement.

A.1 Introduction

Imitation learning (IL) is an emerging paradigm where autonomous agents learn from human demonstration to perform complex task [117]. IL views a human demonstration as an exemplar of prior knowledge in their working system and leverages a set of human demonstrations to work on tasks whose reward functions are hard to be defined *a priori*. The paradigm has been successfully applied to several applications such as autonomous driving [118] and an initialization for reinforcement leaning in robotics [119]. In such applications, eye movement is a major representation that can bridge human physical demonstrations and training of virtual agents [120].

This appendix presents a conceptual framework of neural autonomous agents based on imitation learning, which enables an agent to mimic the visual attention of an expert via his/her eye movement. In this framework, the agent is represented by a context-based attention model that consists of encoder/decoder network [121]. The framework regard programmers' gaze-fixation as a state-action sequence that represents how a programmer addressed the output that he/she

finally made. Figure 13 shows an example of state-action sequence inferred from gaze-fixation data. The state-action sequences are created for every task period and used to train the autonomous agent to imitate the visual attention of an expert. Historically, several mathematical models have been proposed to describe dynamics of eye movement behavior [122] [123]. Their main goals were to get a biologically plausible model that can well account for real data. In comparison with the models, a primary goal of the IL framework is to maximize agents' performance on a specific task using expert eye movements as a prior knowledge.

Many studies have described how to build an agent which can automatically perform a software development task, e.g. bug fix [124], semantic parsing [125], patch generation [126], and code generation [127]. In most cases they used only feature representations based on textual characteristics, but a few additionally utilized human gaze-fixation data [128]. It has already been demonstrated that programmers use attention strategies to save time for program comprehension and maintenance [129]. For example, expert programmers tend to automatically concentrate their attention onto informative parts of a program [11] and skim only the relevant keywords in source code [130]. Incorporating gaze-fixation data allows autonomous agents to learn attention strategies that are hard to learn solely from textual characteristics. IL-based agents can fully utilize the valuable information sources with less information loss and potentially improve their performances on a variety of software development tasks.

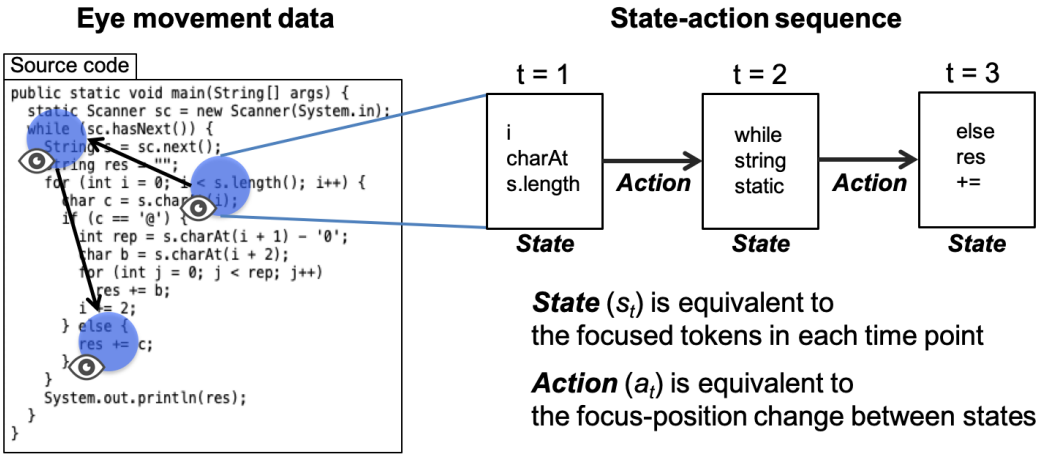

Figure 13. **Programmers' eye movement as a demonstration for a state-action sequence.** The state-action sequence represents how a programmer addressed the output that he/she finally made.

A.2 Proposed Framework

This study proposes IL for training an agent to mimic the visual attention of an expert programmer toward performing several software development tasks. The agent can be trained using the Behavioral Cloning (BC) algorithm which is commonly used in robotics [131] and natural language processing [132]. The code snippet or the environment for the agent can be considered as a sequence of tokens or keywords and the agent needs to focus on a particular subset of tokens that mimics an expert programmers' visual attention. Using demonstration by an expert, the agent can be adapted to perform specific tasks such as bug fixing or algorithm detection which can be considered as an auxiliary task of the agent.

This study formulates a task in IL as a sequence of states and actions. For a software development task, the state (s_t) is a feature representation of the current token being attended to and the action (a_t) is a reference to the next token. The agent performs the task using a *policy* function, commonly referring to the strategy used by the agent. The policy takes as input the current state and should output the desired action. The representations used for the state and action are crucial to ensure good performance of the network. The policy

can also provide task-relevant outputs on attending to the code snippet (o_T). For example, for the task of algorithm classification, the task output can be a discrete label indicating the correct algorithm class.

This study proposes to represent the agents' policy using deep neural networks, capable of learning complex feature mappings, as shown in Fig. 14. As source code is a text sequence, insights from natural language processing can be used to design the agent. The study proposes an agent that consists of two components. The first component is a recurrent neural network (RNN) that can be used to encode the global context of the code snippet. The second component is a task-specific decoder model using an RNN that takes as input the encoded context vector at a particular token and outputs the next token to attend as the action. The task decoder also provides a task-relevant output such as a discrete label indicating algorithm class or a token index in the code snippet indicating the bug location. Typically code snippets can be of variable lengths and requires attention over variable length sequences. To address this problem of variable size input, this study proposes the use of Pointer Networks [121], which is a hard-attention mechanism capable of handling sequences of variable lengths.

In BC, the agent is trained using two loss functions. A primary loss function is used to train the agent so as to mimic the visual attention of an expert for a particular code snippet. Depending on the task, auxiliary loss functions can be included to perform the specific task:

$$\mathcal{L}_{\text{BC}}(a_t, \hat{a}_t) = w_{\text{att}}\mathcal{L}_{\text{att}}(a_t, \hat{a}_t) + w_{aux}\mathcal{L}_{\text{aux}} \tag{1}$$

where $a_t, \hat{a}_t$ are the predicted action and expert action respectively. The attention loss function $\mathcal{L}_{\text{att}}$ can be a cross entropy loss with one-hot vectors representing the true and predicted tokens for attention. For the auxiliary loss function, $\mathcal{L}_{\text{aux}}$, cross-entropy loss can be used to match the algorithm class or a particular token in the code snippet sequence. The weights assigned to each loss function ($w_{\text{att}}, w_{\text{aux}}$) are assigned depending on the relative importance of the attention mechanism and task-specific output.

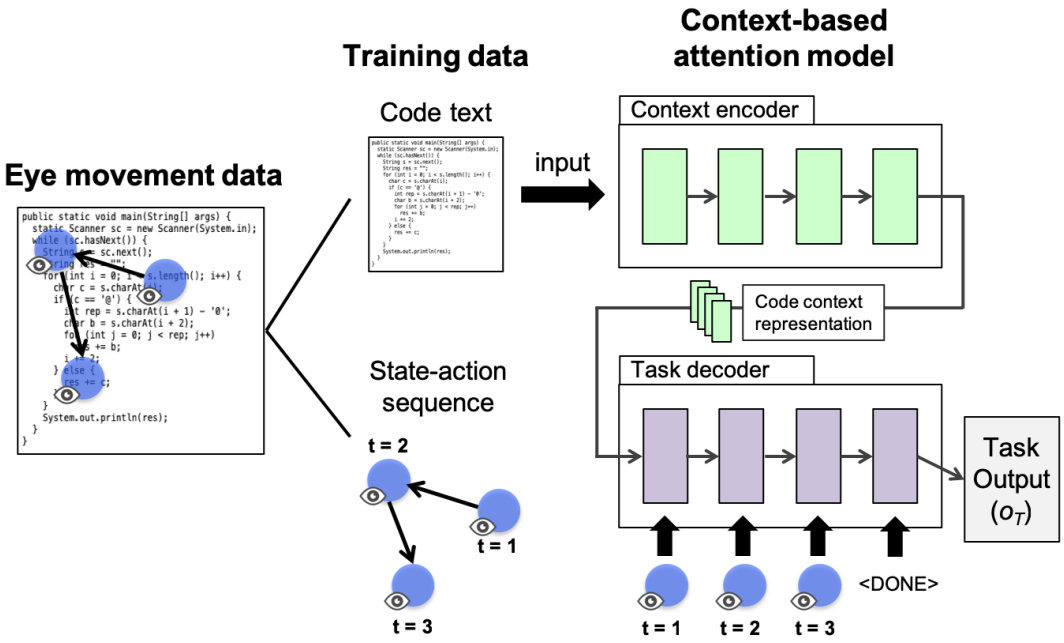

Figure 14. **Overview of imitation learning agent that relies on expert's gaze data to perform source code comprehension tasks.** This study proposes an agent that consists of two recurrent neural networks that one encodes global context of the code snippet and the other decodes the context vector at a particular token and outputs the next token to attend as the action.

A.3 Challenges

The prospective challenges to implement an IL-based autonomous agent specialized for software development task were discussed below.

Feature Representations

A common challenge with effective implementation of IL is the state correspondence problem where the expert and agent might have different feature representations. The feature representation should sufficiently capture the functional meaning of the code without adding any redundant information not considered by the expert. Several state representations can be considered for the analysis. Source code unlike natural language is structured and so binary feature repre-

sentations such as Bag-of-Words (BoW) and bag-of-n-grams [133] could prove effective. Word embeddings such as code2vec [106] can be considered to handle long term dependencies common for source code. For the action representation, considering spatial coordinates in the image space could be ineffective as this could vary depending on irrelevant factors such as code formatting. An effective action could be location to a particular token index or to a group of tokens that represent a functional unit in the code.

Data Efficiency for Imitation Learning

One of the challenges of using BC is that it requires a large number of demonstrations to avoid covariate shift where the state space encountered by the agent could be drastically different from expert demonstrations. However, collecting a large dataset involving expert eye-gaze information could be prohibitive. To improve the data efficiency of IL, this study considered several strategies. A typical gaze-fixation of the human is usually on a group of tokens. This uncertainty in gaze fixation can be exploited to perform data-augmentation where a single trial can result in multiple state-action sequences by assigning different weights to the group of tokens. An alternate approach is to learn a distribution over the expert demonstrations rather than to just mimic the actions. This can be performed by using Generative Adversarial Imitation Learning (GAIL) [134] which is based on generative adversarial training where a generator neural network is trained to model the distribution of expert demonstrations. This has been shown to provide superior performance over traditional IL in several application domains. The data-efficiency can be further improved by learning from multiple experts. However, different experts could be using different strategies and it is important to disambiguate between such latent factors. This can be performed by using techniques such as InfoGAIL [135] which can learn to disambiguate between multiple experts through a learned discrete latent representation.

Possible Software Development Tasks

The intersection of machine learning and software engineering is an emerging hot research topic [136]. The IL-based framework will open up a new way of building intelligent systems/agents.

Fault localization. Fault localization is the act of identifying the locations of faults in a program. To support this time-consuming task, many machine-learning based approaches have been proposed [137]. By designing a token index in the code as the task-relevant output, this study can propose new systems incorporating the art of identifying bugs by experts. Preparing experimental settings for collecting experts' eye-movements is also an important challenge. With similar approaches, this study may target identifying performance issues, vulnerabilities, etc.

Classification. Classification is a fundamental machine learning technique and can be applied to specific applications, such as language detection [138], algorithm identification. Task-relevant outputs will be discrete labels indicating classes.

Description generation and code generation. Generating the description of code or code changes based on deep neural networks is a promising active research topic, such as code summarization [139], commit message generation [140], etc. Generating (part of) code with deep neural networks [124–127] is another hot topic. Although the current concept of using Pointer Networks does not work for generating sequences, the idea of incorporating experts' visual attention is also interesting for these tasks. Designing an appropriate imitation learning framework is a desired future challenge.

Extension with Electroencephalography as Auxiliary Input

It may be possible to complement the system above with electroencephalography (EEG) to reflect the human cognition other than visual attention [141]. For example, event-related potentials (ERPs) including error-related potentials reflect events where a human subject feels wrong or strange. It is known that ERPs can reflect semantic incongruity grammatical violoations in language processing [142, 143]. ERPs could be informative for programming comprehension, especially for bug fix. To integrate gaze data and EEG data, multiple ways can be considered. One is independent subsystem from the imitation learning, and the other is modification of the architecture to include the EEG data as an auxiliary input. Especially, the latter will be a tough challenge, but it will also be an attractive topic not only for software engineering but also for artificial intelligence and modelling of human cognition.

A.4 Concluding remarks

A baby learns numerous things from the demonstration by parents without any lingual explanations because demonstrations can represent more than language descriptions. So far, researchers investigated eye movements of programmers and typically converted them into human-understandable numbers and descriptions, such as fixations and saccades. However, this conversion has caused considerable information loss. This study proposes neural attention models trained using expert programmers' eye movements as a means of implicit learning without information loss. The study presented a plausible framework to achieve this goal using IL and discussed several challenges that will need to be addressed to make the framework practical. IL-based agents can fully utilize the valuable information sources with less information loss and potentially improve their performances on a variety of software development tasks.

Publication list

The early version of the work in this thesis was published as listed below.

- Yoshiharu Ikutani, Takatomi Kubo, Satoshi Nishida, Hideaki Hata, Kenichi Matsumoto, Kazushi Ikeda, and Shinji Nishimoto. Expert programmers have fine-tuned cortical representations of source code. *eNeuro*, volume 8, Issue 1, 2020.

- Yoshiharu Ikutani, Nishanth Koganti, Hideaki Hata, Takatomi Kubo, and Kenichi Matsumoto. Toward imitating visual attention of experts in software development tasks. *Proceedings of the IEEE/ACM 6th International Workshop on Eye Movements in Programming*, 2019.