JOURNAL OF POLITICAL STABILITY ARCHIVE

Online ISSN: 3006-5879 Print ISSN: 3006-5860
DOI: https://doi.org/10.63468/jpsa.3.3.110
Vol. 3 No. 3 (2025)
https://journalpsa.com.pk/index.php/JPSA/about

Recognized by: Higher Education Commission (HEC), Government of Pakistan

# From Brain to Binary: The Neuroscience Behind Writing and Understanding Code

Riaz Ahmed
Bachelors Student at National Research Tomsk State University, Tomsk Oblast, Russian Federation
riazahmed5231@gmail.com

ABSTRACT

Although in the modern, technology-driven world, programming is significant, the psychological and biological mechanisms of coding have been comparatively understudied. This paper is a discussion of the interface between neuroscience and computer programming with the topic of cognitive processes of program understanding, code writing and debugging. We also look at the relationship between these processes and natural language and mathematical reasoning, and find common and different neural substrates in these processes. With the support of neuroimaging, cognitive psychology, and computational modelling, we suggest a clear and systematic way of studying the knowledge of programming. Combining the results of these areas, we introduce a complete overview of the neural and cognitive mechanisms that underlie programming, the involvement of general executive functions, i.e., working memory, attention, and cognitive control. Our review indicates that the degree of activation of these processes varies with the development of programming skills, which is an indicator of neuroplasticity and higher efficiency of neural processing. We also determine significant gaps in existing research and provide future research directions, especially in the domains of learning optimisation, accessibility, and design of human-AI collaborative programming environments. This combined method focuses on the interaction of general cognitive systems and specialised programming skills, which gives information on learning, interface design and improvement of programming skills.

**Keywords**: Neuroscience of Programming, Code Comprehension, Program Writing, Debugging, Multiple-Demand Networks, Cognitive Stress, Neurocompatibility, Programming Skills, Human-AI Interfaces.

## INTRODUCTION

Developing and interpreting software has become an important intellectual undertaking of the 21st century. Since the advent of technology in nearly all walks of human life, such as communication, commerce, health as well and education, there has been a huge demand for individuals capable of learning programming languages. Although much has been done in terms of research on the cognitive and neural basis of other knowledge or skills like language, mathematics or music, the same cannot be said about coding. This paper aims to disclose the technicalities of programming through an analysis of the psychological processes, as well as neurotic mechanisms of the process, the brain planning and controlling, and executing the operations that occur during coding.

Computer programming is not technical alone, it is more than that, it is a human-performed intellectual job involving a combination of reasoning, imagination, individual competencies and thinking capabilities. Any piece of code, which is written by a programmer, triggers various mental processes. They need to bear in mind the right list of symbols, deduce, have several elements simultaneously in active thinking (working memory), and simulate the reaction of a conceptual machine. The skill of programming is at the crossroads of various fields, such as a wide range of language skills, as well as mathematics and higher-order thinking, unlike traditional types of linguistic literacy, which have attracted considerable attention both among linguists and psychologists.

During the last several decades, new neuroscientific studies have emerged on the peculiarities of mental challenges that computer programming presents. Researchers have found that with the techniques of functional magnetic resonance imaging (fMRI) and other neuroimaging techniques, programming involves a wide network of brain regions. Besides having an overlap with regions involved in natural language comprehension (including that of the Broca area), programming also involves the multiple-demand network (MDN), which deals with higher-level cognitive control and problem-solving. The question is interesting: is programming a language like exercise, a mathematical exercise or something different?

The learning to code process assists in visualising the remarkable potential of the change in the form and functioning of the brain. The problems of the novices in understanding and excessive dependence on syntax prefer primary linguistic ways of thinking. The professionals optimise neural processing, and thus, abstraction and reasoning become central. Not only does the novice-expert transition help to shed light on the brain's way of remembering new symbolic systems, but it also compares it to the acquisition of literacy or numeracy in humans.

The implications of the neuroscience involved in coding for education researchers include the ability to direct these processes to be consistent with the natural learning processes of the brain, thus simplifying the processes of coding and making them less stressful. To cognitive scientists, studying programming offers them an excellent chance to investigate the limits of human symbolic thinking. In the case of technologists, by understanding the neural programming process, it will

be possible to inspire the creation of AI systems that will be more like humans in their thinking.

The following paper will discuss programming as a cognitive process, neuroscientific evidence on the way that the brain interacts with the code, compare the aspects of programming with the way the human brain perceives a natural language, and explore how the development of skills changes the neural structure. Lastly, it will discuss the implications of this realisation to education, accessibility, and the future of human-computer interaction. Following the route of the brain towards the binary, we are seeking to explain how ideas are converted into code, and how code is the basis of our digital age.

## Programming as a Cognitive Task

Programming is not merely a technical exercise which entails learning syntax and code. It is rather a complicated mental activity that, in most cases, involves numerous mental faculties at the same time. The overall structure of software, the logic of a particular function, and the actions of data structures are simply a few components of a programme, which can be said to be interrelated and are therefore considered by the programmers in writing or in understanding code. To be able to organise codes algorithmically and at the same time to have variable relationships, one requires a highly active working memory.

Also, programming requires executive functions that deal with planning and putting in order some tasks, especially debugging, which is one of the most mentally demanding tasks in the software development process. Good attentional control is required in the case of programming since a small syntactic error can cause severe logical errors. Coding involves more focused attention to the exact form of code and the logical development that it entails, compared to more automatic cognitive activities.

More so, problem-solving is the essence of programming. The developers should always be able to think critically about the constraints and unforeseen events, use the knowledge that they have to address the emerging issues, and create new solutions. This type of problem-solving uses analogical reasoning and abstract concepts that are applied in other fields, such as logic or mathematics. The greatest problem in programming is the cognitive load posed by conceptual images and stratified design. In comprehending a kind of function, say it might be essential to consider not only its explicit syntax and activities but also its preceding role within an algorithm, its connection to other elements, and the expected results for different inputs. The cognitive load theory highlights that the capacity of working memory of the brain is limited. Inexperienced programmers are often overwhelmed by unknown syntax and problems with combining the various levels of abstraction.

Proficient programmers, on the other hand, develop mental models that can assist them in organising data effectively. The expert can concentrate on advanced thinking and systems design since the approaches lessen the mental strain of understanding discrete parts. In particular, debugging also evidences the level of knowledge in programming. It includes the identification of the point of the error

committed, developing and proving the theories, assessing the results, and changing strategy based on new information. This is consistent with the scientific method and requires one to have both logical thinking ability and metacognition, or the capacity to monitor and regulate the thought process that one carries out. Neuroscientific studies reveal that brain activity involving areas of attention in hypothesis testing and strategic control is performed during debugging activities. The process of debugging is not only a technical procedure, but a very cognitive process as well, because the programmer must supervise an ongoing process of prediction and confirmation. Programming mostly involves a large variety of cognitive skills such as abstract thinking, memory, of management of attention and logic. Coding seems to apply domain-general mental functions as opposed to operating as part of a single cognitive domain. This is possible through a system network of related systems, which enable the programmer to convert the human intentions into machine-comprehensible logic.

## Neuroimaging of Programmers

The neuroimaging studies have given significant insights into the brain functioning during programming, displaying that no one programming centre is activated due to the coding, but instead it involves a distributed network of regions functioning together. Functional magnetic resonance imaging (fMRI) is found regularly to have strong activation in the prefrontal cortex that regulates executive functions such as planning, decision-making, and cognitive control. These areas are specifically activated in tasks involving complicated problem-solving, debugging or arranging multi-step algorithms. In the comprehension of code, the inferior frontal gyrus has long been regarded as part of natural language processing, also displays prominent activation, meaning that the neural processes involved in linguistic and programming tasks are similar. It is also used intensively in programming, which is not only a task in a language and not a domain-specific one, but a domain-general system known as the multiple-demand network (MDN). The involvement of parietal parts, including the intraparietal sulcus, in the process of doing tasks that require algorithmic thinking and manipulation of symbols highlights the relationship between mathematical thinking and programming. The temporal lobe aids in decoding programming structures and problem-solving methods through the activation of the conceptual knowledge retrieval and semantic memory. The fMRI results are also supported by further exposure of oscillatory patterns associated with working memory load, attention allocation, and error detection during coding EEG studies. A multimodal account of programmer cognition is achieved through the eye-tracking and neuroimaging studies that show how the brain can divide attention on code. All these studies demonstrate that the uniquely hybrid cognitive demands of programming are satisfied by a coherent network that is a composite of linguistic, mathematical and executive operations.

## Finding bugs and understanding code: cognitive psychology in software development

Cognitive psychology regards two of the most mentally challenging parts of programming to be code comprehension and debugging. Determining the existence of anomalies, writing theories about their origins, testing possible remedies and measuring outcomes are all processes that are cognitive in debugging. These tasks require both metacognitive and analytical skills since programmers have to always strategise and adjust their thoughts in order to monitor them. Code comprehension also requires high working memory, as well as selective attention. The programmers must be aware of how functions interact with each other to follow multiple variables simultaneously and how algorithmic processes will result. Eye-tracking research suggests that proficient programmers develop successful scanning patterns that help them to quickly identify significant pieces of code, whereas beginners often get stuck on a single code line without perceiving larger-scale dependencies. Also, cognitive psychology is interested in the way expertise is developed through schema formation. Mental models of programme structure and interaction of functions are developed by experts and are normally explicit among novices. These schemas reduce cognitive load, hence allowing fast error detection, predictive cognition and higher-order thought. Attention, problem-solving, and metacognition are all variants of working memory and all of them are involved in debugging and comprehension, which are not merely technical skills.

## Comparisons Between Novices and Experts in Software Engineering

The difference between expert and novice programmers lies in the fact that they differ not only in the way their brains and minds work. Natural learners tend to struggle with the process of abstract thinking or system-wide comprehension, as they primarily focus on linguistic parts of the brain and memorisation of explicit syntax. The problem-solving approach they take is often linear, based on immediate errors and not the whole program. As language-dependent processing fades away in favour of domain-general executive systems, the analogy-based reasoning and schema composition networks of problem-solving begin to be intertwined in intermediate learners. Professional programmers, on the other hand, exhibit successful multiple-demand network activation that gives them the ability to handle complex dependencies that give priority to abstraction and take strategic plans. In neuroimaging research specialists, it is found to have more prefrontal and parietal network activity and reduced reliance on language-specific areas. The behaviour of experts exhibits selective allocation of attention in effective code scanning and predictive error detection, and also novices have a higher concentration of their cognitive resources in the simple comprehension and syntax checking. These changes in the brain in neuroplasticity show that exposure to programming and practice changes neural circuits towards better working memory integration, executive function competence, and higher-order problem-solving capacity. Internal structures are developed by expert programmers in their attempts to handle cognitive load efficiently so that they can focus on the optimisation of abstraction

and strategic reasoning instead of rote memorisation based on other cognitive ergonomics studies.

## Computational neuroscience and Cognitive Ergonomics.

Computational neuroscience models can be used to understand how programming entails the representation, processing and integration of symbolic information. To give predictive information on the cognitive bottlenecks in completion of complex coding tasks, neural network simulations can reveal the dynamics of working memory, hierarchical reasoning and allocation of attention. These models focus on the fact that the working memory capacity has limitations that distract vulnerability and that multi-step problem-solving has certain effects on the cognitive load of which are particularly relevant in the case of inexperienced programmers. These understandings are improved by focusing on the development of programming tools and environments that do not exceed the cognitive limits of the human being, cognitive ergonomics. The minimisation of unnecessary cognitive load through the use of visualisation tools, real-time debugging feedback, and syntax highlighting assists in understanding and solving problems in the integrated development environments (IDEs). Memory and error management are also considered in the ergonomically informed designs that assist programmers to remain focused and easily navigate through complex code bases. Multimodal integration can contribute to effective programming through the visual motor and memory systems in developing adaptive AI-assisted coding tools. The communication point between cognitive ergonomics and computational neuroscience provides a set of concepts in understanding programming as a technological and cognitive enterprise. It brings out the interaction between brain potentials, task needs and environmental facilitation and offers ways to enhance the availability of productivity, access to learning and design of the tools. It is a combination strategy that could be used to inspire the creation of AI, which is more closely related to human thinking patterns, to optimise programming spaces and shape educational interventions.

## Evidence from Neuroscience

Neuroscience research on programming has aimed at discovering the neural pathways that are involved in the process of understanding and writing code. Neuroimaging research that uncovers code activation in brain regions that are associated with executive functions, logical reasoning, and certain aspects of language understanding is of the view that code activation in the brain can involve a combination of different systems. When it is necessary to code or troubleshoot, the prefrontal cortex is involved in planning, making decisions and solving problems. Natural language processing has been associated with the inferior frontal gyrus, which is highly activated in understanding codes that hint that programming and language processing tasks are likely to use the same neural mechanisms. The parietal lobes, on the other hand, facilitate symbolic manipulation and spatial reasoning, both of which are usually important in unravelling data structure and algorithmic processes. The temporal lobe is very active, especially in the context of semantic memory and the retrieval of conceptual information, which helps in the

understanding of the concept of programming. Research on functional MRI has been especially educative. Ivanova and co-authors demonstrated that even the neural bases of comprehending code and natural language are intertwined, programming activates another network in which logical reasoning takes precedence over linguistic syntax only (2020). Even more studies prove that programming tasks result in stronger activation in the multiple-demand network, a domain-general system that controls cognitive control in a range of tasks that support this differentiation. This indicates that though programming involves the inclusion of aspects of language, math, and logic, it also adopts a hybrid neural structure that can no longer be reduced to any of the aforementioned fields. Brain circuits that are related to formal reasoning and mathematics seem to be more similar to programming. The intraparietal sulcus region that is associated with numerical reasoning is also activated by doing algorithmic problem-solving. This implies that at the interface between symbolic reasoning structures, mathematical abstraction and linguistic understanding interrelate. It is due to this mix of programming that makes neuroscientific research in programming a very fertile domain.

**Code vs. Natural Language Processing.**

Programming languages' organised grammar and vocabulary can provide them with a superficial similarity to the natural languages their cognitive requirements are fundamentally different. Due to the ambiguity and flexibility of natural language, contextual meaning and semantic unification are imperative. Programming languages, on the other hand, have both rigid syntactic and semantic rules to minimise ambiguity and raise the level of cognitive specificity needed to be understood. The code comprehension and natural language processing involve the networks that are involved in preserving logical consistency and high-level reasoning and semantic understanding, respectively, based on neuroimaging evidence. Since programming is dualistic, existing in both the language and mathematical realms, it has been imagined by a number of scholars to exist in the interstitial space between language and logic. As an illustration, the area of Boca is involved in both language and coding processes, but constant programming requires an increased dependence on the multiple-demand network. This would mean that rules and logical structure are more relevant to understanding programming than semantic knowledge. It seems to have a different activation of the brain when reading and writing code. To learn code, it is necessary to unite syntax and semantics into a single and coherent concept, just as it is in the case of learning text. Coding, on the other hand, is similar to composition and is characterised by the abstraction of strategies and the transformation of abstract concepts into the form of organised instructions. Despite being grounded on a similar symbolic background, neuroimaging studies indicate varying activity when people are generating new code or relying on the information stored in the code when supporting the notion that these two processes are brain distinct.

## Knowledge and Experience in Coding

The ability to write a programme is the manifestation of the amazing neuroplasticity of the brain. New programmers may fail to memorise syntax and simple rules since they are dependent on the parts of the brain associated with understanding language. As the learners accumulate experience using the language processing less and relying more on their neural strategies of abstraction and executive control, they eventually develop more efficient neural strategies. Coding proficiency is reflected in the capacity to minimise unnecessary mental load and engage only in problem-solving, in this case, by the efficient activation of the multiple-demand network, as evidenced by the proficient computer programmer. This form of development has various stages. The abstract concepts are not always easy to master by the students during the beginner stage, when they are mostly dependent on memorisation and language comparisons. Their reasoning becomes increasingly assimilated with the formation of schema analogies and problem-solving networks towards intermediate stages. The expert level is a process with which one applies efficient abstraction and finds patterns in which errors are revealed and removed nearly automatically. This is not unlike other areas of knowledge, like music or mathematics, where practice in a regular and routine way alters neural pathways to enhance specific types of reasoning. These results have serious educational implications. Teaching methods need to be different in case the programme skills indicate the progressive restructuring of brain resources. The assistance to beginners focusing on syntax and language similarities can be helpful, whereas the obstacles forcing the learners to greater levels of abstraction are required for advanced learners. An expanded group of students will be able to be educated to code more easily and efficiently within learning environments that accommodate these levels of ability.

## Implications for Education, Technology, and Artificial Intelligence

Knowledge of neuroscience underlying programming presents opportunities in various spheres, such as access to education and technological design. Understanding the way in which the brain decodes programming can be used to guide educators to develop programmes that consider the strengths and weaknesses of students in their cognitive abilities. By means of visual aids, interactive simulation, and a variety of instructional strategies, students can concentrate on their abstract reasoning abilities by decreasing the cognitive load. In the future, neural signs of challenge would be monitored by customised learning platforms, which would consequently alter teaching to accommodate the individual mental condition of each student. These lessons can also increase accessibility. Persons with intellectual disabilities, like autism, dyslexia or ADHD, might find it difficult to learn programming. The cognitive understanding of neuroscience in relation to programming could guide the design of special resources and teaching techniques which consider these differences. Teachers can promote inclusivity in a field that is rapidly gaining relevance in contemporary life by combining programming education and the various methods that brains process information. Another field of

convergence between neuroscience and programming is technological advancement. The design of development tools and coding environments that minimise unnecessary mental effort and improve understanding can be done by applying cognitive principles. Emerging technology like the brain-computer interface has become an indicator of the potential of having adaptive programming environments which can react in real time to the cognitive state of a user. To enhance the efficiency of the coding process and to reduce the number of errors made by editors, one may think of adding contextual support and improving the examples of cognitive overload. Ultimately, learning the process by which the brain interprets programming can be used to guide the artificial intelligence programmes. The common, non-static, symbolic reasoning that human programmers make use of is usually something that modern AI systems cannot comprehend, even though they are adept at discerning statistical patterns. Neural coding mechanisms of code understanding can be investigated so as to build AI systems that have a better ability to reason about algorithms and logical forms, just like humans.

### Future Directions

Although recent studies have revealed that there are a lot of things about programming neuroscience, several other things should be investigated. Future research may explore how the coding instruction at various stages of life development affects brain development, especially in children compared to adults. The interaction between cognition and cultural environment could be highlighted with the help of cross-cultural research that could also prove the stimulation of various neural pathways by different programming techniques. Similar to the work done on musicians or long-term research of seasoned programmers by mathematicians, could reveal structural variations within the brain that could be caused by years of practice. Another potential direction is multimodal integration. Programming does not only involve thinking abstractly but also involves the use of motor systems to feed text memory systems so as to internalise ideas and structures and visually analyse code structures. The knowledge of how these various systems work together to facilitate programming can perhaps give a better insight into the underlying cognitive system. New instructional resources and settings that more closely correspond to the inherent inclinations of the brain can also be created based on the results of this research.

## METHODOLOGY

To study programming as a complicated cognitive and neural process, this paper will combine results of cognitive psychology, neuroscience, computational modelling and cognitive ergonomics through a narrative review method with aspects of theoretical synthesis. It is the narrative approach that makes the synthesis of different research methodologies, the identification of general trends, and the mentioning of gaps in the existing knowledge of programming cognition possible in comparison to empirical studies, which introduce original data. Since the research on programming is multidisciplinary, involving experimental studies that examine

neuroimaging outcomes, computational models and educational literature, the review largely follows a theoretical synthesis model. Inclusion of studies was done based on clearly identified criteria to ensure rigour and relevance: foundational studies published before 2000 were included as they formulate useful theoretical knowledge, whereas literature published between 2000 and 2025 received first preference to ensure the inclusion of recent developments. We used databases, such as PubMed, Web of Science, Scopus, IEEE Xplore and Google Scholar to query the keywords such as programming cognition, code comprehension, debugging, neuroimaging of programmers, cognitive load, programming expertise, multiple-demand network, computational neuroscience and cognitive ergonomics. More references were found by citing when important articles were used. Research was not considered to be related to human cognition or written in English as peer-reviewed research that focused on neural mechanisms, cognitive processes, and behaviour of human programming. The selected evidence was analysed with the help of a narrative synthesis method by categorising studies into a domain that provides a shared set of cognitive and neural mechanisms that compare various methodological approaches and combine the findings of behavioural, neural, and computational studies. Along with the identification of common themes in the development of working memory, attention, problem-solving expertise and the involvement of neural networks in this process, inconsistency and gaps in the literature were also identified in this analysis. Ultimately, the information was synthesised to investigate the implications of software development processes, accessibility learning and AI-assisted programming aids. According to the standards dictated by the HEC-recognised journals, this methodology does increase the rigour, transparency and reproducibility of the papers by making apparent the type of review selection criterion and method of analysis.

## RESULTS

The review of the literature proves that programming is a cognitively demanding activity that involves the engagement of more than the traditional language processing systems. The neuroimaging studies, in particular, functional magnetic resonance imaging (fMRI) and electroencephalography (EEG), typically demonstrate strong activation in the multiple-demand network (MDN) comprising of lateral prefrontal cortex and posterior parietal regions. These parts are critical to the working memory executive control task switching and problem-solving- all of which are required to code, debug and comprehend code. The highly visible involvement of the MDNs points to the reliance of programming on domain-general processes of thought that distinguish it among language-only or math-only tasks. There are certain similarities between natural language processing and programming, especially for inexperienced programmers. In the regions of code comprehension that are generally associated with the language and syntax, such as the inferior frontal gyrus and Broca's area, moderate activation occurs, which indicates a reliance on verbalised rules and memorised syntax. The higher temporal gyrus, which is

involved in semantic integration, is also activated when interpreting code elements that replicate a sequence of natural language. Although programming and reading or writing have some facade similarities, their underlying neural processes extend past the language network, as can be seen through the fact that the activation is often less strong and inconsistent than compared to activities that are exclusively linguistic. The use of programming, in contrast, activates areas of logic and mathematics very much. The angular gyrus intraparietal sulcus, and the dorsolateral prefrontal cortex are used when solving algorithmic problems and tasks with much abstraction. As demonstrated by Ivanova and others. (2020), coding attracts a network which overlays complex reasoning based on abstraction, rule-based logic and symbolic manipulation in a manner that mimics the formal mathematical thought. The MDN, as per functional connecting studies, combines a variety of cognitive streams to achieve effective problem-solving in the course of complex coding by communicating with language-related and numerical reasoning areas. The brain activity of a beginner and a professional programmer is different. The language-related areas that are more pronounced in beginners are the left inferior frontal gyrus and Broca's area, which implies the reliance on verbal strategies and syntax memorisation. Moreover, during the processing or debugging of multiple-line code, they also carry a larger working memory load and prefrontal activation. Expert programmers, on the other hand, demonstrate reduced cognitive load and schema-based reasoning by more effective MDN and parietal regions activation. Experience and neuroplasticity are two factors that experts underscore in the development of programming skills by relying on abstract mental models to make predictions about how the programme would behave and to spot mistakes early on. Comparisons of studies provide key differences between language-like and math-like programming processing. Tasks involving the reading of variable naming comments or syntactic order involve the recruitment of language networks that are moderately recruited, and mathematical and executive networks are always recruited by algorithmic reasoning, recursion, and abstraction. The most explanation-homogeneous finding in all the studies is MDN engagement that underscores the role of executive function, attentional control, as well as cognitive flexibility as being paramount. The mathematics reasoning areas provide essential assistance to abstraction and logical problem solving, as compared to other auxiliary language areas. Though the expertise type of tasks of participants and the programming language used can all influence the regional activation, the general distribution of the MDN and the parietal regions as predominant and the language regions as secondary is robust. Table 1, which compares the activation during programming language and math tasks, gives an overview of the neural areas. This shows a small area of language recruitment among novices with much overlap with areas of numerical reasoning, and strong parietal and MDN involvement in programming. The programming cognition hybrid feature of linguistic, logical, and executive network intersection is graphically representable in a corresponding figure.

**Table 1: Neural Regions Activated During Programming Compared with Language and Math Tasks**

| Brain Region | Programming | Natural Language | Mathematical/Logical Reasoning | Notes |
|---|---|---|---|---|
| Lateral Prefrontal Cortex (DLPFC) | ✓✓ Strong | ○ Moderate | ✓✓ Strong | Executive control, working memory, task switching |
| Inferior Frontal Gyrus (Broca's Area) | ○ Moderate | ✓✓ Strong | ○ Weak | Syntax processing; more active in novices |
| Superior Temporal Gyrus | ○ Moderate | ✓✓ Strong | ○ Weak | Semantic integration: minor activation in code comprehension |
| Posterior Parietal Cortex / Intraparietal Sulcus | ✓✓ Strong | ○ Weak | ✓✓ Strong | Symbolic manipulation, numerical reasoning, algorithmic problem solving |
| Angular Gyrus | ✓✓ Moderate | ○ Weak | ✓✓ Moderate | Supports abstraction and logical reasoning |
| Multiple-Demand Network (MDN) | ✓✓ Strong | ○ Weak | ✓✓ Strong | Integrates executive control, attention, and problem-solving |
| Temporal Lobe (Semantic Memory) | ○ Moderate | ✓✓ Strong | ○ Weak | Conceptual knowledge retrieval; used for variable/function understanding |
| Anterior Cingulate Cortex (ACC) | ✓✓ Moderate | ○ Weak | ✓✓ Moderate | Error monitoring, conflict detection during debugging |

**Legend:** ✓✓ Strong activation | ○ Moderate activation | ○ Weak or minimal activation

Figure 1 displays the Neural Activity Patterns in Programming language and Mathematical tasks (description). Programming involves executive control, symbolic reasoning and abstraction as it mainly involves using the angular gyrus lateral prefrontal cortex posterior parietal cortex, and the MDN. Regions in language that are moderately recruited, particularly in novices, are the superior temporal gyrus and Broca's area. Though it hardly involves language parts, mathematical reasoning

activates parietal and frontal parts that overlap with programming. The functional integration of the linguistic and mathematical networks of mathematical networks is revealed in programming arrows. All these findings indicate that programming is not a totally language-like or a totally math-like thing. It instead occupies a separate cognitive domain on top of high volumes of domain-general executive functions and incorporates elements of both fields. More evidence that proficiency involves a switch between language-based processing in favour of abstraction-based reasoning, as well as increased neural efficiency and decreased cognitive load, is the differences between novices and experts. The implications of the findings for education imply that the knowledge may be supported through scaffolding learning through exercises that focus on syntax, to problem solving that involves a substantial amount of abstraction. They are also involved in the creation of artificial intelligence, as the knowledge of human neural strategies of symbolic reasoning can be used to create AI systems, which can be programmed flexibly and logically. Everything taken together, we find programming to be a cognitively rich and multifaceted activity, and its neural correlates reflect this, and the aspect of being a hybrid that depends on executive control and experience.

## CONCLUSION

The synchronised work of multiple neural networks produces a programming of a unique human capacity of cognition. This hybrid cognitive domain that is neither linguistic nor mathematical is merged with executive control, logical reasoning, and structured symbolic communication. The paper is a synthesis of cognitive psychology, neuroimaging, computational modelling and cognitive ergonomics research to demonstrate that programming engages the multiple-demand network parietal regions associated with symbolic and numerical reasoning and other language circuits, particularly among novices. The neuroplasticity of acquiring the skill is emphasised by comparative research involving novices and experts that demonstrates that mastery of the skill changes brain pathways and reduces cognitive effort, and promotes schema-based thinking. The findings underscore the major contributions of the papers that encompass the clarification of the neural and cognitive architecture of programming that differentiates language code and mathematical processing and provides insight into the processes that promote the attainment of expertise. This research has interdisciplinary implications, which make it have a wider meaning. Education can be scaffolded by learning syntax-based exercises, problem-solving based on abstraction, and considering the neural and cognitive requirements of programming. The concept of cognitive load and neural engagement may be used to create neurodiversity-specific interventions so that they can be made more accessible to learners. Regarding technology and artificial intelligence, knowing human programming cognition can be regarded as an example of creating AI systems that are capable of reasoning in a flexible, rule-based fashion. It also provokes the development of programming environments that will be able to respond to changing cognitive states in real-time. Future research must be

conducted on the dynamics of the neural architecture over time as programmers become more experienced in the way programming cognition differs across cultures, as well as how the visual motor and memory systems combine multimodal during coding. The present study provides prospects in education technology development and human-AI cooperation by helping to bridge the areas of neuroscience and programming to comprehend one of the most intriguing nexus points between artificial intelligence and computation.

## REFERENCES


Binz, M., et al. (2025). A foundation model to predict and capture human cognition. *Nature*, 616(7957), 1–6. https://doi.org/10.1038/s41586-025-09215-4

Duran, R. (2022). Cognitive load theory in computing education research. *Communications of the ACM*, 65(3), 44–53. https://doi.org/10.1145/3483843

Fan, G., et al. (2025). The impact of AI-assisted pair programming on student motivation, programming anxiety, and performance. *International Journal of STEM Education*, 12(1), 16. https://doi.org/10.1186/s40594-025-00537-3

Frank, M. C., et al. (2023). Cognitive modelling using artificial intelligence. *Trends in Cognitive Sciences*, 27(6), 479–491. https://doi.org/10.1016/j.tics.2023.03.005

Gkintoni, E., et al. (2025). Challenging cognitive load theory: The role of intrinsic and extraneous load in programming education. *Educational Psychology Review*, 37(1), 1–22. https://doi.org/10.1007/s10648-024-09723-1

Ivanova, A. A., et al. (2020). Computer programming is a novel cognitive tool that has transformed modern society. *eLife*, 9, e58906. https://doi.org/10.7554/eLife.58906

Khan, S. A., Khan, M. L., & Shehzad, M. (2024). Personality factors, perceived parenting styles as predictors of substance use among university students. *Spry Contemporary Educational Practices*, *3*(1).

Khan, S., Ullah, S., Abbas, M.M., Akhtar, M., KAleem, M.F. & Ali, R.(2024). Effect of social media usage patterns on academic performance and psychological well-being of undergraduate students. Migration Letters21 (S3), 1261-1274.

Khan, S., Ullah, S., Siraj, D., & Amjad, I, A., (2025). Examining Classroom Strategies: Reciprocal Teaching and its Effect on Reading Comprehension. *Sage Open*, *15*(4), 21582440251380709.

Krueger, R., Huang, Y., Liu, X., Santander, T., Weimer, W., & Leach, K. (2020). Neurological divide: An fMRI study of prose and code writing. *Proceedings of the 42nd International Conference on Software Engineering*, 2020, 1–12. https://doi.org/10.1145/3377811.3380394

Lebiere, C. (2025). Cognitive models for machine theory of mind. *Topics in Cognitive Science*, 17(1), 1–17. https://doi.org/10.1111/tops.12773

Malloy, T., & Gonzalez, C. (2024). Applying generative artificial intelligence to cognitive models of decision making. *Frontiers in Psychology*, 15, 1387948. https://doi.org/10.3389/fpsyg.2024.1387948

Masih, S., Punchanathan, U. E., & Osigwe, L. K. (2024). Influence of Green Transformational Leadership on Environmental Citizenship Behaviors via Green HRM Practices. Application of Resource-Based View Theory in the Tourism Industry of Pakistan. *International Journal of Business and Economic Affairs*, *9*(3), 1-15.

MIT News Office. (2020). To the brain, reading computer code is not the same as reading English. *MIT News*. https://news.mit.edu/2020/brain-reading-computer-code-1215

Quintero-Manes, R., et al. (2024). Differentiated measurement of cognitive loads in computer programming. *Education and Information Technologies*, 29, 123–145. https://doi.org/10.1007/s12528-024-09411-7

Rosen, J. (2020). This is your brain on code: JHU researchers decipher neural activity while reading code. *Johns Hopkins Hub*. https://hub.jhu.edu/2020/12/17/brain-activity-while-reading-code/

Sandoval-Medina, C., Arévalo-Mercado, C. A., Muñoz-Andrade, E. L., & Muñoz-Arteaga, J. (2024). Self-explanation effect of cognitive load theory in teaching basic programming. *Journal of Information Systems Education*, 35(3), 303–312. https://doi.org/10.62273/GMIV1698

Tanimoto, S. (2022). Programming in an fMRI scanner: A report from the field. *Proceedings of the 2022 IEEE Symposium on Visual Languages and Human-Centric Computing*, 1–5. https://doi.org/10.1109/VLHCC54747.2022.00010

ul Haq, A., & ur Rehman, K. Major Challenges and Opportunities for Islamic Banking and SMEs in Pakistan.

Ullah, S., Akhtar, M., & Kaleem, M. F. (2025). Assessing the Role of Metacognition in Mathematics Education for Female Students: empirical Evidence from Pakistan. *ACADEMIA International Journal for Social Sciences*, *4*(1), 927-932.

Ullah, S., Khatoon, M., Abbas, M. M., Chaudhery, F. R., Kaleem, M. F., & Akhtar, M. (2023). Effect of Collaborative Learning on Elementary School Students' Academic Achievement in Science. *Journal of Hunan University Natural Sciences*, *50*(10).

Zhao, J., et al. (2022). Cognitive psychology-based artificial intelligence review. *Frontiers in Psychology*, 13, 958215. https://doi.org/10.3389/fpsyg.2022.958215