Convergent Representations of Computer Programs in Human and Artificial Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

What aspects of computer programs are represented by the human brain during comprehension? We leverage brain recordings derived from functional magnetic resonance imaging (fMRI) studies of programmers comprehending Python code to evaluate the properties and code-related information encoded in the neural signal. We first evaluate a selection of static and dynamic code properties, such as abstract syntax tree (AST)-related and runtime-related metrics. Then, to learn whether brain representations encode fine-grained information about computer programs, we train a probe to align brain recordings with representations learned by a suite of ML models. We find that both the Multiple Demand and Language systems–brain systems which are responsible for very different cognitive tasks, encode specific code properties and uniquely align with machine learned representations of code. These findings suggest at least two distinct neural mechanisms mediating computer program comprehension and evaluation, prompting the design of code model objectives that go beyond static language modeling. We make all the corresponding code, data, and analysis publicly available.[1]

1 Introduction

Computer code comprehension is a complex task which recruits multiple cognitive processes—from syntactic parsing to mentally simulating programs. Despite the prevalence of this task, the representations of code processed in the human brain during code comprehension remain uninvestigated. Is it possible that common code properties and program semantics are faithfully represented in brain activity patterns when code is read and evaluated? A few prior works have used data derived from functional magnetic resonance imaging (fMRI) and electroencephalography (EEG) to locate physical regions in the brain involved in code-related activities like code comprehension and debugging [Siegmund et al., 2017, Floyd et al., 2017, Peitek et al., 2018, Castelhano et al., 2019, Ivanova et al., 2020, Liu et al., 2020, Ikutani et al., 2021, Peitek et al., 2021], code writing [Krueger et al., 2020, Karas et al., 2021], and data structure manipulation [Huang et al., 2019]. These studies have helped determine whether code-related activities join other activities supported by those brain regions, such as working memory (processed by a set of brain regions known as the Multiple Demand system) or language processing (processed by the Language system–another set of brain regions). While these results improve our understanding of the brain regions involved in code comprehension, it still remains unclear what specific code-related information these regions encode. For example, does the response in a region seen during code comprehension encode specific syntactic or semantic code properties? Do responses from multiple brain regions correspond to the same set of properties? Or, are different code properties encoded in different regions?

[1]Project URL - `https://github.com/anonmyous-author/anonymous-code`

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

A possible approach and its limitation. One way to learn what information is encoded in the brain is to decode a code property of interest from recordings of brain signals when reading code (through fMRI or EEG). Being able to decode the property accurately from a specific region of the brain establishes that information related to that code property is faithfully represented in that brain region. A question central to such a decoding analysis is the choice of the target code property–what code properties should be investigated? We can hand-select a set of fundamental properties of code and test if they can be decoded. While helpful, such a set will not preclude other, more complex aspects of code possibly being encoded.

ML models of code as a tool to reverse engineer what is encoded. To address the limited scope of hand-selected properties, we look to machine learning (ML) models trained on code. Dubbed *code models*, they are trained on large corpora of code, in an unsupervised manner, to learn *ML model representations* of computer programs. Code models are increasingly being used in software engineering workflows [Allamanis et al., 2018a], and have been shown to perform well on tasks like code summarization [Alon et al., 2019], detecting variable misuse [Bichsel et al., 2016], and more recently, code auto-completion [Chen et al., 2021]. These representations have been shown to encode and describe complex code properties [Bichsel et al., 2016, Allamanis et al., 2018b, Srikant et al., 2021]. Successfully decoding these code model representations from brain activity data then allows us to probe whether complex code properties are also encoded in the brain.

Besides serving as a tool to reverse engineer what is encoded in different brain regions, another distinct advantage of decoding code model representations is its potential to serve as a tool to reverse engineer our cognitive processes. It is currently unclear what mechanisms drive code comprehension. If we find one class of ML models (say, masked language models) to be more predictive than another (say, autoencoders), it is reasonable to suspect that our brains optimize objectives more similar to that of masked-LMs than that of autoencoders when comprehending code. As a corollary, a poor correspondence between the information encoded by our brains and ML models suggests the possibility of unexplored neural architectures and objectives which may better model our cognition, which in turn may outperform extant ML models. Yamins et al. [2013] first showed how information encoded in our visual system resembles what convolutional neural networks learn when trained to recognize images. We attempt to establish a similar correspondence between code models and the human brain in comprehending code.

Why code comprehension? We focus on code comprehension in this work because very little of this important skill has been analyzed from a cognitive neuroscience perspective while steady advances are being made in training ML models to understand code and increase programmer productivity [Hellendoorn and Sawant, 2021]. Extant ML models for understanding programming are direct adoptions of the state-of-the-art in language processing research. However, recent works in neuroimaging like Liu et al. [2020] and Ivanova et al. [2020] suggest that code comprehension does not share the same neural bases as natural language comprehension. Do code models then mimic human cognition of programs? Do language models? If not, can we consider other model architectures and training objectives which are directly inspired by results from neuroscience? Our work provides early considerations towards these directions.

Further, characterizing the representations of different code properties in the human brain can help learn more about the nature of human algorithmic problem solving more generally. We can characterize signatures of brain activity corresponding to fundamental logic operators like iterative reasoning, conditional reasoning, retrieving calculations that were previously computed, *etc.* seen in individual participants when solving carefully designed code comprehension tasks. Such signatures will then allow us to characterize other tasks which involve similar logic operators, but which are not naturally described as computer programs *e.g.* general logic reasoning, problem solving, and more generally, our ability to employ abstractions and form concepts [Rule et al., 2020].

Our setup. We introduce a framework to evaluate the code properties encoded in human brain representations of code by analyzing fMRI recordings of programmers comprehending Python programs. We present two means of proceeding— one, probing of brain region representations for specific code properties, and two, analyzing the mapping of these representations onto various code models with differing model complexity. We utilize the publicly available dataset from Ivanova et al. [2020] for all our analyses as it offers high quality, granular brain response data on code comprehension stimuli controlled across multiple code properties. We train affine models on brain representations to predict hand-selected code properties which express syntactic and semantic behavior of programs. Similarly,

we predict code representations obtained from a suite of code models. We investigate the effects that brain regions, the nature of code properties, and the complexity of the code models have on the accuracies of these prediction tasks.

Key findings. We explore two questions: (1) Do brain systems encode specific code properties? Are there differences in how well each brain system encodes each code property? (2) Do brain systems encode more complex properties of code, derived from the representations of code models? We find the Multiple Demand system and the Language system consistently encode both hand-selected code properties and data-derived code model representations, using features which cannot be explained by only low-level visual characteristics of the code. Within this set, the Multiple Demand system most effectively decodes runtime properties of code like the number of steps involved in a code's execution, while the Language system decodes syntax-related properties like the number of tokens in a program and the control structures present in it.

Our results improve the cognitive science domain of human program comprehension–by improving our understanding of the functional organization of the brain systems we study, as well as objectives employed to train machine learning models of code–by offering new objectives for the design of code models.

We provide an open-source framework to replicate our experiments, and we release our data and analysis publicly. Link - `https://github.com/anonmyous-author/anonymous-code`. This should enable authors from other neuroimaging studies or code model developers to collaborate and analyze data across these works, which will also help amortize the high costs of carrying out such experiments.

2 Related Work

Of the prior works that have investigated the neural bases of programming through fMRI and EEG techniques [Siegmund et al., 2017, Floyd et al., 2017, Peitek et al., 2018, Castelhano et al., 2019, Huang et al., 2019, Krueger et al., 2020, Ivanova et al., 2020, Liu et al., 2020, Ikutani et al., 2021, Peitek et al., 2021] and through behavioral studies [Prat et al., 2020, Casalnuovo et al., 2020, Crichton et al., 2021], the following probe brain recordings for program properties encoded in them.

Floyd et al. [2017] learn a linear model to successfully classify whether an observed brain activity corresponds to reading code or reading text. Ikutani et al. [2021] study expert programmers and show that it is possible to classify code into the four problem categories–math, search, sort, and string from the brain activations corresponding to the code. Similarly, Liu et al. [2020] classify whether a brain signal corresponds to code implementing an `if` condition or not. Peitek et al. [2021] analyze correlations between brain recordings of participants reading code and a set of code complexity metrics.

In testing for code properties, our work uses a similar methodology (a linear model trained on fMRI data), but we evaluate a larger set of static and dynamic code properties, often reflecting key programming constructs like control flow. Further, we perform these tests in the brain regions identified by Liu et al. [2020] and Ivanova et al. [2020] as being responsive specifically to code comprehension, offering finer insight into the content of these specific regions' representations. In addition, we study representations generated by a suite of ML models with varying complexity and compare those learned representations to brain representations.

Brain representations have also been studied in domains like natural language, vision, and motor control. Among related works in natural language, a domain that resembles programming languages, Mitchell et al. [2008], Pallier et al. [2011], Brennan and Pylkkänen [2017], Jain and Huth [2018], Gauthier and Levy [2019], Schwartz et al. [2019], Wang et al. [2020], Schrimpf et al. [2021], Cao et al. [2021], Caucheteux et al. [2021], Toneva and Wehbe [2019] have studied brain representations of words and sentences by relating them to representations produced by language models. While the broader tools we use to investigate these representations, like multi-voxel pattern analysis (MVPA), are similar to some of these prior works, our focus is on properties specific to code and not natural language.

3 Background

We provide a brief background on fMRI signals as a proxy for brain representations and describe the brain systems that we probe in this work.

Measuring brain activity with fMRI. Functional magnetic resonance imaging (fMRI) is a brain imaging technique used to measure brain activity in specific brain regions. When a brain region is active, blood flows into the region to aid its processing. An MRI machine measures this change in blood flow, and reports BOLD (blood oxygen level dependent) values sampled at the machine's frequency [Glover, 2011, usually 2 seconds]. The smallest unit of brain tissue for which BOLD signal is recorded is called a voxel (an equivalent of a 3D pixel); it comprises several cubic millimeters of brain tissue. For our analyses, we select subsets of voxels belonging to specific brain systems. Following common practice, the parameters of a general linear model, fit to time-varying BOLD values, are used as a measure of the overall activation in each voxel in response to a given input. It is the values of these parameters that, in concordance with common practices in the neuroscience community, can be accessed as brain representations.

Brain systems. A system of brain regions can span different areas of the brain but behaves as a holistic unit, showing similar patterns of engagement across a given cognitive task. We probe the following systems in our work: (a) **Multiple Demand (MD) system:** this system of regions is known to engage in cognitively demanding, domain agnostic tasks like problem solving, logic, and spatial memory tasks. Liu et al. [2020] and Ivanova et al. [2020] reported that this system is active during code comprehension. (b) **Language system (LS):** this system responds during language production and comprehension across modalities (speech, text) and languages (across 11 language families, including American sign language). (c) **Visual system (VS):** these regions respond primarily to visual inputs. (d) **Auditory system:** these regions respond primarily to auditory inputs.

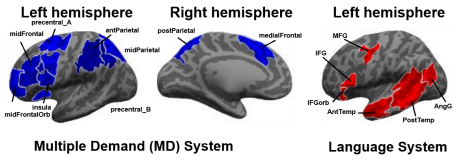

Figure 1: The approximate locations of MD and the Language systems in the human brain. The regions depicted are used as a starting point to functionally localize these systems in individual participants.

We probe the Visual system and Auditory system since they are involved in general perception. While we do not expect activity in Auditory system, we expect the Visual system to reflect low-level visual properties of the code (*e.g.* code length and indentation to reflect code-related properties. See Appendix A for a detailed description of these brain systems along with relevant references.

4 Brain and Model Representations

We describe in this section the method we follow to gather representations of code in the brain (Section 4.1), evaluate the different code properties they encode (Section 4.2), and how we compare brain representations to those generated by ML models (Section 4.3).

4.1 Brain representations and decoding

We provide a summary of how we process activation signals in the brain elicited by code comprehension, to probe whether they encode any specific code properties. We provide details in Appendix B.

Dataset. We use the publicly available brain recordings released as part of the study by Ivanova et al. [2020] (MIT license). It contains brain recordings of 24 participants, each of whom gave consent and is not personally identifiable according to Ivanova et al. [2020] protocol requirements, reading 72 programs from a set of 108 unique Python programs. The 72 programs were presented in 12 blocks of 6 programs each. These programs were 3-10 lines in length and contained simple Python constructs, such as lists, *for* loops and *if* statements. A whole program was presented at once, and the task required participants to read the code and mentally compute the expected output, press a button when done, and select one of four choices presented to them which matched their calculated output.

From dataset to brain representations. The original dataset contains 3D images of the brain of each participant. Each voxel value in these images is an estimate of the response strength in this voxel when a particular code (or sentence) problem is presented. To determine which brain systems contain information about particular code properties, we focus our analyses to four systems – MD, Language, Visual, and Auditory (Section 3). A vector of voxels' activation values in each brain system is then taken to constitute that system's *representation* of a computer program and serves as an input to all our analyses. See Appendix B for details on data processing and voxel selection.

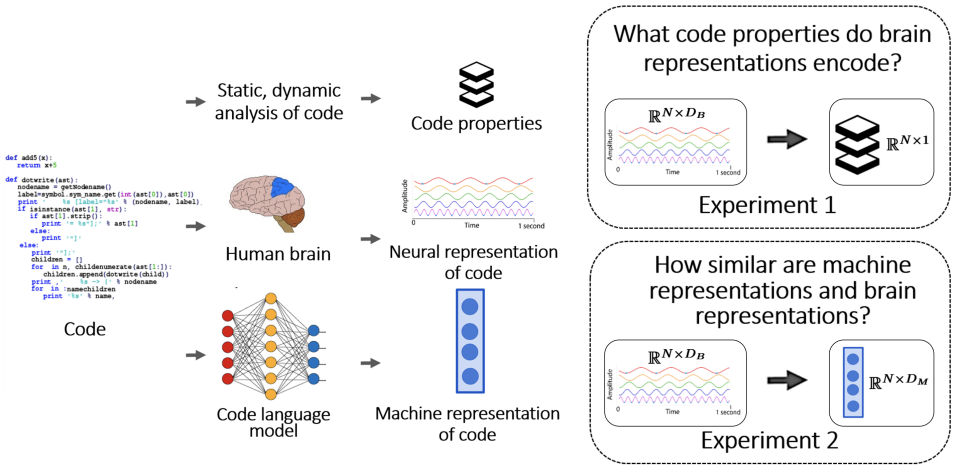

Figure 2: **Overview.** The goal of this work is to relate brain representations of code to (1) specific code properties and (2) representations of code produced by language models trained on code. In Experiment 1, we predict the different static and dynamic analysis metrics from the brain MRI recordings (each of dimension D_B) of 24 human subjects reading 72 unique Python programs (N) by training separate linear models for each subject and metric. In Experiment 2, we learn affine maps from brain representations to the corresponding representations generated by code language models (each of dimension D_M) on these 72 programs.

Analyzing brain representations of code. We probe brain representations from each participant separately. We do not average data across participants since the regions which respond to any task (comprehending code in our case) need not align anatomically. For each of two experiments—decoding different code properties, and mapping to code model representations—we train ridge regression/classification models which take as input normalized brain representations per participant. We hence learn 24 different regression models, for each code property or code model (one per participant), and then report the mean performance of these models across participants. This procedure is also referred to as multi-voxel pattern analysis (MVPA) [Norman et al., 2006]. Linear models are conventionally preferred for probes into brain representations since there has been evidence supporting the idea that other brain areas linearly map information from such brain representations [Kamitani and Tong, 2005, Kriegeskorte, 2011]. We choose a linear model primarily to control for over-fitting in light of the relatively small dataset.

A remark on data scarcity. For each participant, we train a linear regression/classification model with L2-regularization for on unique cross-validated leave-one-run-out folds of the 72 programs they attempted (or 48 programs when sentences are removed). On the order of 1000 voxels were selected from each brain region responding to any given program per participant, thus resulting in a feature set of dimensions 72×1000. Such dimensions allow for strong statistical tests to support the robustness of the predictions made. As a baseline for the model predictions, we use the accuracy of a null permutation distribution generated from sampling 1000 random assignments of the labels.

4.2 Code properties

We attempt to decode the following code properties from brain representations. See Appendix B for details. (a) **Code vs. sentences:** classify whether an input stimulus is a code or an English sentences describing a code (referred to as *sentences*). See Figure B, Appendix B for an example. (b) **Variable language:** classify whether a program contains variable names written in English or Japanese (written in English characters). (c) **Control flow:** predict whether a program contains a loop (`for` loop), a branch (`if` condition), or has sequential instructions. (d) **Data type:** predict whether a program contains string or numeric operations. (e) **Static analysis:** predict static properties of a program like *token count*: number of tokens in the program, *node count*: number of AST nodes, *cyclomatic complexity*, and *Halstead difficulty*. The latter two metrics have been used by software engineering practitioners to quantify the complexity of code, and to quantify the difficulty a human would experience when comprehending code respectively. We defer predicting other advanced static analysis metrics such as tracking abstract interpretation joins, data flow analysis-related metrics, *etc*. to future work. (f) **Dynamic analysis:** predict information about a code's execution behavior like *runtime steps*: number of instructions executed in the program, and *bytecode ops*: number of bytecode operations executed in running the program.

Program length as a potential confound. Since the properties we examine can also potentially be differentiated using program length and other low-level code features, it is a potential confound in our experiments. We measured the inter-correlations of these properties, and their correlation to the number of tokens in the program (program length; see Appendix G). We expectedly found the four *static analysis* properties to be highly correlated to each other and to *bytecode ops*. We hence use one representative metric each from the two categories of properties for the rest of our analysis–*token count* for *static analysis*, and *runtime steps* for *dynamic analysis*. Importantly, the other properties we examine cannot be explained by program length alone, and therefore program length is not a confound in our experiments.

Mapping to code properties. The brain representations (Section 4.1) are mapped to each of the code properties by training a ridge regression (for *static analysis* and *dynamic analysis* properties; continuous values) or a classification model each for every participant-property pair. To evaluate model performance, we use classification accuracy when the predicted values are categorical (*e.g.* string vs. numeric data types), and the Pearson correlation coefficient when the predicted values are continuous (*e.g.* number of runtime steps). We choose Pearson correlation over RMSE, the canonical distance metric for continuous values, for its simplicity and interpretability. See Appendix I for the RMSE results, which lead to the same conclusions as the correlation results. When testing for the significance of these predictions, we perform false discovery rate (FDR) correction for the number of brain systems tested and the number of properties tested. See Appendix B for a detailed description of model hyper-parameters, cross-validation settings, and significance testing.

4.3 Model representations and decoding.

We evaluate a bench of unsupervised language models, spanning from count-based language models to transformer neural networks [Vaswani et al., 2017]. These models were all trained on large ($\sim$1M programs) Python datasets [Husain et al., 2019, Puri et al., 2021]. We use the output of the trained encoders (raw logits) in each of the neural network models as representations of the code input to the model. We vary the general complexity of these models to test whether that variation is meaningful in establishing the quality of brain to model fits. Model complexity here is the number of a model's learnable parameters. We evaluate the following models, ordered by their increasing model complexity: simple frequency-based language models—*bag-of-words*, *TF-IDF*; auto-encoder based unsupervised models—*seq2seq* [Sutskever et al., 2014], *CodeTransformer* [Zügner et al., 2021], *CodeBERT* [Feng et al., 2020], *CodeBERTa* [HuggingFace, 2020]; auto-regressive models with similar model complexity—*XLNet* [Yang et al., 2019], *CodeGPT* [Microsoft, 2021].

Baseline: Token projection model. We compare the results of the above models against an aggressive baseline (relative to the null-distribution labeling baseline), a *token projection model* provided by using unique Gaussian-distributed random vectors for the token embeddings in a vocabulary, and returning the sum of these token embeddings across a program. The resultant embedding is not transformed by any model or any weights–it instead serves as a proxy for the tokens that appear in the program. The results of this baseline model should be interpreted as the level of performance achievable from the presence of tokens alone with no semantic or syntactic information.

Mapping to code model representations. The brain representations (Section 4.1) are mapped to code representations by training another set of ridge regression models to learn an affine map, and a ranked accuracy metric is used to compare outputs. Ranked accuracy scores are commonly used in information retrieval where several elements in a range are similar to the correct one. In our case, the top-ranked prediction by the linear model indicates the closest fit (Euclidean distance) to the code model's representation. When reporting result significance, we perform false discovery rate (FDR) correction for the number of brain systems and the number of models. See Appendix B for a detailed discussion of the implemented code models and the corresponding metrics. All experiments in this work were run on a single 8-core laptop in under an hour following setup.

5 Experiments & Results

Our experiments address two research questions:

- **Experiment 1.** How well do the different brain systems encode specific code properties? Do they encode the same properties?

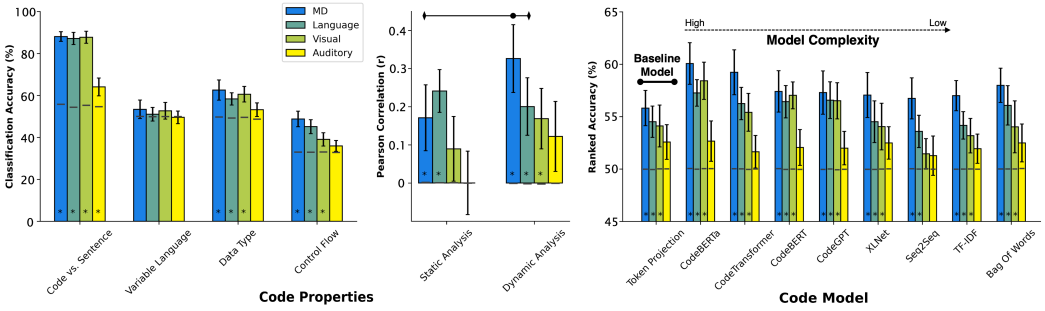

Figure 3: Affine models are learned on brain representations to predict each of the code properties described in Section 4.2, and a collection of code models described in 4.3, for each of the 24 participants. The mean decoding score across subjects is shown here, and error bars reflect the 95% confidence interval of the mean subject score. A solid line on each bar presents the empirical baseline for a null permutation distribution on shuffled labels. All decoding scores were compared to this permuted null distribution using a one-sample z-test, and the significance threshold was defined at $p < 0.001$; FDR-corrected for the number of tests in each panel. Statistically significant results are denoted with a $*$, marked at the base of the bars. Additionally, a solid bar above the middle panel highlights that the MD system decodes *dynamic analysis* more effectively than the LS and more selectively than *static analysis*.

- **Experiment 2.** Do brain systems encode additional code properties represented by computational language models of code?

5.1 Experiment 1 - How well do the different brain systems encode specific code properties? Do they encode the same properties?

We analyze the classification models and regressions trained on brain representations to predict each of the code properties described in Section 4.2. The results of our analyses are summarized in Figure 3. The classification and regression tests are marked on the x-axis of the left and right subplots respectively; the classification accuracy or Pearson correlation for each of the tasks is marked on the y-axes. We plot dynamic and static properties separately from the others because their baselines are different due to a difference in the similarity metric (classification accuracy vs. Pearson correlation). The baselines for the categorical code properties differ from each other due to variation in the number of target classes.

Auditory and Visual systems. The Auditory system and the Visual system serve as negative and baseline controls for the other systems. Since our code comprehension task is visual, we do not decode any meaningful information between programs from Auditory system, although we do observe decoding of *code vs sentences*, perhaps explained by previous work showing auditory cortex activation during silent sentence reading [Perrone-Bertolotti et al., 2012]. The Visual system serves as a baseline for low-level visual features of the code (the layout and indentation of the code, the presence of letters and alphabets in the programs, *etc.*). In Analysis 2, we show that the MD and the Language systems encode more than such visual features. The MD and the Language systems yield the following observations.

Analysis 1 - How accurately are different properties predicted by MD and LS? The MD and LS together decode all the properties well above chance barring variable language. The variable language finding is consistent with Ivanova et al. [2020], who show a lack of any significant difference in the aggregate neural activity between English and Japanese variables names—variable names seem to not be encoded any differently in the context of programs we study. To analyze the other results, we use paired two-sample t-tests ($p < 0.05$; FDR-corrected) and examine whether for a given property, any one brain system decodes it significantly more effectively than another (details in Table 8, Appendix E). We find that the MD system decodes the *dynamic analysis* property better than the Language system. We additionally test if any brain system has a preference for a specific code property over another. We find that the MD decodes the *dynamic analysis* property significantly better than the *static analysis* property. **These findings establish the unique role of the MD system in encoding code-simulation and execution related information—an important of aspect of code comprehension.**

Analysis 2 - Multi-system partial regression analysis. The decoding performance of the VS is comparable to that of the MD and LS (Figure 3). To assess the possibility that all three systems - MD, LS, and VS encode the same properties (all potentially related to low level program features), we employ a multi-system partial regression analysis. For each brain system, MD and LS (S_i), we train two models–one which decodes from VS, and another which decodes from S_i+VS. If the difference in the prediction accuracies between the two models is significant, we conclude that S_i encodes at least some information which is orthogonal to the information encoded by the Visual system. For all core properties, *control flow*, *data type*, *dynamic analysis*, and *static analysis*, we find the MD to encode information orthogonal to the VS. For *control flow* and *static analysis*, the LS also encodes information orthogonal to the VS. This suggests that low-level code properties are insufficient to explain the key results from Experiment 1. Other combinations in the regression model reveal that the MD encodes information orthogonal to the LS when predicting *code vs. sentences* and *dynamic analysis*. Detailed results are tabulated in Table 15, Appendix F.

Key learnings from Experiment 5.1. All core code properties are decodable from the representations of high-level brain systems, and this information is beyond that which can be explained from low-level visual information alone. Although no property is exclusively encoded in any one brain system, the MD system significantly encodes *dynamic analysis*-related properties–more than what the LS encodes, and more than *static analysis* properties. Similarly, we find evidence for the LS to also significantly encode *static analysis* and *control flow* related properties. These are new results on the nature of code properties different brain systems seem to process and encode. To explore properties which may not be specified by the set we investigate in this experiment, in the following section, we leverage code models as hypothesis-free proxy representations for code syntax and semantics, and see if any one system preferentially encodes a code model.

5.2 Experiment 2 - Do brain systems encode additional code properties encoded by computational language models of code?

We train ridge regression models with brain representations of programs from specific brain systems to predict code model representations of the same programs. The set of results from this experiment are summarized in Figure 3.

Auditory and Visual systems. Similar to Experiment 1, these systems perform as expected, with the Auditory system exhibiting the lowest decoding performance across code models, and the Visual system acting as a proxy for low-level information (Table 10, Appendix E).

Analysis 1 - How well do brain representations in MD and LS map to code model representations? We find that the MD and LS map to all the models in our suite significantly more accurately than the null permutation baseline (Figure 3). Further, we find that the MD ranked accuracy is higher than LS for *CodeBERTa*, *CodeTransformer*, *seq2seq* and *TF-IDF* (differences evaluated using two-sample t-tests; $p < 0.05$; FDR-corrected. See Table 10, Appendix E).

Analysis 2 - The effect of model complexity on decoding to code models. We investigate the impact model complexity has on the performance of the mapping between brain and code representations. Model complexity here is the number of a model's learnable parameters. In analysis 1 above, we saw that the MD system mapped to the representations of *CodeBERTa* and *TF-IDF* significantly more accurately than the Language system. These models vary substantially in their complexity, thus suggesting the lack of a specific relationship between model complexity and preferential mapping to one of these given brain systems. We also compare each of these code models against the *Token Projection* baseline model. In a set of paired two-tailed t-tests ($p < 0.05$; FDR-corrected), we find that the MD maps to *CodeBERTa*, *CodeTransformer*, and *bag-of-words* significantly more accurately than to the *Token Projection* model (Table 11, Appendix E).

Key learnings from Experiment 5.2. The MD decodes representations from several models of code more accurately than LS, providing evidence that some aspects of code are more faithfully represented in the MD. A follow up partial regression analysis, as in 5.1, reveals that for nearly every single model evaluated, MD encodes information orthogonal to the LS, and each system encodes information above low-level VS (Table 16, Appendix F). Curiously, we also note that a non-trivial portion of these mappings can be explained using only token-level information, suggesting that the brain signals we access largely encode token-level information. We discuss this in the following section.

6 Discussion

Through this study, we learn what computer program-related information can be decoded from the brain, and which brain systems primarily encode that information.

We show that the MD system preferentially encodes *dynamic analysis*-related properties when compared to other brain systems and other properties. Further, we find the Language system encodes syntax-related properties like *control flow* and *static analysis*. These findings are complementary to the results from Liu et al. [2020] and Ivanova et al. [2020]–they show how the MD system is recruited in both mentally simulating code and comprehending code. They however do not find any consistent response in the Language system to either code simulation or comprehension. Our results show that despite not exhibiting significant average responses, these systems do encode code-specific properties, improving our understanding of these brain systems' functional organization.

Another key contribution of our work, from Experiment 2, is demonstrating that it is possible to map brain representations to representations learned by code models. In particular, we observe a general preferential encoding of the properties represented by code models in the MD over LS. This is particularly noteworthy since these models, which are trained on source code symbols, can be mapped more faithfully from the representations of a network implicated in problem-solving than one associated with composition in languages. We also considered model complexity as a relevant feature, but do not observe any consistent trend across this dimension. In fact, the MD and LS map to a combination of token projection embeddings almost as well as to complex models like *XLNet*. One plausible explanation for this surprising result is that the program stimuli are simple enough to allow the different properties evaluated in our work (*control flow*, *data type*, *etc.*) to be discerned from token level information alone (as validated in Experiment 1), which is likely why the *Token Projection* model also predicts these properties very well (Table 4, Appendix C.2). Taken together, these data suggest that the information being decoded from brain activations in these two regions is driven at least by the information conveyed by tokens in the programs. This finding is notably consistent with work in the field of natural language which show swaths of cortex are predicted primarily by the token-level properties of sentence stimuli [Toneva et al., 2022a]. That said, the precise causal relationship between stimulus properties and predictivity of cortical activations merits further investigation, *e.g.*, via the approach of Toneva et al. [2022b].

Way forward. Our findings have the potential to improve our understanding of the organization of the human brain, which can in turn lead to the design of better code models. In computer vision, results by Tschopp et al. [2018], Schrimpf et al. [2020] show how deep network architectures that mimic the visual system exhibit superior image classification rates on image recognition tasks. Our findings prompt yet another reconsideration of the current design of ML models of code. Extant code model architectures do not explicitly model the Multiple Demand system in any way–they only model syntactic information and infer dependency information from program ASTs. Taking inspiration from the role of the Multiple Demand system we identified in this work, modeling dynamic runtime information as well as static code structure should be explored.

We also provide initial results supporting the ability to decode specific and fundamental code-related primitives like control flow information. This sets us up to study other complex human cognitive processes which involve such primitives, but which are not naturally described in terms of code, like general problem solving, employing hierarchical and complex decision making strategies, *etc*. Such a study also promises to support the recent proposal of a child as a hacker Rule et al. [2020].

Limitations. The average program in any software project exhibits non-trivial control and data dependencies, object manipulation, function calls, types, and state changes. However, the programming tasks in Ivanova et al. [2020] are short snippets of procedural code with limited program properties. Responses to longer programs, with more complex properties and across multiple languages, should be studied on a larger number of participants in the future, in order to build on the trends provided by our work. Further, while there are multiple equally important aspects to programming like designing solutions, selecting appropriate data structures, and writing programs, here we have chosen to study only a single specific activity—code comprehension. Future work should explore these other aspects. Our results do not allow us to infer whether the MD and LS are driven by the same underlying features of code that are used to discriminate between code properties and code models, so future work might consider an encoding analysis. Finally, as with all neural decoding analyses, extracting information from the mental states of participants should be done with caution to ensure it is not used for any exploitative purpose.

References

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. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 140–150, 2017.

Benjamin Floyd, Tyler Santander, and Westley Weimer. Decoding the representation of code in the brain: An fmri study of code review and expertise. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 175–186. IEEE, 2017.

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. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2018.

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.

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.

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.

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*, 8(1), 2021.

Norman Peitek, Sven Apel, Chris Parnin, André Brechmann, and Janet Siegmund. Program comprehension and code complexity metrics: An fmri study. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 524–536. IEEE, 2021.

Ryan Krueger, Yu Huang, Xinyu Liu, Tyler Santander, Westley Weimer, and Kevin Leach. Neurological divide: an fmri study of prose and code writing. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 678–690. IEEE, 2020.

Zachary Karas, Andrew Jahn, Westley Weimer, and Yu Huang. Connecting the dots: rethinking the relationship between code and prose writing with functional connectivity. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 767–779, 2021.

Yu Huang, Xinyu Liu, Ryan Krueger, Tyler Santander, Xiaosu Hu, Kevin Leach, and Westley Weimer. Distilling neural representations of data structure manipulation using fmri and fnirs. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 396–407. IEEE, 2019.

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

Uri Alon, Omer Levy, and Eran Yahav. code2seq: Generating sequences from structured representations of code. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1gKYo09tX`.

Benjamin Bichsel, Veselin Raychev, Petar Tsankov, and Martin Vechev. Statistical deobfuscation of android applications. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 343–355, 2016.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. In *International Conference on Learning Representations*, 2018b. URL https://openreview.net/forum?id=BJOFETxR-.

Shashank Srikant, Sijia Liu, Tamara Mitrovska, Shiyu Chang, Quanfu Fan, Gaoyuan Zhang, and Una-May O'Reilly. Generating adversarial computer programs using optimized obfuscations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PH5PH9ZO_4.

Daniel L Yamins, Ha Hong, Charles Cadieu, and James J DiCarlo. Hierarchical modular optimization of convolutional networks achieves representations similar to macaque it and human ventral stream. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/9a1756fd0c741126d7bbd4b692ccbd91-Paper.pdf.

Vincent J Hellendoorn and Anand Ashok Sawant. The growing cost of deep learning for source code. *Communications of the ACM*, 65(1):31–33, 2021.

Joshua S Rule, Joshua B Tenenbaum, and Steven T Piantadosi. The child as hacker. *Trends in cognitive sciences*, 24(11):900–915, 2020.

Chantel S Prat, Tara M Madhyastha, Malayka J Mottarella, and Chu-Hsuan Kuo. Relating natural language aptitude to individual differences in learning programming languages. *Scientific reports*, 10(1):1–10, 2020.

Casey Casalnuovo, Kevin Lee, Hulin Wang, Prem Devanbu, and Emily Morgan. Do programmers prefer predictable expressions in code? *Cognitive Science*, 44(12):e12921, 2020.

Will Crichton, Maneesh Agrawala, and Pat Hanrahan. The role of working memory in program tracing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2021.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. Predicting human brain activity associated with the meanings of nouns. *science*, 320 (5880):1191–1195, 2008.

Christophe Pallier, Anne-Dominique Devauchelle, and Stanislas Dehaene. Cortical representation of the constituent structure of sentences. *Proceedings of the National Academy of Sciences*, 108(6):2522–2527, 2011.

Jonathan R Brennan and Liina Pylkkänen. Meg evidence for incremental sentence composition in the anterior temporal lobe. *Cognitive science*, 41:1515–1531, 2017.

Shailee Jain and Alexander Huth. Incorporating context into language encoding models for fmri. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Jon Gauthier and Roger Levy. Linking artificial and human neural representations of language. *arXiv preprint arXiv:1910.01244*, 2019.

Dan Schwartz, Mariya Toneva, and Leila Wehbe. Inducing brain-relevant bias in natural language processing models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Shaonan Wang, Jiajun Zhang, Nan Lin, and Chengqing Zong. Probing brain activation patterns by dissociating semantics and syntax in sentences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 9201–9208, Apr. 2020. doi: 10.1609/aaai.v34i05.6457.

Martin Schrimpf, Idan Asher Blank, Greta Tuckute, Carina Kauf, Eghbal A. Hosseini, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. The neural architecture of language: Integrative modeling converges on predictive processing. *Proceedings of the National Academy of Sciences*, 118(45), 2021.

Lu Cao, Dandan Huang, Yue Zhang, Xiaowei Jiang, and Yanan Chen. Brain decoding using fnirs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12602–12611, May 2021.

Charlotte Caucheteux, Alexandre Gramfort, and Jean-Remi King. Decomposing lexical and compositional syntax and semantics with deep language models. In *International Conference on Machine Learning*, pages 1336–1348. PMLR, 2021.

Mariya Toneva and Leila Wehbe. Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/749a8e6c231831ef7756db230b4359c8-Paper.pdf.

Gary H Glover. Overview of functional magnetic resonance imaging. *Neurosurgery Clinics*, 22(2):133–139, 2011.

Kenneth A Norman, Sean M Polyn, Greg J Detre, and James V Haxby. Beyond mind-reading: multi-voxel pattern analysis of fmri data. *Trends in cognitive sciences*, 10(9):424–430, 2006.

Yukiyasu Kamitani and Frank Tong. Decoding the visual and subjective contents of the human brain. *Nature neuroscience*, 8(5):679–685, 2005.

Nikolaus Kriegeskorte. Pattern-information analysis: from stimulus decoding to computational-model testing. *Neuroimage*, 56(2):411–421, 2011.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *CoRR*, abs/1909.09436, 2019. URL http://arxiv.org/abs/1909.09436.

Ruchir Puri, David Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladmir Zolotov, Julian Dolby, Jie Chen, Mihir Choudhury, Lindsey Decker, Veronika Thost, Luca Buratti, Saurabh Pujar, Shyam Ramji, Ulrich Finkler, Susan Malaika, and Frederick Reiss. Codenet: A large-scale ai for code dataset for learning a diversity of coding tasks, 2021.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Daniel Zügner, Tobias Kirschstein, Michele Catasta, Jure Leskovec, and Stephan Günnemann. Language-agnostic representation learning of source code from structure and context. In *International Conference on Learning Representations (ICLR)*, 2021.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

HuggingFace. Codeberta - a roberta-like model trained on the codesearchnet dataset from github. *HuggingFace*, 2020. URL https://huggingface.co/huggingface/CodeBERTa-small-v1.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Microsoft. Codegpt-small-py. *HuggingFace*, 2021. URL https://huggingface.co/microsoft/CodeGPT-small-py.

Marcela Perrone-Bertolotti, Jan Kujala, Juan R. Vidal, Carlos M. Hamame, Tomas Ossandon, Olivier Bertrand, Lorella Minotti, Philippe Kahane, Karim Jerbi, and Jean-Philippe Lachaux. How silent is silent reading? intracerebral evidence for top-down activation of temporal voice areas during reading. *Journal of Neuroscience*, 32(49):17554–17562, 2012.

Mariya Toneva, Tom M. Mitchell, and Leila Wehbe. Combining computational controls with natural text reveals new aspects of meaning composition. *bioRxiv*, 2022a.

Mariya Toneva, Jennifer Williams, Anand Bollu, Christoph Dann, and Leila Wehbe. Same cause; different effects in the brain. In *First Conference on Causal Learning and Reasoning*, 2022b. URL https://openreview.net/forum?id=j5e7PielLe2.

Fabian David Tschopp, Michael B. Reiser, and Srinivas C. Turaga. A connectome based hexagonal lattice convolutional network model of the drosophila visual system, 2018.

Martin Schrimpf, Jonas Kubilius, Ha Hong, Najib J. Majaj, Rishi Rajalingham, Elias B. Issa, Kohitij Kar, Pouya Bashivan, Jonathan Prescott-Roy, Franziska Geiger, Kailyn Schmidt, Daniel L. K. Yamins, and James J. DiCarlo. Brain-score: Which artificial neural network for object recognition is most brain-like? *bioRxiv*, 2020.

John Duncan. The multiple-demand (md) system of the primate brain: mental programs for intelligent behaviour. *Trends in cognitive sciences*, 14(4):172–179, 2010.

Evelina Fedorenko, John Duncan, and Nancy Kanwisher. Broad domain generality in focal regions of frontal and parietal cortex. *Proceedings of the National Academy of Sciences*, 110(41):16616–16621, 2013.

Marie Amalric and Stanislas Dehaene. A distinct cortical network for mathematical knowledge in the human brain. *NeuroImage*, 189:19–31, 2019.

David Glenn Clark and Jeffrey L Cummings. Aphasia. In *Neurological Disorders*, pages 265–275. Elsevier, 2003.

Evelina Fedorenko, Po-Jang Hsieh, Alfonso Nieto-Castañón, Susan Whitfield-Gabrieli, and Nancy Kanwisher. New method for fmri investigations of language: defining rois functionally in individual subjects. *Journal of neurophysiology*, 104(2):1177–1194, 2010.

Jennifer Hu, Hannah Small, Hope Kean, Atsushi Takahashi, Leo Zekelman, Daniel Kleinman, Elizabeth Ryan, Victor Ferreira, and Evelina Fedorenko. The language network supports both lexical access and sentence generation during language production. *bioRxiv*, 2021.

Idan A Blank and Evelina Fedorenko. Domain-general brain regions do not track linguistic input as closely as language-selective regions. *Journal of Neuroscience*, 37(41):9999–10011, 2017.

Aaron J Newman, Ted Supalla, Nina Fernandez, Elissa L Newport, and Daphne Bavelier. Neural systems supporting linguistic structure, linguistic experience, and symbolic communication in sign language and gesture. *Proceedings of the National Academy of Sciences*, 112(37):11684–11689, 2015.

Dima Ayyash, Saima Malik-Moraleda, Jeanne Gallée, Josef Affourtit, Malte Hoffman, Zachary Mineroff, Olessia Jouravlev, and Evelina Fedorenko. The universal language network: A cross-linguistic investigation spanning 45 languages and 11 language families. *bioRxiv*, 2021.

David H. Hubel and Torsten N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148, 1959.

Thad A. Polk, Matthew Stallcup, Geoffrey K. Aguirre, David C. Alsop, Mark D'Esposito, John A. Detre, and Martha J. Farah. Neural Specialization for Letter Recognition. *Journal of Cognitive Neuroscience*, 14(2): 145–159, 2002.

Russell A. Epstein and Nancy Kanwisher. A cortical representation of the local visual environment. *Nature*, 392: 598–601, 1998.

Nancy Kanwisher, Josh McDermott, and Marvin M. Chun. The fusiform face area: A module in human extrastriate cortex specialized for face perception. *The Journal of Neuroscience*, 17:4302 – 4311, 1997.

Sam Norman-Haignere, Nancy G Kanwisher, and Josh H McDermott. Distinct cortical pathways for music and speech revealed by hypothesis-free voxel decomposition. *Neuron*, 88(6):1281–1296, 2015.

Sam V Norman-Haignere, Nancy Kanwisher, Josh H McDermott, and Bevil R Conway. Divergence in the functional organization of human and macaque auditory cortex revealed by fmri responses to harmonic tones. *Nature neuroscience*, 22(7):1057–1060, 2019.

John Ashburner, Jesper L.R. Andersson, and Karl J. Friston. High-dimensional image registration using symmetric priors. *NeuroImage*, 9(6):619–628, 1999.

Edmund T. Rolls, Chu-Chung Huang, Ching-Po Lin, Jianfeng Feng, and Marc Joliot. Automated anatomical labelling atlas 3. *NeuroImage*, 206:116189, 2020.

Thomas J McCabe. A complexity measure. *IEEE Transactions on software Engineering*, pages 308–320, 1976.

Martin Shepperd. A critique of cyclomatic complexity as a software metric. *Software Engineering Journal*, 3(2): 30–36, 1988.

Maurice H Halstead. *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc., 1977.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Main claims are accurately reflected.
   (b) Did you describe the limitations of your work? [Yes] Limitations are noted in Section 6: Limitations.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Societal implications are noted in Section 6: Limitations.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] There are no theoretical proofs presented in this work.
   (b) Did you include complete proofs of all theoretical results? [N/A] There are no theoretical proofs presented in this work.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All code and data is available at the public link `https://github.com/anonmyous-author/anonymous-code`, mentioned on the first page of this draft. A single command is provided to wrap the entire pipeline from environment setup to final plots and tables.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B: MVPA cross-validation and hyper-parameters for all training details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figure 3 for error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.3: Model representations and decoding. The experiments we present in this work are not compute-heavy and can be run on a single 8-core laptop in under an hour.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use a public dataset by Ivanova et al. [2020] and cite the authors.
   (b) Did you mention the license of the assets? [Yes] We mention the license in 4.1: Dataset.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] No new assets were included, except results derived from existing data. All original links to public data and code operating over them are made available and appropriately licensed to do so.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We mention participant consent in Section 4.1: Dataset, and link the original data source for more information.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We mention participant unidentifiability in Section 4.1: Dataset, and link the original data source for more information. No offensive content is included.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not conduct research with human subjects, but this information is available in Ivanova et al. [2020]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We did not conduct research with human subjects, but this information is available in Ivanova et al. [2020]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not conduct research with human subjects, but this information is available in Ivanova et al. [2020]

A Brain regions

Regions of Interest (ROIs). We investigate the representations of code in well-studied systems of brain regions–the MD system, Language system, Visual system, and the Auditory system (details below). A *region* (also referred to as *parcel*) here denotes a contiguous chunk of brain mass involved in a cognitive task. A *system* of regions (also referred to as a *network*) can comprise multiple disjoint regions that exhibit shared activity patterns across a range of tasks. For many cognitive tasks, a region marks only the approximate location of voxel populations involved in a cognitive task. *Functional* ROIs (fROIs) are voxels within these broad regions that respond most strongly to a given task (language, working memory, *etc*.). The use of fROIs enables accounting for the exact anatomical locations of these task-sensitive voxels, which vary across individuals.

Multiple Demand (MD) system. Located in the prefrontal and parietal areas of the brain, this system of regions is active in a host of tasks requiring working memory and general problem solving skills, including math and logic [Duncan, 2010, Fedorenko et al., 2013, Amalric and Dehaene, 2019].

Language system. These regions have been identified to respond to both comprehension and production of language across modalities (written, speech, sign language), respond to typologically diverse languages (45 languages, from across 11 language families), form a functionally integrated system, reliably and robustly track linguistic stimuli, and have been shown to be causally important for language [Clark and Cummings, 2003, Fedorenko et al., 2010, Hu et al., 2021, Blank and Fedorenko, 2017, Newman et al., 2015, Ayyash et al., 2021].

Visual system. These regions in the occipital lobe of the brain respond to visual input, ranging from low-level features like lines and edges, through intermediate categories like shapes, letters, and numbers, through higher order structures like faces and scenes. [Hubel and Wiesel, 1959, Polk et al., 2002, Epstein and Kanwisher, 1998, Kanwisher et al., 1997].

Auditory system. The auditory system is located in the superior temporal region of the brain. This region uniquely encodes pitch, speech, and music, but is not involved in high-level language comprehension and production [Norman-Haignere et al., 2015, 2019]. In our experiments pertaining to programming language comprehension, we use the activity seen in the auditory system as a negative control.

B Method - Details

Selecting brain representations. For each trial in the fMRI experiment, stimulus responses in each voxel were extracted from the parameters of a General Linear Model (GLM) fit to the time-varying BOLD signal in which each experimental condition was modeled with a boxcar function convolved with the canonical hemodynamic response function (HRF). For the localizer experiments, conditions were modeled as the entire block. For the Python program comprehension experiment, individual programs were modeled using the period from the onset of the code/sentence problem until the button press. The predictors for the GLM included trial ID (equivalent to problem ID), run number, and motion regressors. The voxels were then filtered using gray-matter masking and (for MD and the Language systems) network localization. Although fMRI measurements return whole brain responses, only a thin layer of cortex dubbed gray matter contains BOLD signal of interest to these analyses. Gray matter voxels were selected using a Bayesian segmentation of the anatomical brain image into standard tissue types, and then returning the set of indices where the posterior gray matter probability exceeds 0.70 [Ashburner et al., 1999]. Next, these sets of voxels were filtered separately for each of the brain regions outlined in Appendix A. For the visual and auditory networks, primary sensory areas were identified using an anatomical atlas [Rolls et al., 2020]. For the MD and the Language systems, voxels were functionally localized as those containing the top 10% of responses to their respective functional localizer tasks, as described in Ivanova et al. [2020]. See Fedorenko et al. [2010] for a discussion of the functional localization approach as it pertains to the language network. GLM modeling and gray matter segmentation was performed using SPM12; functional localization was performed using the toolbox released by Ivanova et al. [2020]. Once voxel responses within each brain region were extracted for each trial of each run, some additional preprocessing was required before finalizing the brain representations, and passing them to downstream models. Due to differences in MRI sensitivity across runs, each of the 12 runs of 6 programs for a given subject appeared with nonuniform mean and variance. In order to normalize these signals, so as to leverage data across a participant's entire scanning session, brain representations were z-transformed within each run to achieve a common scale. In order to avoid data-leakage from this preprocessing step, all downstream analyses were cross-validated via a leave-one-run-out approach, such that no intra-run rescaling could be used for prediction.

MVPA cross-validation and hyperparameters. For each brain system and each code property or code model, we run a separate MVPA analysis. For each subject, we train a separate ridge regression model on each cross-validated leave-one-run-out fold, and then predict the remaining targets using the brain representations from the left-out run. We use L2 regularization to control for model complexity, and employ a leave-one-out cross validation scheme to select λ. We score resulting predictions using a series of different metrics to maximize interpretability. For the categorical code properties, we report classification accuracy of model estimates. For the continuously scaled code properties, we report Pearson correlation between the model estimates and true targets. For the model representations, we report rank accuracy between model representation predictions and the true model representations of left-out samples. We calculate rank accuracy as the percentile of a predicted and true target pairing within the full set of possible pairings sorted by a Euclidean distance metric. Rank accuracy was chosen for this experiment as it returns a more interpretable 50% baseline through which to evaluate the brain to model mappings. Following metric calculation on each cross-validation fold, we take the mean of those estimates to report an out-of-sample score over the entire dataset for each subject. We then take the mean of those scores across subjects to derive an overall performance measure. As a baseline, we repeat the above process in its entirety 1000 times for each MVPA analysis, to provide a null permutation distribution against which to compare scores. These null distributions are fit with a univariate Gaussian and used to calculate z-statistics for MVPA scores. For each statistical comparison, a single-tail one-sample z-test is performed between the MVPA score and the distribution fit to the null permutations. Each of these p-values is then FDR-corrected for the set of comparisons in a given experiment, e.g., 4 brain networks $*$ 9 code models = 36 comparisons for experiment 2 (Figure 3). Following FDR-correction, the significance threshold for the resulting p-value is set at $alpha = 0.001$, so decoding is deemed significant at $p < 0.001$. The code to run our MVPA analysis, and all resulting statistics, is available in `https://github.com/anonmyous-author/anonymous-code`.

Code properties. The code properties we consider are the following –

- ***Token count.*** The total number of tokens that appear in the program. This is a sanity check to measure brain activity against, since it is natural to expect the activity in the brain to be correlated

to the length of a program, and in general, the amount of content being comprehended. By their design, the dataset had program lengths with a small standard deviation.

- ***Node count.*** The total number of AST nodes that appear in a program. Similar to the number of tokens, we verify if brain activity correlates to a proxy for the amount of syntactic content in it.
- ***Runtime steps.*** We execute the programs and measure the number of instructions the program steps through. While two programs can have the same program length, the number of instructions executed can differ (*e.g.*`for 1:10` vs. `for 1:50`). We measure if any brain regions capture the number of *mental operations* needed to compute the output of a program.
- ***Bytecode ops.*** We execute the programs and count the number of bytecode operations performed. This metric should mimic the number of *mental operations* performed needed to compute the output of a program.
- ***Cyclomatic complexity.*** This metric [McCabe, 1976] is used to measure the general complexity of software systems. It is defined as a function of the number of nodes, edges, and the connected components in a program's control flow graph. While its efficacy as a metric to measure software complexity has been contested [Shepperd, 1988], we include it to see if brain activity is correlated to any explicit syntactic constructs of a program.
- ***Halstead difficulty.*** This metric [Halstead, 1977] was defined to measure how difficult any piece of software would be for a programmer to comprehend or write. It is defined as a function of the number of tokens, operations, vocabulary that appears in a program.

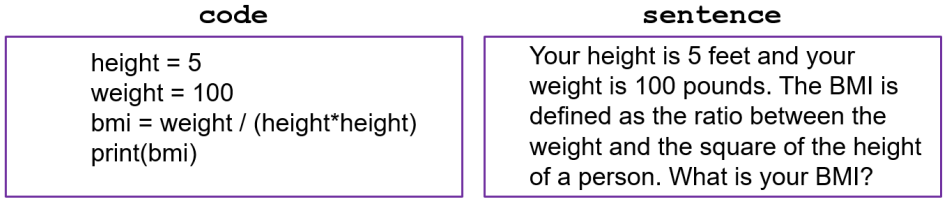

Figure 4: An example of a code problem and its sentence equivalent from Ivanova et al. [2020]

The inter-correlations between these metrics have been tabulated in Table 18, Appendix G.

Code models - Model details. We evaluate the following models in this work–

- ***bag-of-words***, ***TF-IDF***. These are count-based language models. They predict the likelihood of a token appearing in a program based on vocabulary statistics and frequencies.
- ***seq2seq*** [Sutskever et al., 2014], ***CodeTransformer*** [Zügner et al., 2021], ***CodeBERT*** [Feng et al., 2020], ***CodeBERTa*** [HuggingFace, 2020]. We evaluate these three autoencoder (AE) models with increasing model complexity. AE based pretraining reconstructs the original program from corrupted input.
- ***XLNet*** [Yang et al., 2019], ***CodeGPT*** [Microsoft, 2021]. We evaluate auto-regressive (AR) models with a model complexity similar to that of the other transformers. AR language modeling estimates the probability distribution of a text corpus in one direction–either forward or backward, while AE models capture dependencies in both directions. We use AR models to mimic a top-down, single pass comprehension style by humans.
- ***Token Projection***. We compare the results of the above models against an aggressive baseline provided by using unique Gaussian-distributed random vectors for the token embeddings in a vocabulary, and returning the sum of these token embeddings across a program. The resultant embedding is not transformed by any model or any weights–it instead serves as a proxy for the tokens that appear in the program. This sets a higher bar than the null-distribution labeling baseline (Section 5.2).

Code models - Configuration details. In setting up our code model bench, we aimed to select a collection of models ranging in their complexity, namely *bag-of-words*, *TF-IDF* (Term frequency-Inverse document frequency), *seq2seq*, *XLNet*, *CodeTransformer*, *CodeGPT*, *CodeBERT*, and *CodeBERTa*, as well as a *Token Projection* model.

- The *bag-of-words*, *TF-IDF*, *seq2seq*, and *Token Projection* models all use the same custom tokenizer defined by the current authors, whereas *XLNet*, *CodeTransformer*, *CodeGPT*, *CodeBERT*, and *CodeBERTa* use tokenizers defined by the original training authors [Zügner et al., 2021, Microsoft, 2021, Feng et al., 2020, HuggingFace, 2020].

- The tokenizer set up for the models trained by the current authors establishes a unique token in the vocabulary for each Python keyword, each Python builtin function, one token for all numeric types, one token for all string types, and N tokens for each of the N variables in a given program.
- In the case of the *Token Projection* baseline, this tokenizer was used to map each token in a piece of source code to an index in the vocabulary, which in turn was used to index a Gaussian distributed random matrix ($D = 128$). The sum of these random token projections was calculated to return a unique embedding for each program.
- For *bag-of-words* and *TF-IDF*, the validation split of the *code-search-net* Python dataset [Husain et al., 2019] was used to enumerate vocabulary and token occurrence statistics. These data were then used to transform each program to a vector where each dimension represented the raw (*bag-of-words*) or document-weighted (*TF-IDF*) count of the unique items in the vocabulary.
- If a given set of programs viewed by a subject never included a specific token, leading that column to equal the zero vector, then those dimensions would be filtered out.
- We used the dataset released by Project Codenet [Puri et al., 2021] to pretrain a *seq2seq* model [Sutskever et al., 2014]. The dataset contains Python programs that are solutions to olympiad-style programming problems in data structures and algorithms. We trained a seq2seq model with a GRU unit for 15 epochs, with a dropout probability of 0.2, dot product attention, and a maximum sentence length of 500.
- For *XLNet*, *CodeTransformer*, *CodeGPT*, *CodeBERT*, and *CodeBERTa*, each program was passed through that model's tokenizer and model pipeline as defined by the original authors. The representations of each program were extracted from the pretrained encoders. *XLNet*, *CodeTransformer*, and *CodeGPT* were originally trained on the Python subset of the *code-search-net* dataset, whereas *CodeBERT* and *CodeBERTa* were originally trained on the full *code-search-net* dataset, which also incorporates *go*, *java*, *javascript*, *php*, and *ruby*.

C Additional Results

C.1 Experiment 1

Using representations from localized brain regions, we attempt to decode static and dynamic properties of comprehended code, and learn maps to code model representations of that same code. We report decoding performance of each brain region to the original Ivanova et al. [2020] code properties in Table 1, the complexity-related code properties in Table 2, and the code model mappings in Table 3. We find that the MD system and the Language system decode all code properties and code models except *variable language* significantly above baseline, as established through a null permutation test. The Visual system decodes 4 of the 6 code properties and 8 of the 9 code models, whereas the Auditory system only decodes the *code vs sentences* property.

Brain Representation Code Properties	Empirical Baseline	MD	Language	Visual	Auditory
Code vs. Sentence	0.55	0.88 (+0.33)	0.87 (+0.32)	0.88 (+0.33)	0.64 (+0.09)
Control Flow	0.33	0.49 (+0.16)	0.45 (+0.12)	0.39 (+0.06)	0.36 (+0.03)
Data Type	0.49	0.63 (+0.14)	0.58 (+0.09)	0.61 (+0.12)	0.53 (+0.04)
Variable Language	0.50	0.53 (+0.03)	0.51 (+0.01)	0.53 (+0.03)	0.50 (-0.00)

Table 1: Brain region decoding performance on original Ivanova et al. [2020] code properties. Scores represent classification accuracy and are contrasted with an empirical baseline from the null permutation analysis. Values in parentheses are units above baseline.

Brain Representation Code Properties	MD	Language	Visual	Auditory
Static Analysis	0.17	0.24	0.09	0.00
Dynamic Analysis	0.33	0.20	0.17	0.12

Table 2: Brain region decoding performance on *static analysis* and *dynamic analysis* code properties. Scores represent Pearson correlation between predicted and true code properties.

C.2 Experiment 2

Brain Representation Code Models	Empirical Baseline	MD	Language	Visual	Auditory
Token Projection	0.50	0.56 (+0.06)	0.55 (+0.05)	0.54 (+0.04)	0.53 (+0.03)
CodeBERTa	0.50	0.60 (+0.10)	0.57 (+0.07)	0.58 (+0.08)	0.53 (+0.03)
CodeTransformer	0.50	0.59 (+0.09)	0.56 (+0.06)	0.55 (+0.05)	0.52 (+0.02)
CodeBERT	0.50	0.57 (+0.07)	0.56 (+0.06)	0.57 (+0.07)	0.52 (+0.02)
CodeGPT	0.50	0.57 (+0.07)	0.57 (+0.07)	0.57 (+0.07)	0.52 (+0.02)
XLNet	0.50	0.57 (+0.07)	0.55 (+0.05)	0.54 (+0.04)	0.52 (+0.02)
Seq2Seq	0.50	0.57 (+0.07)	0.54 (+0.04)	0.51 (+0.01)	0.51 (+0.01)
TF-IDF	0.50	0.57 (+0.07)	0.54 (+0.04)	0.53 (+0.03)	0.52 (+0.02)
Bag Of Words	0.50	0.58 (+0.08)	0.56 (+0.06)	0.54 (+0.04)	0.52 (+0.02)

Table 3: Brain region decoding performance on code model mappings. Scores represent rank accuracy and are contrasted with an empirical baseline from the null permutation analysis. Values in parentheses are units above baseline.

We additionally evaluate whether code models contain linearly decodable information about the code properties we explore in this work (Section 4.2). The models decode with near perfect accuracy on most classification tasks (avg. accuracy= 0.97 ± 0.03), and with high correlations on the regression tasks (avg. $r = 0.89 \pm 0.05$). This is expected for the classification tasks, since the dataset contains tokens which help with perfectly separable decision boundaries. For example, for the *control flow* code property, the dataset contains programs with `if`, `for`, or neither, but never both. These results suggest that the code models we evaluate faithfully encode the set of properties we evaluate them on.

Code Properties Empirical Baseline Model Representation	Control Flow 0.31	Data Type 0.48	Dynamic Analysis 0.00	Static Analysis 0.00
Token Projection	1.00	0.94	0.88	1.00
CodeBERTa	0.98	1.00	0.87	0.86
CodeTransformer	1.00	0.95	0.90	0.91
CodeBERT	1.00	0.99	0.89	0.92
CodeGPT	0.98	1.00	0.84	0.82
XLNet	0.92	0.92	0.76	0.88
Seq2Seq	0.95	0.97	0.87	0.91
TF-IDF	1.00	0.94	0.94	0.89
Bag Of Words	1.00	0.92	0.93	0.98

Table 4: Decoding performance of all models on all tasks. Scores represent classification accuracy for *control flow* and *data type*, and Pearson correlation for the remaining benchmarks. These can be contrasted with an empirical baseline from the null permutation analysis.

D Analysis of Variance

We run a series of one-way ANOVA statistical tests to assess whether decoding performance varies across brain regions for each code property and code model, and whether decoding performance varies across code models for each brain region. We report these results in Tables 5, 6, and 7. We find that decoding performance varies across brain region for all code properties and code models, except the *variable language* property and the *Token Projection* model, and decoding performance varies across code models for the MD system and the Visual system.

Code Property	F	p	p (corrected)	Is Significant?
Code vs. Sentence	53.57	4.03e-20	2.42e-19	1
Control Flow	12.52	6.13e-07	1.84e-06	1
Static Analysis	6.71	3.82e-04	7.65e-04	1
Data Type	4.46	5.66e-03	8.50e-03	1
Dynamic Analysis	4.15	8.38e-03	1.01e-02	1
Variable Language	0.82	4.84e-01	4.84e-01	0

Table 5: Results from statistical testing of variance across brain regions for each code property.

Code Model	F	p	p (corrected)	Is Significant?
Bag Of Words	5.66	1.33e-03	1.71e-03	1
CodeBERT	8.73	3.72e-05	9.75e-05	1
CodeBERTa	12.48	6.40e-07	5.60e-06	1
CodeGPT	7.00	2.73e-04	4.09e-04	1
CodeTransformer	11.85	1.25e-06	5.60e-06	1
Seq2Seq	8.40	5.42e-05	9.75e-05	1
TF-IDF	8.46	5.08e-05	9.75e-05	1
XLNet	3.48	1.90e-02	2.14e-02	1
Token Projection	2.32	8.04e-02	8.04e-02	0

Table 6: Results from statistical testing of variance across brain regions for each code model.

Brain Region	F	p	p (corrected)	Is Significant?
Language	2.49	1.35e-02	2.70e-02	1
Visual	5.13	7.61e-06	3.05e-05	1
Auditory	0.30	9.66e-01	9.66e-01	0
MD	1.79	7.97e-02	1.06e-01	0

Table 7: Results from statistical testing of variance across code models for each brain region.

E Pairwise Analysis

To extend the findings from the ANOVA analyses towards specific pairwise comparisons between brain regions and code models, we compare scores between brain regions for each code property or code model, and compare scores between code models and a subset of code properties for each brain region, using paired two-tailed t-tests. We report all pairwise brain region comparisons in Tables 8 and 10, and report all pairwise code property and model comparisons in Tables 9 and 11. To highlight a few key results from code properties, we find that the MD system decodes *dynamic analysis* significantly better than the LS (Table 8). Moving onto code models, we find that the MD system decodes *CodeBERTa*, *CodeTransformer*, *seq2seq* and *TF-IDF* significantly more accurately than the Language system (Table 10). Finally, an investigation into model complexity reveals that *CodeBERTa*, *CodeTransformer*, and *bag-of-words* are significantly more accurately decoded from the MD system than the *Token Projection* model.

Code Property	Brain Region A	Brain Region B	t	p	p (corrected)	Is Significant?
Code vs. Sentence	Language	Auditory	8.21	2.74e-08	3.29e-07	1
Code vs. Sentence	MD	Auditory	10.42	3.52e-10	6.33e-09	1
Code vs. Sentence	Visual	Auditory	10.52	2.92e-10	6.33e-09	1
Control Flow	Language	Auditory	3.90	7.20e-04	3.24e-03	1
Control Flow	MD	Auditory	4.86	6.63e-05	5.50e-04	1
Control Flow	MD	Visual	4.22	3.26e-04	1.96e-03	1
Data Type	Language	Auditory	2.50	1.98e-02	4.75e-02	1
Data Type	MD	Auditory	2.99	6.61e-03	2.16e-02	1
Data Type	Visual	Auditory	3.59	1.56e-03	6.23e-03	1
Dynamic Analysis	MD	Auditory	2.93	7.59e-03	2.28e-02	1
Dynamic Analysis	MD	Language	2.65	1.44e-02	3.98e-02	1
Dynamic Analysis	MD	Visual	2.53	1.86e-02	4.75e-02	1
Static Analysis	Language	Auditory	4.80	7.64e-05	5.50e-04	1
Static Analysis	Language	Visual	4.06	4.89e-04	2.52e-03	1
Static Analysis	MD	Auditory	3.35	2.80e-03	1.01e-02	1
Code vs. Sentence	Language	Visual	-0.39	6.99e-01	7.40e-01	0
Code vs. Sentence	MD	Language	0.59	5.64e-01	6.34e-01	0
Code vs. Sentence	MD	Visual	0.27	7.92e-01	7.92e-01	0
Control Flow	Language	Visual	2.43	2.34e-02	5.26e-02	0
Control Flow	MD	Language	1.75	9.36e-02	1.87e-01	0
Control Flow	Visual	Auditory	1.44	1.62e-01	2.57e-01	0
Data Type	Language	Visual	-1.03	3.15e-01	4.36e-01	0
Data Type	MD	Language	1.56	1.33e-01	2.53e-01	0
Data Type	MD	Visual	0.71	4.88e-01	5.93e-01	0
Dynamic Analysis	Language	Auditory	1.26	2.22e-01	3.19e-01	0
Dynamic Analysis	Language	Visual	0.52	6.06e-01	6.61e-01	0
Dynamic Analysis	Visual	Auditory	0.67	5.11e-01	5.93e-01	0
Static Analysis	MD	Language	-1.45	1.61e-01	2.57e-01	0
Static Analysis	MD	Visual	1.76	9.10e-02	1.87e-01	0
Static Analysis	Visual	Auditory	1.44	1.64e-01	2.57e-01	0
Variable Language	Language	Auditory	0.67	5.11e-01	5.93e-01	0
Variable Language	Language	Visual	-0.94	3.55e-01	4.56e-01	0
Variable Language	MD	Auditory	1.46	1.59e-01	2.57e-01	0
Variable Language	MD	Language	0.98	3.36e-01	4.48e-01	0
Variable Language	MD	Visual	0.32	7.49e-01	7.70e-01	0
Variable Language	Visual	Auditory	1.35	1.89e-01	2.84e-01	0

Table 8: Results from paired two-tailed t-tests of brain regions for each code property. $+t$ reflects $A > B$, whereas $-t$ reflects $A < B$.

Brain Region	Code Property A	Code Property B	t	p	p (corrected)	Is Significant?
MD	Static Analysis	Dynamic Analysis	-2.72	1.21e-02	4.83e-02	1
Auditory	Static Analysis	Dynamic Analysis	-1.96	6.27e-02	1.25e-01	0
Language	Static Analysis	Dynamic Analysis	0.79	4.39e-01	4.39e-01	0
Visual	Static Analysis	Dynamic Analysis	-1.33	1.97e-01	2.63e-01	0

Table 9: Results from paired two-tailed t-tests of continuous code metrics across brain regions. Only this subset of properties was selected so as to evaluate scores with a consistent metric and baseline. $+t$ reflects $A > B$, whereas $-t$ reflects $A < B$.

Code Model	Brain Region A	Brain Region B	t	p	p (corrected)	Is Significant?
Bag Of Words	Language	Auditory	2.84	9.22e-03	2.36e-02	1
Bag Of Words	MD	Auditory	4.19	3.54e-04	2.73e-03	1
Bag Of Words	MD	Visual	2.50	1.98e-02	4.28e-02	1
CodeBERT	Language	Auditory	3.64	1.36e-03	4.77e-03	1
CodeBERT	MD	Auditory	3.77	9.97e-04	4.14e-03	1
CodeBERT	Visual	Auditory	5.59	1.09e-05	2.95e-04	1
CodeBERTa	Language	Auditory	3.63	1.41e-03	4.77e-03	1
CodeBERTa	MD	Auditory	4.82	7.26e-05	9.80e-04	1
CodeBERTa	MD	Language	2.69	1.30e-02	3.05e-02	1
CodeBERTa	Visual	Auditory	3.66	1.30e-03	4.77e-03	1
CodeGPT	Language	Auditory	3.89	7.41e-04	4.00e-03	1
CodeGPT	MD	Auditory	4.07	4.73e-04	2.84e-03	1
CodeGPT	Visual	Auditory	3.81	9.10e-04	4.14e-03	1
CodeTransformer	Language	Auditory	4.21	3.37e-04	2.73e-03	1
CodeTransformer	MD	Auditory	5.31	2.17e-05	3.91e-04	1
CodeTransformer	MD	Language	2.44	2.29e-02	4.59e-02	1
CodeTransformer	MD	Visual	3.10	5.05e-03	1.44e-02	1
CodeTransformer	Visual	Auditory	2.89	8.36e-03	2.26e-02	1
Seq2Seq	MD	Auditory	4.11	4.26e-04	2.84e-03	1
Seq2Seq	MD	Language	2.82	9.63e-03	2.36e-02	1
Seq2Seq	MD	Visual	4.58	1.33e-04	1.44e-03	1
TF-IDF	Language	Auditory	2.54	1.85e-02	4.16e-02	1
TF-IDF	MD	Auditory	5.77	7.03e-06	2.95e-04	1
TF-IDF	MD	Language	3.25	3.51e-03	1.05e-02	1
TF-IDF	MD	Visual	3.79	9.52e-04	4.14e-03	1
Token Projection	MD	Auditory	2.46	2.20e-02	4.56e-02	1
XLNet	MD	Auditory	3.31	3.09e-03	9.82e-03	1
XLNet	MD	Visual	2.41	2.45e-02	4.73e-02	1
Bag Of Words	Language	Visual	1.48	1.52e-01	2.32e-01	0
Bag Of Words	MD	Language	1.51	1.44e-01	2.29e-01	0
Bag Of Words	Visual	Auditory	0.95	3.54e-01	4.34e-01	0
CodeBERT	Language	Visual	-0.73	4.74e-01	5.45e-01	0
CodeBERT	MD	Language	0.82	4.22e-01	4.95e-01	0
CodeBERT	MD	Visual	0.31	7.56e-01	7.85e-01	0
CodeBERTa	Language	Visual	-1.28	2.13e-01	2.88e-01	0
CodeBERTa	MD	Visual	1.36	1.86e-01	2.64e-01	0
CodeGPT	Language	Visual	0.06	9.53e-01	9.53e-01	0
CodeGPT	MD	Language	0.54	5.92e-01	6.53e-01	0
CodeGPT	MD	Visual	0.63	5.38e-01	6.05e-01	0
CodeTransformer	Language	Visual	0.88	3.86e-01	4.64e-01	0
Seq2Seq	Language	Auditory	1.82	8.19e-02	1.42e-01	0
Seq2Seq	Language	Visual	2.23	3.57e-02	6.64e-02	0
Seq2Seq	Visual	Auditory	0.14	8.91e-01	9.08e-01	0
TF-IDF	Language	Visual	1.04	3.11e-01	3.90e-01	0
TF-IDF	Visual	Auditory	1.34	1.94e-01	2.69e-01	0
Token Projection	Language	Auditory	1.81	8.39e-02	1.42e-01	0
Token Projection	Language	Visual	0.37	7.15e-01	7.72e-01	0
Token Projection	MD	Language	1.09	2.85e-01	3.66e-01	0
Token Projection	MD	Visual	1.11	2.79e-01	3.66e-01	0
Token Projection	Visual	Auditory	1.38	1.80e-01	2.62e-01	0
XLNet	Language	Auditory	1.58	1.27e-01	2.09e-01	0
XLNet	Language	Visual	0.34	7.35e-01	7.79e-01	0
XLNet	MD	Language	1.98	6.03e-02	1.09e-01	0
XLNet	Visual	Auditory	1.47	1.55e-01	2.32e-01	0

Table 10: Results from paired two-tailed t-tests of brain regions for each code model. $+t$ reflects $A > B$, whereas $-t$ reflects $A < B$.

Brain Region	Code Model A	Code Model B	t	p	p (corrected)	Is Significant?
Language	CodeBERTa	Seq2Seq	3.36	2.71e-03	2.94e-02	1
Language	CodeBERTa	TF-IDF	4.66	1.09e-04	3.15e-03	1
Language	Token Projection	CodeBERTa	-4.16	3.83e-04	6.89e-03	1
MD	Token Projection	Bag Of Words	-3.79	9.39e-04	1.23e-02	1
MD	Token Projection	CodeBERTa	-4.02	5.38e-04	8.60e-03	1
MD	Token Projection	CodeTransformer	-3.21	3.92e-03	3.53e-02	1
Visual	CodeBERT	Seq2Seq	6.25	2.22e-06	2.45e-04	1
Visual	CodeBERT	TF-IDF	4.57	1.36e-04	3.15e-03	1
Visual	CodeBERTa	Bag Of Words	3.83	8.57e-04	1.23e-02	1
Visual	CodeBERTa	CodeTransformer	3.06	5.55e-03	4.71e-02	1
Visual	CodeBERTa	Seq2Seq	6.07	3.40e-06	2.45e-04	1
Visual	CodeBERTa	TF-IDF	4.88	6.21e-05	2.24e-03	1
Visual	CodeBERTa	XLNet	3.34	2.86e-03	2.94e-02	1
Visual	CodeGPT	Seq2Seq	4.52	1.53e-04	3.15e-03	1
Visual	CodeGPT	TF-IDF	3.27	3.39e-03	3.26e-02	1
Visual	CodeTransformer	Seq2Seq	3.68	1.24e-03	1.48e-02	1
Visual	Token Projection	CodeBERTa	-5.20	2.82e-05	1.36e-03	1
Auditory	CodeBERT	Bag Of Words	-0.37	7.17e-01	8.96e-01	0
Auditory	CodeBERT	CodeGPT	0.12	9.08e-01	9.73e-01	0
Auditory	CodeBERT	Seq2Seq	0.80	4.33e-01	6.93e-01	0
Auditory	CodeBERT	TF-IDF	0.10	9.23e-01	9.73e-01	0
Auditory	CodeBERT	XLNet	-0.50	6.22e-01	8.54e-01	0
Auditory	CodeBERTa	Bag Of Words	0.14	8.94e-01	9.73e-01	0
Auditory	CodeBERTa	CodeBERT	0.90	3.77e-01	6.54e-01	0
Auditory	CodeBERTa	CodeGPT	0.97	3.43e-01	6.33e-01	0
Auditory	CodeBERTa	CodeTransformer	0.92	3.66e-01	6.54e-01	0
Auditory	CodeBERTa	Seq2Seq	1.15	2.62e-01	5.39e-01	0
Auditory	CodeBERTa	TF-IDF	0.60	5.56e-01	8.01e-01	0
Auditory	CodeBERTa	XLNet	0.20	8.41e-01	9.44e-01	0
Auditory	CodeGPT	Bag Of Words	-0.40	6.94e-01	8.84e-01	0
Auditory	CodeGPT	Seq2Seq	0.71	4.83e-01	7.32e-01	0
Auditory	CodeGPT	TF-IDF	0.05	9.61e-01	9.83e-01	0
Auditory	CodeGPT	XLNet	-0.54	5.95e-01	8.32e-01	0
Auditory	CodeTransformer	Bag Of Words	-0.74	4.69e-01	7.18e-01	0
Auditory	CodeTransformer	CodeBERT	-0.40	6.90e-01	8.84e-01	0
Auditory	CodeTransformer	CodeGPT	-0.36	7.22e-01	8.96e-01	0
Auditory	CodeTransformer	Seq2Seq	0.31	7.58e-01	9.02e-01	0
Auditory	CodeTransformer	TF-IDF	-0.32	7.51e-01	9.02e-01	0
Auditory	CodeTransformer	XLNet	-1.07	2.98e-01	5.90e-01	0
Auditory	Seq2Seq	Bag Of Words	-0.88	3.86e-01	6.54e-01	0
Auditory	Seq2Seq	TF-IDF	-0.58	5.65e-01	8.06e-01	0
Auditory	TF-IDF	Bag Of Words	-0.61	5.50e-01	8.01e-01	0
Auditory	Token Projection	Bag Of Words	0.16	8.77e-01	9.64e-01	0
Auditory	Token Projection	CodeBERT	0.47	6.39e-01	8.60e-01	0
Auditory	Token Projection	CodeBERTa	-0.07	9.47e-01	9.83e-01	0
Auditory	Token Projection	CodeGPT	0.52	6.11e-01	8.46e-01	0
Auditory	Token Projection	CodeTransformer	0.90	3.77e-01	6.54e-01	0
Auditory	Token Projection	Seq2Seq	1.01	3.21e-01	6.08e-01	0
Auditory	Token Projection	TF-IDF	0.75	4.60e-01	7.15e-01	0
Auditory	Token Projection	XLNet	0.11	9.14e-01	9.73e-01	0
Auditory	XLNet	Bag Of Words	-0.00	1.00e+00	1.00e+00	0
Auditory	XLNet	Seq2Seq	1.06	2.99e-01	5.90e-01	0
Auditory	XLNet	TF-IDF	0.69	4.98e-01	7.43e-01	0
Language	CodeBERT	Bag Of Words	0.33	7.44e-01	9.01e-01	0
Language	CodeBERT	CodeGPT	-0.19	8.53e-01	9.45e-01	0
Language	CodeBERT	Seq2Seq	2.29	3.14e-02	1.13e-01	0
Language	CodeBERT	TF-IDF	2.53	1.87e-02	9.61e-02	0
Language	CodeBERT	XLNet	2.04	5.29e-02	1.52e-01	0
Language	CodeBERTa	Bag Of Words	1.37	1.83e-01	4.11e-01	0
Language	CodeBERTa	CodeBERT	1.54	1.37e-01	3.41e-01	0
Language	CodeBERTa	CodeGPT	0.89	3.84e-01	6.54e-01	0
Language	CodeBERTa	CodeTransformer	1.41	1.72e-01	3.99e-01	0
Language	CodeBERTa	XLNet	2.81	9.99e-03	7.17e-02	0
Language	CodeGPT	Bag Of Words	0.45	6.60e-01	8.80e-01	0
Language	CodeGPT	Seq2Seq	2.73	1.19e-02	7.17e-02	0
Language	CodeGPT	TF-IDF	2.60	1.59e-02	8.80e-02	0
Language	CodeGPT	XLNet	2.24	3.47e-02	1.16e-01	0
Language	CodeTransformer	Bag Of Words	0.22	8.31e-01	9.44e-01	0
Language	CodeTransformer	CodeBERT	-0.21	8.36e-01	9.44e-01	0
Language	CodeTransformer	CodeGPT	-0.36	7.22e-01	8.96e-01	0
Language	CodeTransformer	Seq2Seq	2.46	2.17e-02	1.04e-01	0
Language	CodeTransformer	TF-IDF	2.63	1.50e-02	8.61e-02	0
Language	CodeTransformer	XLNet	1.95	6.36e-02	1.76e-01	0
Language	Seq2Seq	Bag Of Words	-2.33	2.90e-02	1.10e-01	0
Language	Seq2Seq	TF-IDF	-0.62	5.43e-01	7.98e-01	0
Language	TF-IDF	Bag Of Words	-2.02	5.49e-02	1.55e-01	0
Language	Token Projection	Bag Of Words	-2.30	3.12e-02	1.13e-01	0
Language	Token Projection	CodeBERT	-2.16	4.16e-02	1.24e-01	0
Language	Token Projection	CodeGPT	-2.19	3.85e-02	1.22e-01	0
Language	Token Projection	CodeTransformer	-2.18	3.98e-02	1.22e-01	0
Language	Token Projection	Seq2Seq	0.93	3.60e-01	6.54e-01	0
Language	Token Projection	TF-IDF	0.42	6.80e-01	8.83e-01	0
Language	Token Projection	XLNet	0.00	1.00e+00	1.00e+00	0
Language	XLNet	Bag Of Words	-1.48	1.52e-01	3.68e-01	0
Language	XLNet	Seq2Seq	0.84	4.10e-01	6.70e-01	0
Language	XLNet	TF-IDF	0.35	7.30e-01	8.97e-01	0
MD	CodeBERT	Bag Of Words	-0.43	6.68e-01	8.83e-01	0
MD	CodeBERT	CodeGPT	0.11	9.10e-01	9.73e-01	0
MD	CodeBERT	Seq2Seq	0.49	6.32e-01	8.58e-01	0
MD	CodeBERT	TF-IDF	0.34	7.35e-01	8.97e-01	0
MD	CodeBERT	XLNet	0.29	7.76e-01	9.09e-01	0
MD	CodeBERTa	Bag Of Words	1.74	9.55e-02	2.50e-01	0
MD	CodeBERTa	CodeBERT	2.74	1.16e-02	7.17e-02	0
MD	CodeBERTa	CodeGPT	2.95	7.26e-03	5.50e-02	0
MD	CodeBERTa	CodeTransformer	0.85	4.03e-01	6.70e-01	0
MD	CodeBERTa	Seq2Seq	2.42	2.38e-02	1.06e-01	0
MD	CodeBERTa	TF-IDF	2.99	6.49e-03	5.19e-02	0
MD	CodeBERTa	XLNet	2.37	2.65e-02	1.06e-01	0
MD	CodeGPT	Bag Of Words	-0.56	5.81e-01	8.20e-01	0
MD	CodeGPT	Seq2Seq	0.42	6.81e-01	8.83e-01	0
MD	CodeGPT	TF-IDF	0.29	7.77e-01	9.09e-01	0
MD	CodeGPT	XLNet	0.21	8.37e-01	9.44e-01	0
MD	CodeTransformer	Bag Of Words	1.03	3.14e-01	6.08e-01	0
MD	CodeTransformer	CodeBERT	1.43	1.65e-01	3.91e-01	0
MD	CodeTransformer	CodeGPT	2.19	3.90e-02	1.22e-01	0
MD	CodeTransformer	Seq2Seq	1.87	7.45e-02	1.99e-01	0
MD	CodeTransformer	TF-IDF	2.36	2.73e-02	1.06e-01	0
MD	CodeTransformer	XLNet	2.40	2.48e-02	1.06e-01	0
MD	Seq2Seq	Bag Of Words	-0.91	3.74e-01	6.54e-01	0
MD	Seq2Seq	TF-IDF	-0.22	8.29e-01	9.44e-01	0
MD	TF-IDF	Bag Of Words	-1.20	2.43e-01	5.06e-01	0
MD	Token Projection	CodeBERT	-1.28	2.14e-01	4.60e-01	0
MD	Token Projection	CodeGPT	-1.38	1.81e-01	4.11e-01	0
MD	Token Projection	Seq2Seq	-0.77	4.50e-01	7.12e-01	0
MD	Token Projection	TF-IDF	-1.48	1.53e-01	3.68e-01	0
MD	Token Projection	XLNet	-1.21	2.39e-01	5.06e-01	0
MD	XLNet	Bag Of Words	-0.84	4.09e-01	6.70e-01	0
MD	XLNet	Seq2Seq	0.20	8.45e-01	9.44e-01	0
MD	XLNet	TF-IDF	0.06	9.49e-01	9.83e-01	0
Visual	CodeBERT	Bag Of Words	2.39	2.53e-02	1.06e-01	0
Visual	CodeBERT	CodeGPT	0.68	5.00e-01	7.43e-01	0
Visual	CodeBERT	XLNet	2.73	1.20e-02	7.17e-02	0
Visual	CodeBERTa	CodeBERT	1.36	1.87e-01	4.15e-01	0
Visual	CodeBERTa	CodeGPT	2.45	2.24e-02	1.04e-01	0
Visual	CodeGPT	Bag Of Words	2.19	3.93e-02	1.22e-01	0
Visual	CodeGPT	XLNet	2.26	3.37e-02	1.16e-01	0
Visual	CodeTransformer	Bag Of Words	1.07	2.95e-01	5.90e-01	0
Visual	CodeTransformer	CodeBERT	-1.63	1.16e-01	2.98e-01	0
Visual	CodeTransformer	CodeGPT	-0.98	3.35e-01	6.27e-01	0
Visual	CodeTransformer	TF-IDF	2.50	2.00e-02	9.95e-02	0
Visual	CodeTransformer	XLNet	1.57	1.30e-01	3.28e-01	0
Visual	Seq2Seq	Bag Of Words	-2.15	4.22e-02	1.24e-01	0
Visual	Seq2Seq	TF-IDF	-2.39	2.57e-02	1.06e-01	0
Visual	TF-IDF	Bag Of Words	-0.80	4.30e-01	6.93e-01	0
Visual	Token Projection	Bag Of Words	0.09	9.25e-01	9.73e-01	0
Visual	Token Projection	CodeBERT	-2.73	1.18e-02	7.17e-02	0
Visual	Token Projection	CodeGPT	-2.28	3.23e-02	1.14e-01	0
Visual	Token Projection	CodeTransformer	-1.35	1.91e-01	4.16e-01	0
Visual	Token Projection	Seq2Seq	2.55	1.77e-02	9.46e-02	0
Visual	Token Projection	TF-IDF	1.02	3.20e-01	6.08e-01	0
Visual	Token Projection	XLNet	0.05	9.62e-01	9.83e-01	0
Visual	XLNet	Bag Of Words	0.02	9.83e-01	9.97e-01	0
Visual	XLNet	Seq2Seq	1.89	7.11e-02	1.93e-01	0
Visual	XLNet	TF-IDF	0.75	4.62e-01	7.15e-01	0

Table 11: Results from paired two-tailed t-tests of code models for each brain region. $+t$ reflects $A > B$, whereas $-t$ reflects $A < B$.

F Multi-system partial regression analysis: Scores and significance

In order to investigate whether each of the brain systems included contribute unique information towards the decoding tasks, we combine brain representations from different systems in paired combinations and evaluate effects on downstream decoding performance across all experiments. We find that the addition of MD or LS to VS, relative to VS alone, improves downstream decoding of *code vs sentences*. The same effect is observed for the combination of MD and LS compared to either alone. These data suggest that the MD system and the Language system encode unique variance relevant to the decoding of *code vs sentences*, and this is above and beyond the information encoded in the Visual system or each other individually. Additionally, we observe that the addition of MD to VS, relative to VS alone, improves downstream decoding of *control flow*, *data type*, *dynamic analysis*, and *static analysis*. This same effect is observed for the addition of LS to Visual system for *control flow* and *static analysis*. These data suggest that the MD system and the Language system encode unique information above and beyond information encoded in the Visual system for these decoding tasks (Table 15). We repeat this process for Experiment 2, where we find that the addition of MD to VS improves decoding for 7 models, the addition of MD to LS improves decoding for 7 models, and the addition of LS to Visual system improves decoding for all 9 models (Table 16).

Brain Representation Code Properties	Empirical Baseline	L+V	MD+L	MD+V
Code vs. Sentence	0.56	0.94 (+0.38)	0.94 (+0.38)	0.93 (+0.37)
Control Flow	0.33	0.45 (+0.12)	0.49 (+0.16)	0.49 (+0.16)
Data Type	0.50	0.62 (+0.12)	0.63 (+0.13)	0.68 (+0.18)
Variable Language	0.50	0.53 (+0.03)	0.53 (+0.03)	0.53 (+0.03)

Table 12: Multi-system partial regression analysis on original Ivanova et al. [2020] code properties. Scores represent classification accuracy and are contrasted with an empirical baseline from the null permutation analysis. Values in parentheses are units above baseline.

Brain Representation Code Properties	L+V	MD+L	MD+V
Dynamic Analysis	0.22	0.32	0.31
Static Analysis	0.19	0.21	0.20

Table 13: Multi-system partial regression analysis on static and dynamic code properties. Scores represent Pearson correlation between predicted and true code properties.

Brain Representation Code Models	Empirical Baseline	L+V	MD+L	MD+V
Token Projection	0.50	0.57 (+0.07)	0.56 (+0.06)	0.57 (+0.07)
CodeBERTa	0.50	0.60 (+0.10)	0.62 (+0.12)	0.62 (+0.12)
CodeTransformer	0.50	0.58 (+0.08)	0.60 (+0.10)	0.60 (+0.10)
CodeBERT	0.50	0.60 (+0.10)	0.59 (+0.09)	0.59 (+0.09)
CodeGPT	0.50	0.59 (+0.09)	0.58 (+0.08)	0.58 (+0.08)
XLNet	0.50	0.57 (+0.07)	0.58 (+0.08)	0.57 (+0.07)
Seq2Seq	0.50	0.55 (+0.05)	0.57 (+0.07)	0.56 (+0.06)
TF-IDF	0.50	0.56 (+0.06)	0.58 (+0.08)	0.57 (+0.07)
Bag Of Words	0.50	0.57 (+0.07)	0.59 (+0.09)	0.59 (+0.09)

Table 14: Multi-system partial regression analysis on code model mappings. Scores represent rank accuracy and are contrasted with an empirical baseline from the null permutation analysis. Values in parentheses are units above baseline.

Code Property	Brain Region A	Brain Region B	t	p	p (corrected)	Is Significant?
Code vs. Sentence	L+V	Language	6.61	9.71e-07	1.75e-05	1
Code vs. Sentence	L+V	Visual	6.94	4.48e-07	1.61e-05	1
Code vs. Sentence	MD+L	Language	5.98	4.22e-06	5.06e-05	1
Code vs. Sentence	MD+L	MD	5.85	5.88e-06	5.30e-05	1
Code vs. Sentence	MD+V	MD	4.17	3.69e-04	1.66e-03	1
Code vs. Sentence	MD+V	Visual	4.71	9.70e-05	5.82e-04	1
Control Flow	L+V	Visual	4.27	2.87e-04	1.48e-03	1
Control Flow	MD+V	Visual	4.72	9.32e-05	5.82e-04	1
Data Type	MD+V	MD	2.88	8.46e-03	2.32e-02	1
Data Type	MD+V	Visual	3.96	6.28e-04	2.51e-03	1
Dynamic Analysis	MD+L	Language	2.85	9.03e-03	2.32e-02	1
Dynamic Analysis	MD+V	Visual	2.89	8.19e-03	2.32e-02	1
Static Analysis	L+V	Visual	3.63	1.39e-03	5.01e-03	1
Static Analysis	MD+V	Visual	3.23	3.73e-03	1.22e-02	1
Control Flow	L+V	Language	0.18	8.56e-01	9.34e-01	0
Control Flow	MD+L	Language	2.22	3.65e-02	8.22e-02	0
Control Flow	MD+L	MD	-0.25	8.07e-01	9.34e-01	0
Control Flow	MD+V	MD	0.06	9.55e-01	1.00e+00	0
Data Type	L+V	Language	1.91	6.81e-02	1.44e-01	0
Data Type	L+V	Visual	0.90	3.77e-01	5.78e-01	0
Data Type	MD+L	Language	2.32	2.97e-02	7.12e-02	0
Data Type	MD+L	MD	0.52	6.11e-01	7.85e-01	0
Dynamic Analysis	L+V	Language	0.42	6.80e-01	8.17e-01	0
Dynamic Analysis	L+V	Visual	1.81	8.34e-02	1.67e-01	0
Dynamic Analysis	MD+L	MD	-0.18	8.57e-01	9.34e-01	0
Dynamic Analysis	MD+V	MD	-0.60	5.52e-01	7.36e-01	0
Static Analysis	L+V	Language	-1.40	1.73e-01	3.29e-01	0
Static Analysis	MD+L	Language	-0.88	3.86e-01	5.78e-01	0
Static Analysis	MD+L	MD	1.34	1.95e-01	3.50e-01	0
Static Analysis	MD+V	MD	1.01	3.22e-01	5.29e-01	0
Variable Language	L+V	Language	1.01	3.24e-01	5.29e-01	0
Variable Language	L+V	Visual	-0.00	1.00e+00	1.00e+00	0
Variable Language	MD+L	Language	0.77	4.52e-01	6.51e-01	0
Variable Language	MD+L	MD	-0.68	5.02e-01	6.95e-01	0
Variable Language	MD+V	MD	-0.44	6.64e-01	8.17e-01	0
Variable Language	MD+V	Visual	0.00	1.00e+00	1.00e+00	0

Table 15: Results from paired two-tailed t-tests of joint brain systems with their partial components for each code property. $+t$ reflects $A > B$, whereas $-t$ reflects $A < B$.

Code Model	Brain Region A	Brain Region B	t	p	p (corrected)	Is Significant?
Bag Of Words	L+V	Visual	4.22	3.29e-04	1.98e-03	1
Bag Of Words	MD+L	Language	2.62	1.52e-02	3.15e-02	1
Bag Of Words	MD+V	Visual	4.02	5.29e-04	2.71e-03	1
CodeBERT	L+V	Language	3.94	6.49e-04	2.71e-03	1
CodeBERT	L+V	Visual	3.74	1.06e-03	3.58e-03	1
CodeBERT	MD+L	Language	2.51	1.97e-02	3.80e-02	1
CodeBERT	MD+L	MD	2.64	1.47e-02	3.15e-02	1
CodeBERTa	L+V	Language	3.94	6.53e-04	2.71e-03	1
CodeBERTa	L+V	Visual	2.81	9.87e-03	2.22e-02	1
CodeBERTa	MD+L	Language	4.26	2.98e-04	1.98e-03	1
CodeBERTa	MD+L	MD	2.44	2.28e-02	3.97e-02	1
CodeBERTa	MD+V	Visual	2.93	7.58e-03	1.79e-02	1
CodeGPT	L+V	Language	2.47	2.14e-02	3.86e-02	1
CodeGPT	L+V	Visual	3.45	2.20e-03	6.61e-03	1
CodeTransformer	L+V	Language	2.39	2.56e-02	4.33e-02	1
CodeTransformer	L+V	Visual	4.25	3.04e-04	1.98e-03	1
CodeTransformer	MD+L	Language	3.96	6.29e-04	2.71e-03	1
CodeTransformer	MD+V	Visual	4.60	1.27e-04	1.45e-03	1
Seq2Seq	L+V	Visual	4.35	2.33e-04	1.98e-03	1
Seq2Seq	MD+L	Language	2.98	6.73e-03	1.79e-02	1
Seq2Seq	MD+V	Visual	4.80	7.70e-05	1.39e-03	1
TF-IDF	L+V	Language	3.10	5.11e-03	1.45e-02	1
TF-IDF	L+V	Visual	5.57	1.14e-05	6.16e-04	1
TF-IDF	MD+L	Language	4.95	5.23e-05	1.39e-03	1
TF-IDF	MD+L	MD	2.54	1.82e-02	3.65e-02	1
TF-IDF	MD+V	Visual	3.84	8.36e-04	3.01e-03	1
Token Projection	L+V	Language	2.34	2.85e-02	4.67e-02	1
Token Projection	L+V	Visual	2.92	7.64e-03	1.79e-02	1
Token Projection	MD+V	Visual	2.96	7.07e-03	1.79e-02	1
XLNet	L+V	Language	2.48	2.11e-02	3.86e-02	1
XLNet	L+V	Visual	4.58	1.34e-04	1.45e-03	1
XLNet	MD+L	Language	3.66	1.30e-03	4.13e-03	1
XLNet	MD+V	Visual	3.85	8.22e-04	3.01e-03	1
Bag Of Words	L+V	Language	0.84	4.10e-01	4.51e-01	0
Bag Of Words	MD+L	MD	1.56	1.32e-01	1.70e-01	0
Bag Of Words	MD+V	MD	0.76	4.55e-01	4.92e-01	0
CodeBERT	MD+V	MD	2.23	3.55e-02	5.48e-02	0
CodeBERT	MD+V	Visual	2.25	3.40e-02	5.40e-02	0
CodeBERTa	MD+V	MD	2.18	3.95e-02	5.92e-02	0
CodeGPT	MD+L	Language	1.58	1.27e-01	1.67e-01	0
CodeGPT	MD+L	MD	1.95	6.36e-02	9.03e-02	0
CodeGPT	MD+V	MD	1.13	2.71e-01	3.18e-01	0
CodeGPT	MD+V	Visual	2.03	5.46e-02	7.97e-02	0
CodeTransformer	MD+L	MD	0.98	3.38e-01	3.83e-01	0
CodeTransformer	MD+V	MD	1.34	1.94e-01	2.33e-01	0
Seq2Seq	L+V	Language	1.51	1.45e-01	1.82e-01	0
Seq2Seq	MD+L	MD	0.46	6.49e-01	6.49e-01	0
Seq2Seq	MD+V	MD	-0.69	4.97e-01	5.26e-01	0
TF-IDF	MD+V	MD	0.49	6.30e-01	6.42e-01	0
Token Projection	MD+L	Language	1.79	8.59e-02	1.19e-01	0
Token Projection	MD+L	MD	0.97	3.41e-01	3.83e-01	0
Token Projection	MD+V	MD	1.38	1.80e-01	2.20e-01	0
XLNet	MD+L	MD	1.65	1.13e-01	1.52e-01	0
XLNet	MD+V	MD	0.52	6.06e-01	6.29e-01	0

Table 16: Results from paired two-tailed t-tests of joint brain systems with their partial components for each code model. $+t$ reflects $A > B$, whereas $-t$ reflects $A < B$.

G All code properties: Scores and Correlations

We analyzed a series of code properties as part of the static and dynamic code analysis. As several of these properties (*token count*, *node count*, *Halstead difficulty*, *cyclomatic complexity*, and *bytecode ops*) were revealed to be highly correlated in a post-hoc analysis, we report only one measure for this subset in Experiment 1, *token count*, but include all scores here for completeness (Table 17). We also report the correlation matrix between all code properties that led us to select *token count* as the representative property for this subset (Table 18).

Brain Representation Code Properties	MD	Language	Visual	Auditory
Token Count	0.17	0.24	0.09	0.00
Node Count	0.12	0.20	0.03	0.01
Halstead Difficulty	0.11	0.17	0.10	-0.01
Cyclomatic Complexity	0.18	0.24	0.10	0.01
Bytecode Operations	0.15	0.18	0.03	0.02
Runtime Steps	0.33	0.20	0.17	0.12

Table 17: Brain region decoding performance on all *static analysis* and *dynamic analysis* code properties. Scores represent Pearson correlation between predicted and true code properties.

	Datatype	Conditional	Iteration	Tokens	Nodes	Halstead	Cyclomatic	Runtime	Bytecode
Datatype	1.00	0.00	0.00	0.41	0.33	0.39	0.18	0.13	0.28
Conditional	-	1.00	-0.50	0.44	0.46	0.50	0.61	-0.41	0.49
Iteration	-	-	1.00	-0.10	-0.14	-0.35	-0.09	0.90	0.01
Tokens	-	-	-	1.00	0.97	0.89	0.80	0.08	0.95
Nodes	-	-	-	-	1.00	0.86	0.79	0.00	0.96
Halstead	-	-	-	-	-	1.00	0.74	-0.17	0.80
Cyclomatic	-	-	-	-	-	-	1.00	0.02	0.84
Runtime	-	-	-	-	-	-	-	1.00	0.13
Bytecode	-	-	-	-	-	-	-	-	1.00

Table 18: Correlation matrix across all code properties. The *control flow* property was split into two binary properties here, *conditional* and *iteration*. Of relevance is that *token count* presents with $r > 0.8$ for all *static analysis* properties in the set and as such is used as the representative *static analysis* code property in Experiment 1.

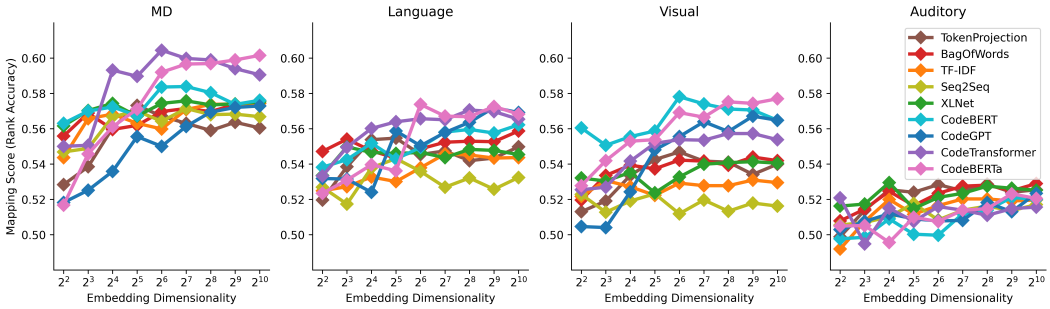

Figure 5: Sensitivity of brain representation mapping to model output dimensions. Each subplot contains the decoding results from a given brain network, and each line reflects a unique code model across a range of controlled embedding dimensions.

H Sensitivity of brain mapping to model output dimensions

Of potential interest to the decoding framework is not just the complexity of code models, but their intrinsic dimensionality as well. In order to investigate to what extent brain model mappings are robust to changes in code model dimensionality, and to assess which model representations are most sensitive to compression and expansion, we rerun our current decoding framework while controlling for embedding size.

For each brain network to code model mapping task, prior to the MVPA analysis, we control for the dimensionality of the code model embedding via projection through a Gaussian random matrix $\mathbb{R}^{d_1 \times d_2}$ drawn from $N(0, 1/d_2)$ where d_1 is the original code model embedding dimensionality and d_2 is the desired dimensionality.

We observe that, on average, models of lower complexity (e.g., *bag-of-words*, *TF-IDF*, *seq2seq*) appear relatively robust to compression, whereas the most complex models (e.g., *CodeTransformer*, *CodeGPT*, *CodeBERTa*) gain ample performance from higher dimensional expression, and suffer considerably when constrained.

These data suggest that higher dimensions of the encoder in complex models encode relevant neural information. Additionally, these effects appear to be most pronounced when decoding from the brain region whose representations yield the strongest mappings, the MD system.

We note here however that we could be observing an interaction effect between model complexity and dimensionality output in these results. Since we cannot fully control for the complexity of these models (by making them all 'equally complex'), this experiment alone cannot drive definitive conclusions.

These results instead constitute a preliminary exploration into the effects of code model dimensionality on brain to model representation mappings, and suggest an avenue for future investigation.

I Robustness of Results to Regression Metric

For the decoding analysis of the continuous valued *dynamic analysis* and *static analysis* properties, it is reasonable to ask why the Pearson correlation metric was chosen as opposed to $RMSE$, as is typically customary for regression tasks. While we present the results using the Pearson correlation metric in the core results for interpretability via the zero-baselne, here we confirm that the use of an $RMSE$ metric leads to the same conclusions. As we see here, the MD, LS, and Visual system decode the *dynamic analysis* property, and MD and LS decode the *static analysis* property with significantly lower $RMSE$ than the null permutation baseline. These results precisely confirm and mirror the patterns observed in Figure 3.

Brain Network	Code Property	RMSE	Null RMSE	Is Significant?
MD	Dynamic Analysis	3.49	4.03 ± 0.08	1
MD	Static Analysis	7.68	8.27 ± 0.15	1
Language	Dynamic Analysis	3.68	3.95 ± 0.08	1
Language	Static Analysis	7.34	8.00 ± 0.15	1
Visual	Dynamic Analysis	3.79	4.05 ± 0.08	1
Visual	Static Analysis	7.99	8.28 ± 0.16	0
Auditory	Dynamic Analysis	3.79	3.98 ± 0.07	0
Auditory	Static Analysis	8.04	8.10 ± 0.15	0

Table 19: $RMSE$ between observed and predicted continuous-valued code properties from the MVPA regression task in Experiment 1, confirming the pattern of results observed in 5.1