Preliminary Pruning on Language Models for Code

Alex Gu, Bharat Runwal, Diganta Misra, Ria Sonecha, Saaketh Vedantam

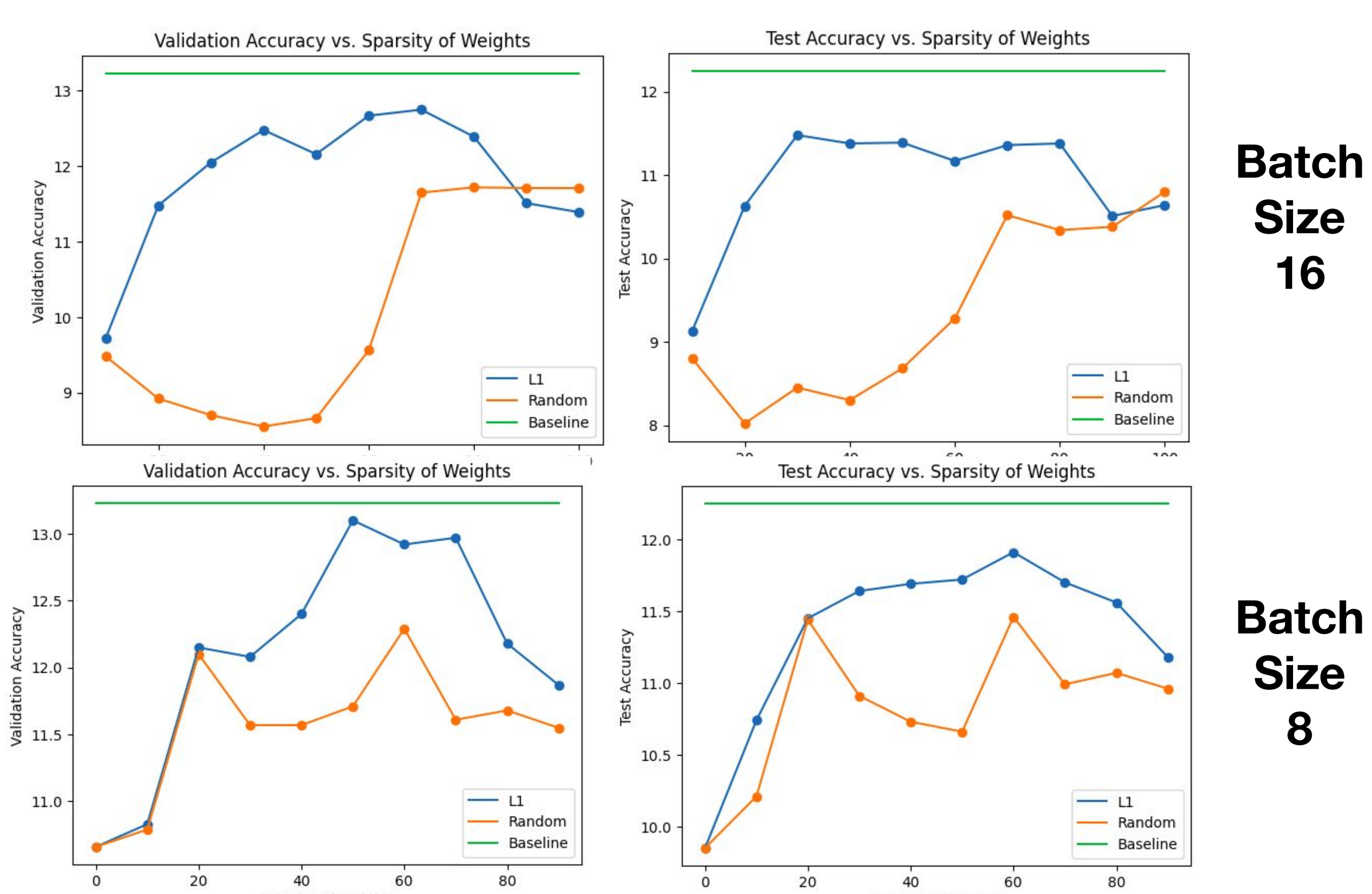




Can pruning be used to understand how language models for text differ from language models for code?

Setting: Ruby code summarization with CodeBERT measured with BLEU score

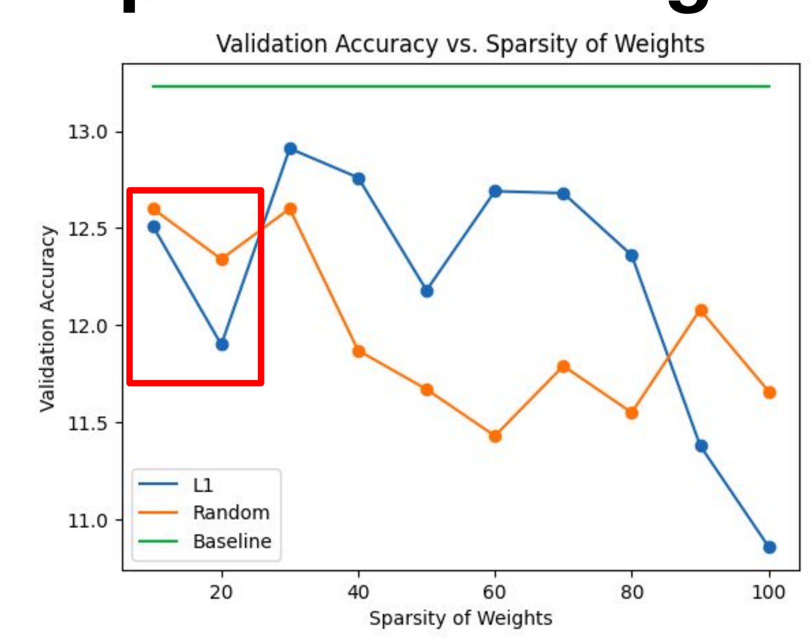
Failed Attempt I: Rewinding to CodeBERT?



Lesson 1: batch size matters a lot
Lesson 2: need to rewind to finetuned CodeBERT?

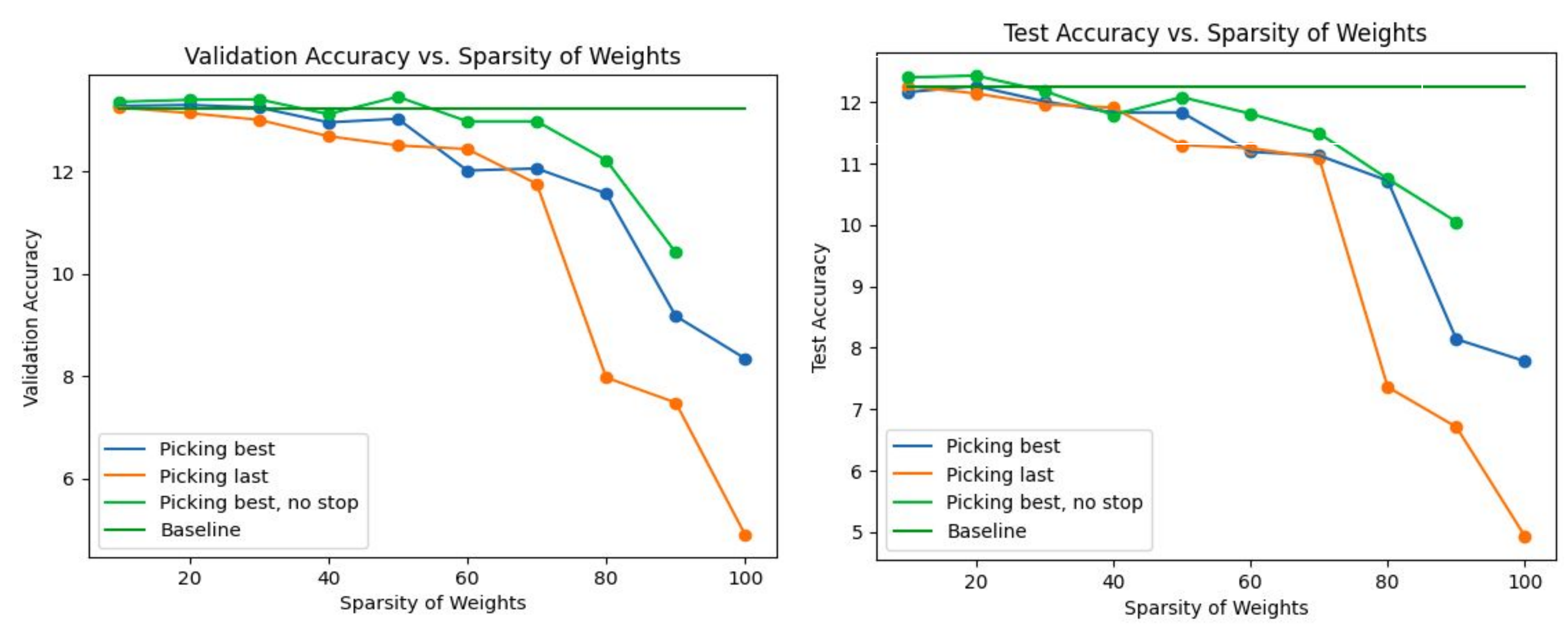
Failed Attempt II: Overfitting?

Random actually performs better?



Hypothesis: are we overfitting every time we finetune?

Final Result

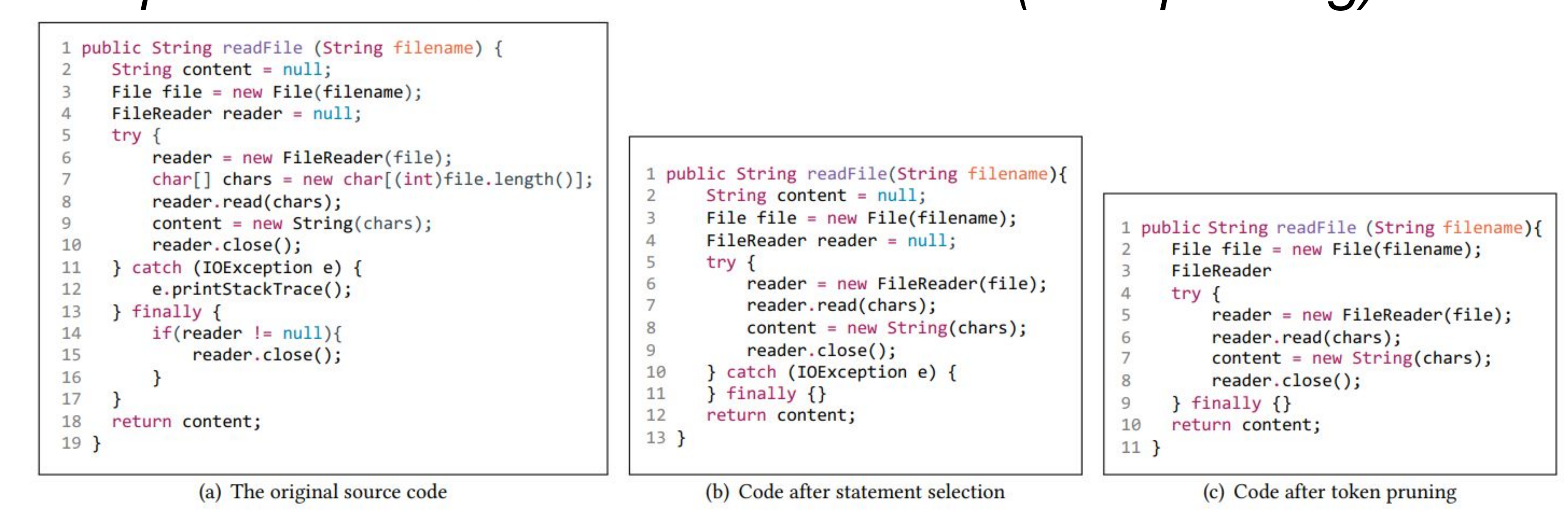


Exciting Directions for Exploration!

Can we look holistically at other code tasks/metrics to understand what is lost when we prune? (CodeT5 on APPS)

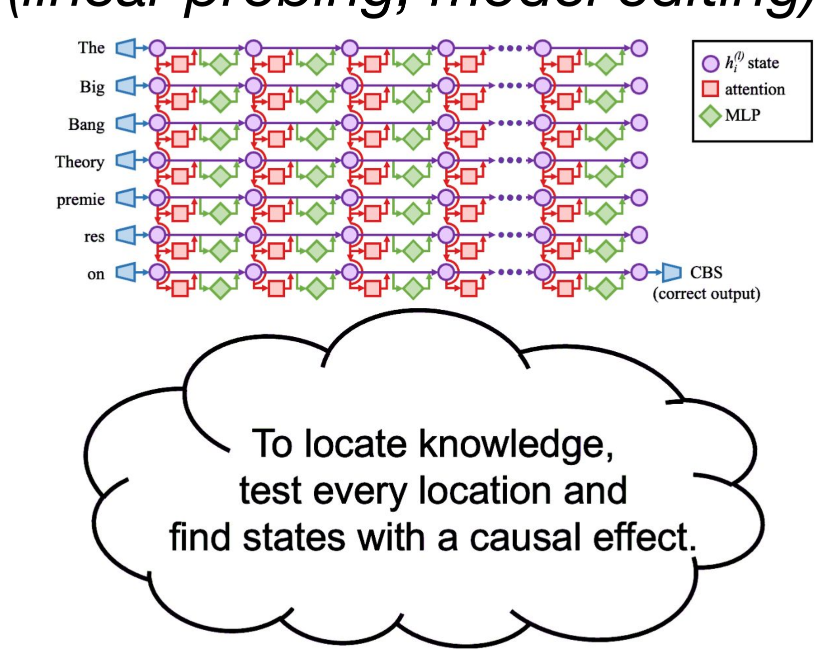
pass@k metric	Dense (%)	OMP(%)	IMP(%)
pass@1	**74.0**	59.0	44.66
pass@2	**93.0**	82.56	69.35
pass@10	**99.99**	99.93	99.71

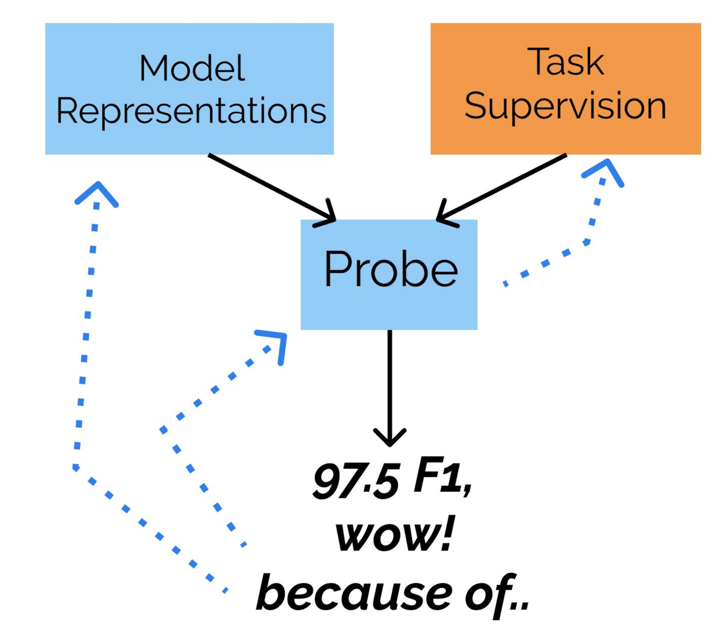
How much information do code models really need to produce and/or summarize code (data pruning)?



Diet Code Is Healthy: Simplifying Programs for Pre-trained Models of Code [Zhang, et al. 2022]

Can we combine pruning and interpretability techniques (linear probing, model editing) to understand what's learned?

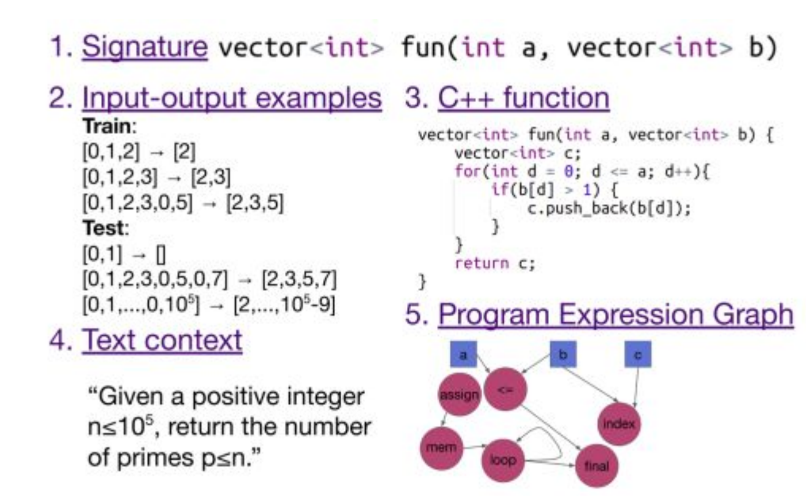




Locating and Editing Factual Associations in GPT [Meng, et al.] *John Hewitt's blog post on Designing and Interpreting Probes*

How can we leverage the structured and precise nature of code to design more efficient language models on them?

- Abstractions
- Precise semantics
- Access to execution information
- Test cases (correctness)
- Compositionality



A large-scale benchmark for few-shot program induction and synthesis [Alet, et al.]