Building Domain-Specific Sub-Models from Large Language Models using Pruning

2024-06-14

Laura Puccioni

LARGE LANGUAGE MODELS

LARGE LANGUAGE MODELS

Gemini

ChatGPT

LLaMA by Meta

MISTRAL AI_

Bard

GitHub Copilot

MAIN CHALLENGES

LARGE LANGUAGE MODELS

Gemini

ChatGPT

LLaMA by Meta

MISTRAL AI_

Bard

GitHub Copilot

MAIN CHALLENGES

- Computational cost
- High memory demand

LARGE LANGUAGE MODELS

Gemini

ChatGPT

LLaMA by Meta

MISTRAL AI_

Bard

GitHub Copilot

MAIN CHALLENGES

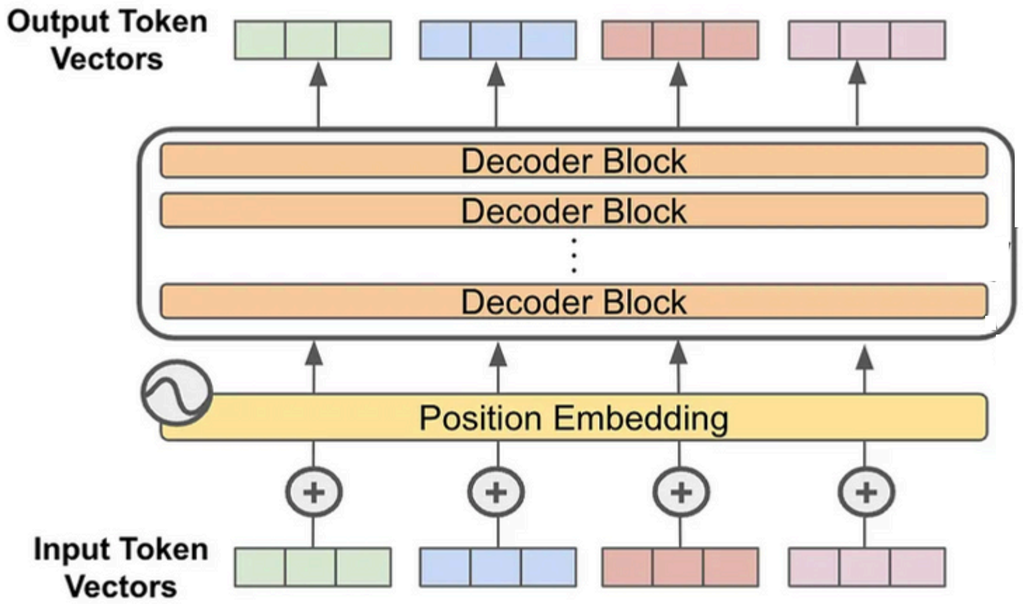
- Computational cost
- High memory demand

A POSSIBLE SOLUTION

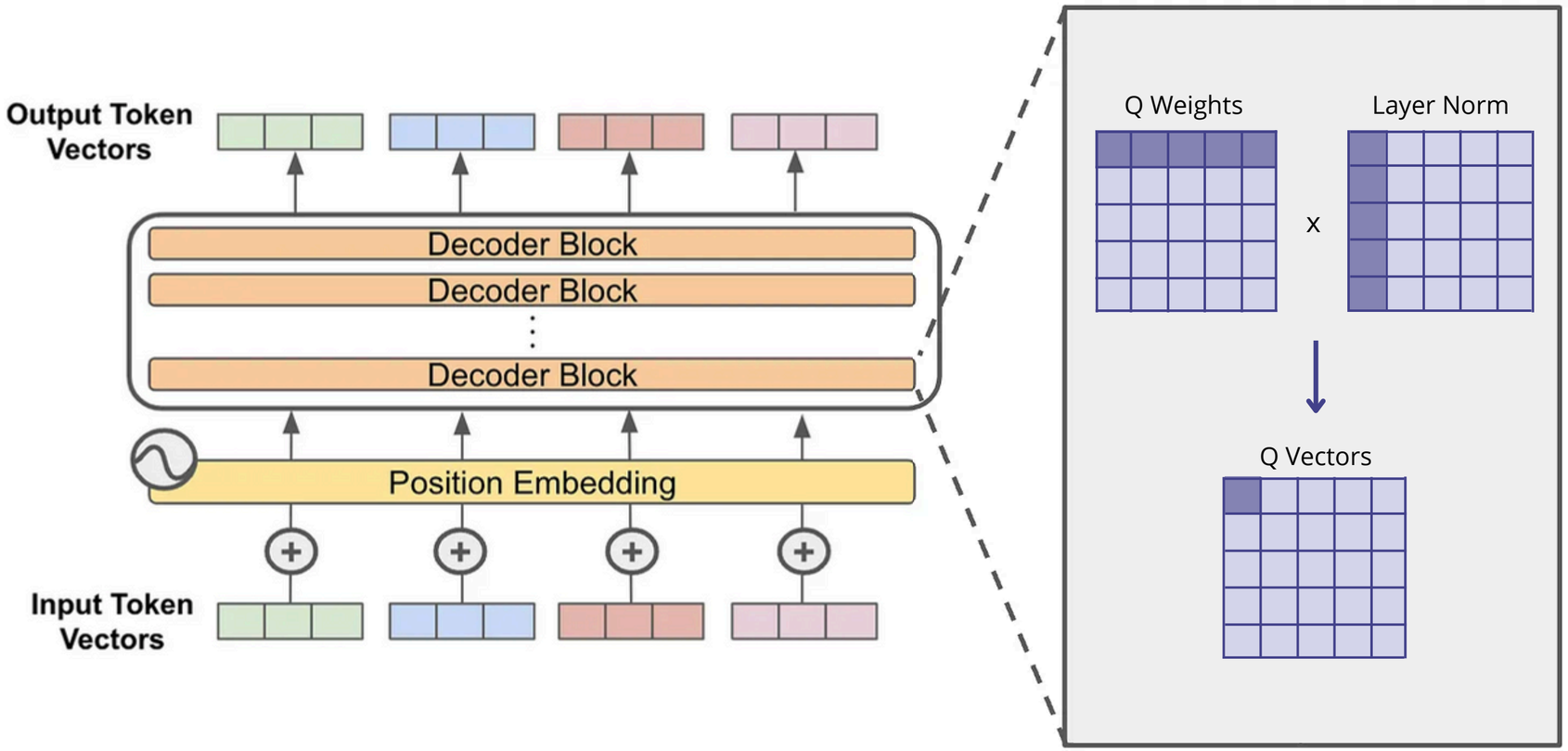
- Model compression

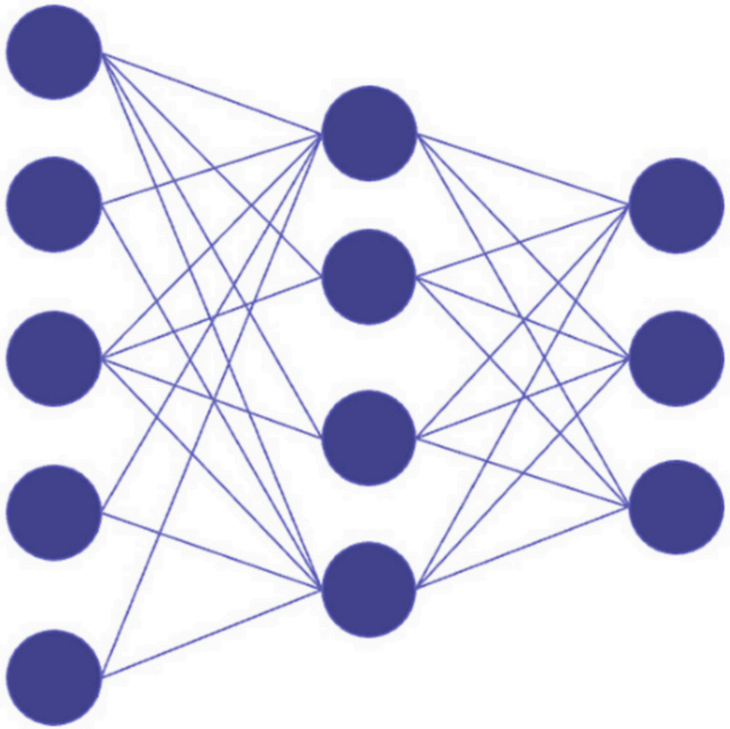
BACKGROUND

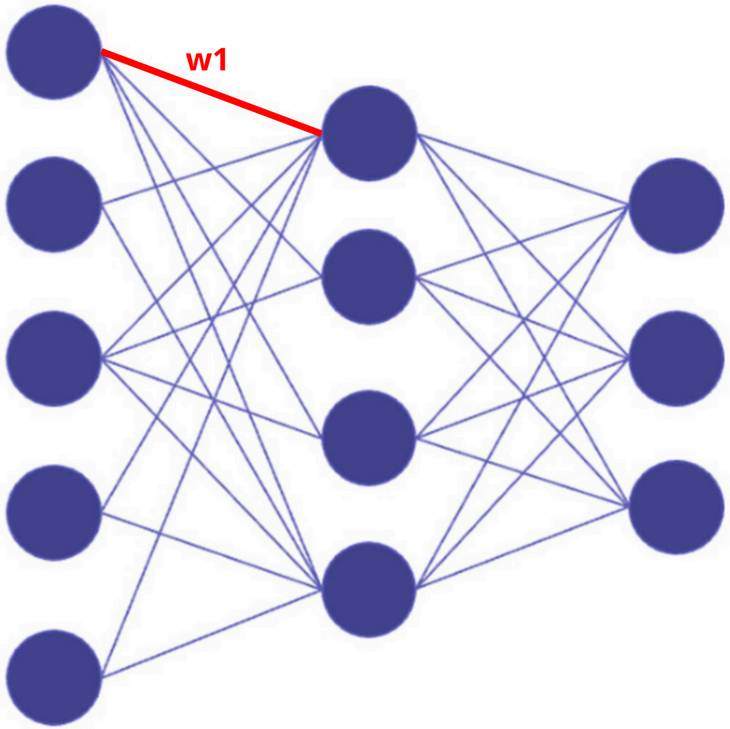
RECENT LLMs ARE DECODER-ONLY MODELS

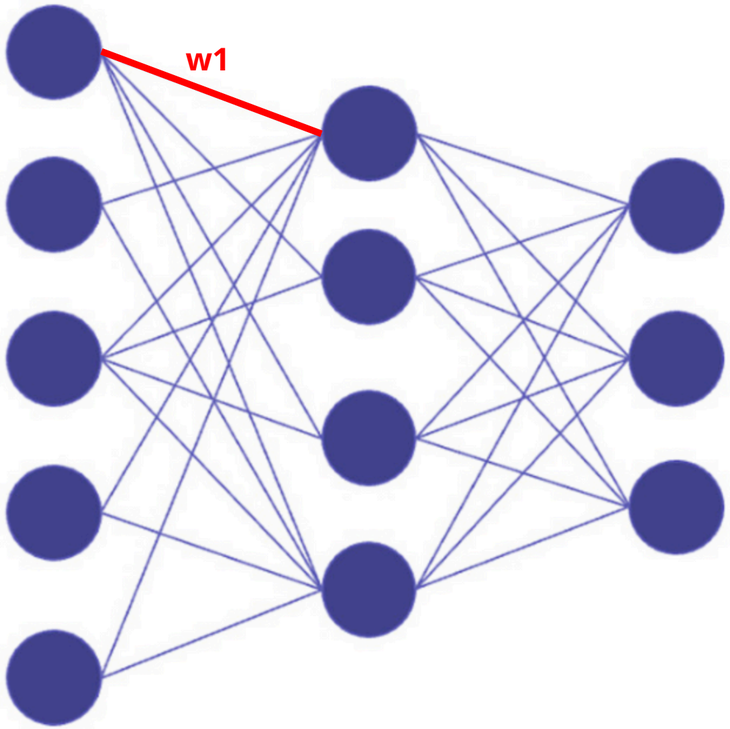


RECENT LLMs ARE DECODER-ONLY MODELS

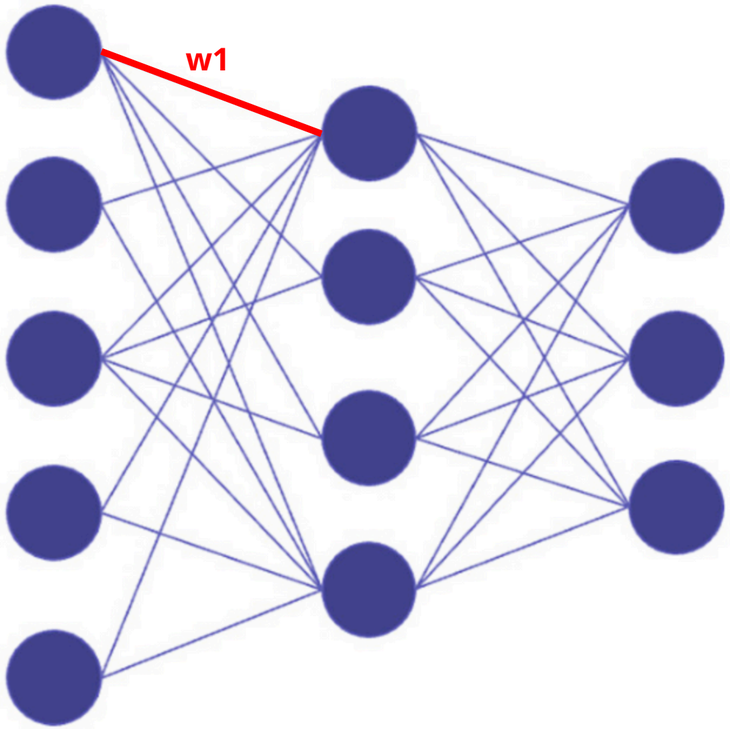





w1



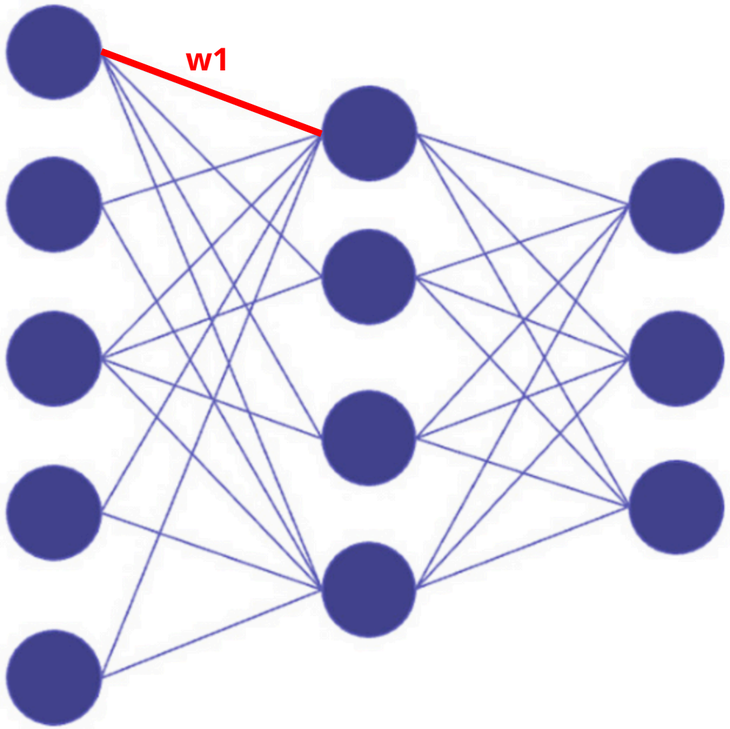
Are all of these weights essential?



Are all of these weights essential?

MODEL COMPRESSION:

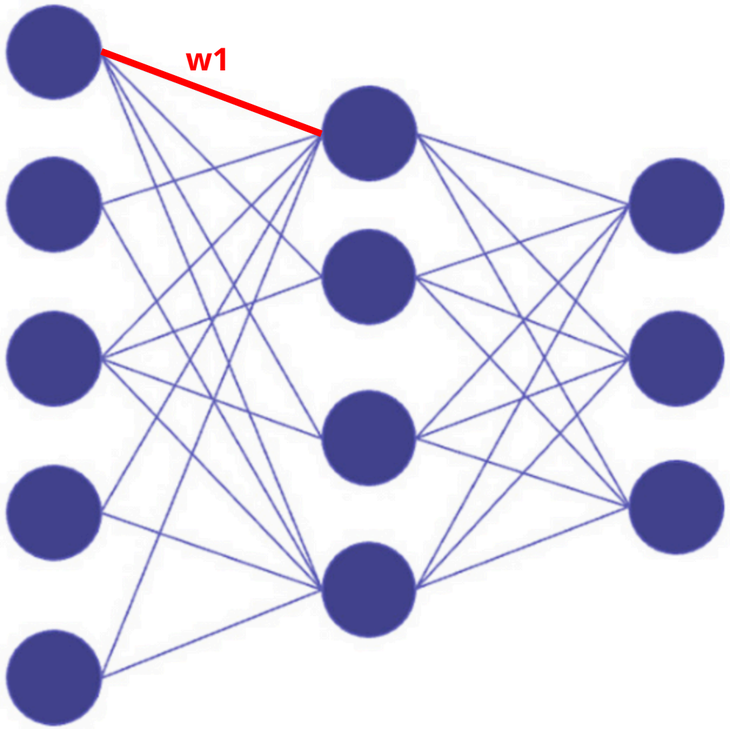
Transform large, resource-intensive models into more compact versions.



Are all of these weights essential?

MODEL COMPRESSION:
Transform large, resource-intensive models into more compact versions.

- **PRUNING**
- **QUANTIZATION**
- **KNOWLEDGE DISTILLATION**
- **LOW-RANK FACTORIZATION**



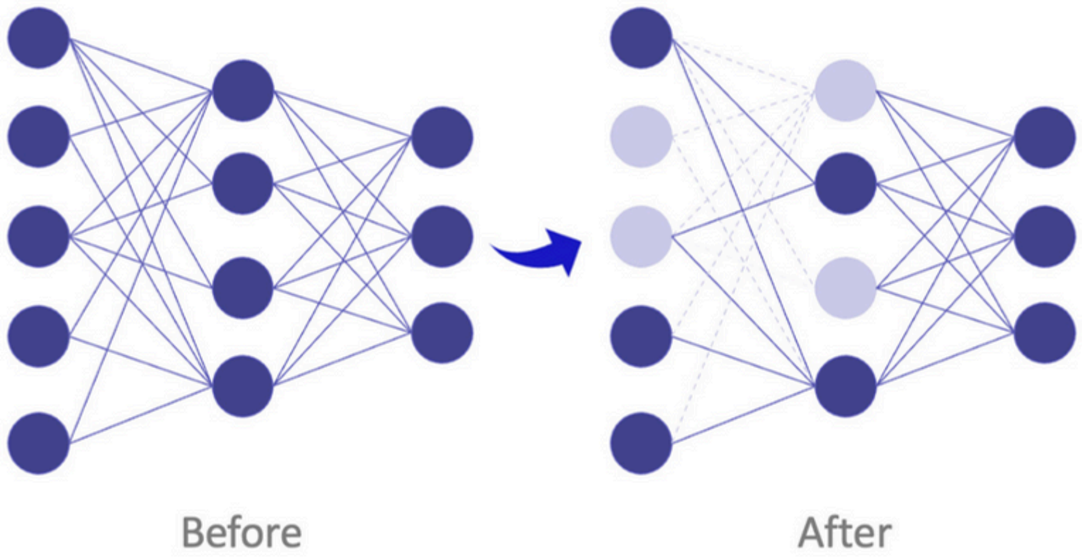
Are all of these weights essential?

MODEL COMPRESSION:
Transform large, resource-intensive models into more compact versions.

- **PRUNING**
- **QUANTIZATION**
- **KNOWLEDGE DISTILLATION**
- **LOW-RANK FACTORIZATION**

PRUNING

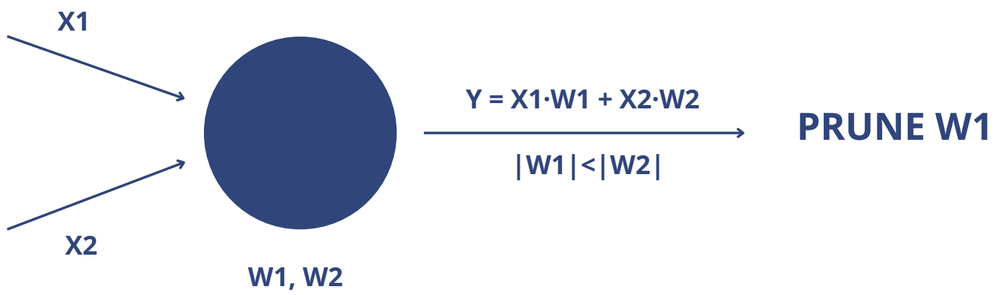
Removing neurons and connections (weights) to shrink the model.



- Efficient **parameter reduction**
- Improves **computational efficiency**
- Potentially reduces **storage requirements**
- Preserves **high accuracy**

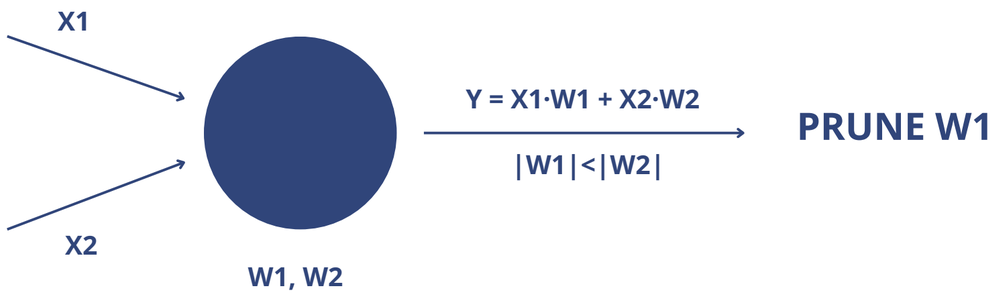
MAGNITUDE PRUNING

Zero out all the weights with **low magnitude.**



MAGNITUDE PRUNING

Zero out all the weights with **low magnitude.**

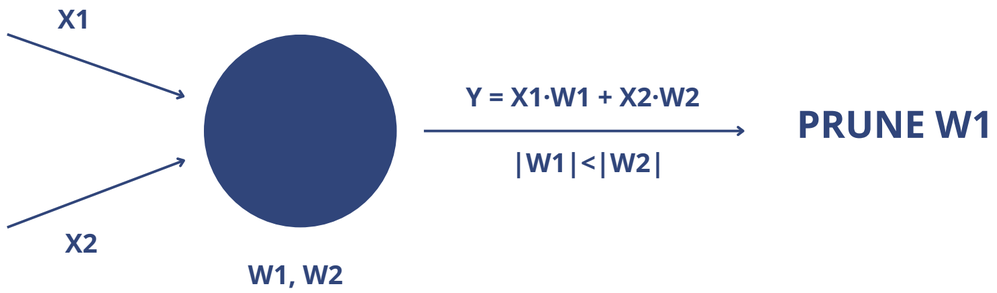


LIMITATIONS:

- Does not consider input activations.
- Dramatically fails with LLMs.

MAGNITUDE PRUNING

Zero out all the weights with **low magnitude.**



LIMITATIONS:

- Does not consider input activations.
- Dramatically fails with LLMs.

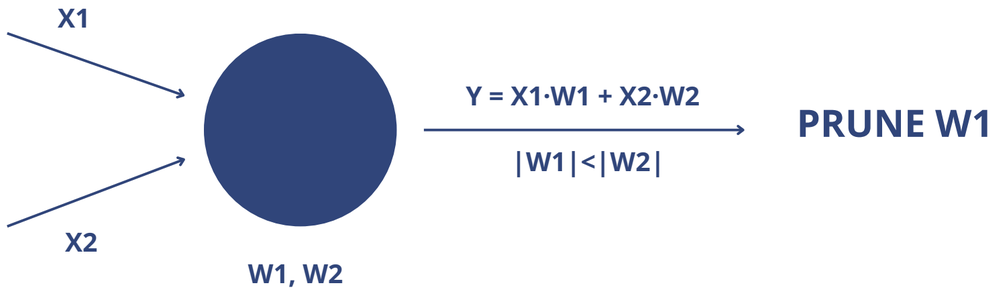
SPARSEGPT

Formalizes the problem of pruning LLMs by solving a **local layer-wise reconstruction problem.**

Minimize $\|W_i \cdot X_i - \hat{W}_i \cdot X_i\|^2$

MAGNITUDE PRUNING

Zero out all the weights with **low magnitude.**



LIMITATIONS:

- Does not consider input activations.
- Dramatically fails with LLMs.

SPARSEGPT

Formalizes the problem of pruning LLMs by solving a **local layer-wise reconstruction problem.**

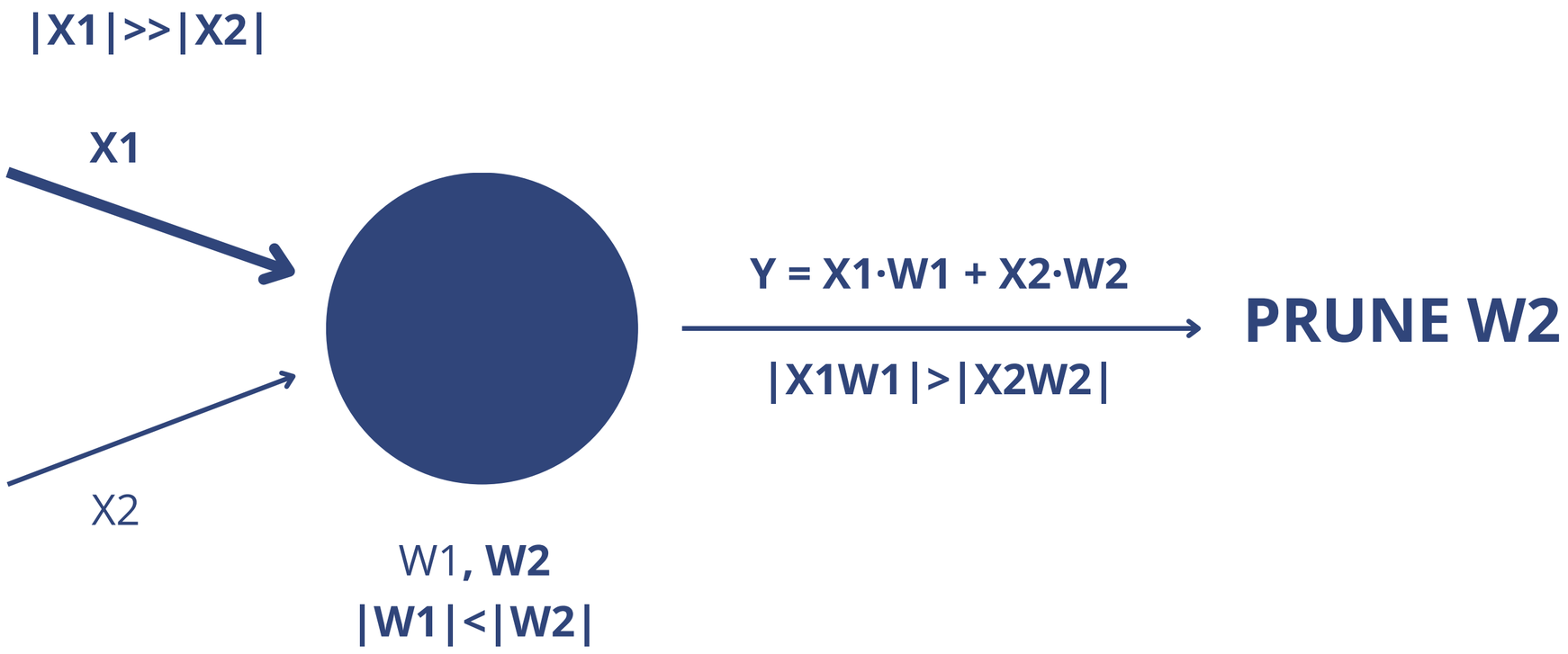
Minimize $\|W_i \cdot X_i - \hat{W}_i \cdot X_i\|^2$

MAIN LIMITATION:

- Compute intensive.

WANDA: Weight and Activations

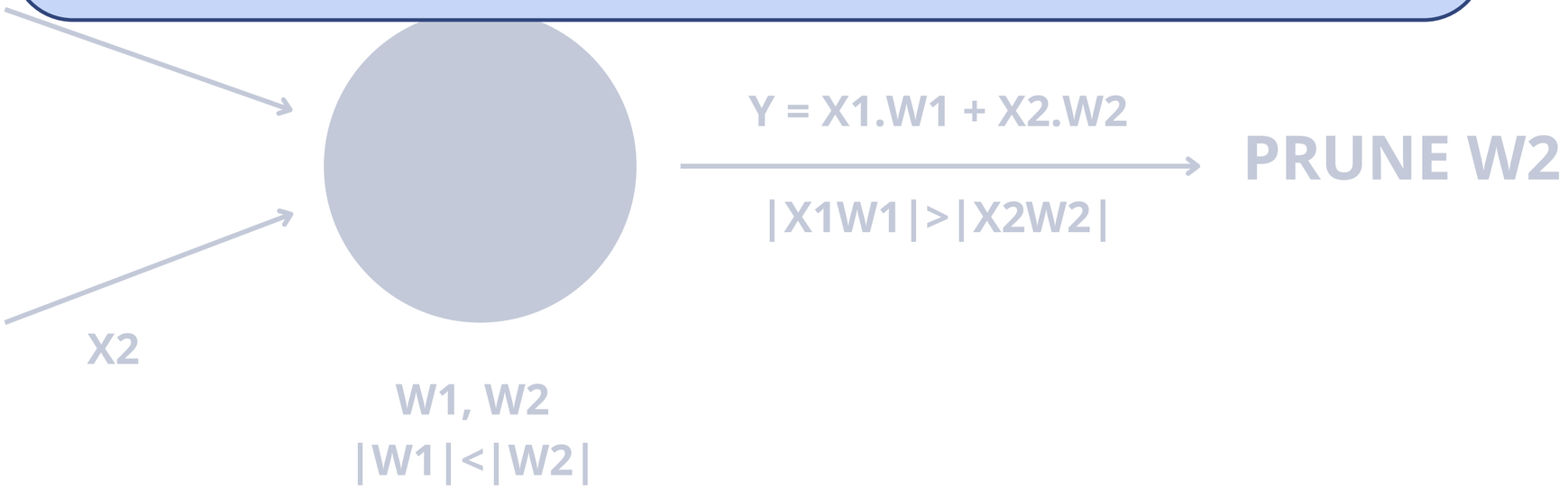
Prune weights based on **magnitude** and **input activations**



WANDA: Weight and Activations

Prune weights based on **magnitude** and **input activations**

Wanda **outperforms** other pruning methods.



WANDA: Weight and Activations

Prune weights based on **magnitude** and **input activations**

Wanda **outperforms** other pruning methods.

Wanda estimates the feature norm statistics using **one single set of calibration samples**: C4.

W2

W1, W2

|W1|<|W2|

WANDA: Weight and Activations

Prune weights based on **magnitude** and **input activations**

Wanda **outperforms** other pruning methods.

Wanda estimates the feature norm statistics using **one single set of calibration samples**: C4.[1]

W2

W1, W2

|W1|<|W2|

[1] C4 dataset is a collection of about 750GB of English-language text sourced from the public Common Crawl web scrape.

Is it possible to build domain-specific sub-models by pruning a Large Language model with task-specific calibration datasets?

HYPOTHESES

01 It is possible to build **task-specific models** by pruning foundational LLMs using the **appropriate dataset.**

02 **Different tasks** activate **different parts** of LLMs, resulting in distinct parameter masks.

METHODOLOGY

DATA COLLECTION

DATA COLLECTION

MODELS

CodeLlama-7B-Instruct

Mistral-7B-Instruct

Gemma-7B-Instruct

CodeLlama-13B-Instruct

DATA COLLECTION

MODELS

CodeLlama-7B-Instruct

Mistral-7B-Instruct

Gemma-7B-Instruct

CodeLlama-13B-Instruct

TASKS

Code Generation

Commonsense Reasoning

Translation

Math

DATA COLLECTION

MODELS

CodeLlama-7B-Instruct

Mistral-7B-Instruct

Gemma-7B-Instruct

CodeLlama-13B-Instruct

TASKS

Code Generation

Commonsense Reasoning

Translation

Math

→

3 DATASETS PER TASK

PRUNING PROCESS

Technique: **WANDA**

PRUNING PROCESS

Technique: **WANDA**

Sparsity: **50% unstructured**

PRUNING PROCESS

Technique: **WANDA**

Sparsity: **50% unstructured**

calibration samples: **128**

PRUNING PROCESS

Technique: **WANDA**

Sparsity: **50% unstructured**

calibration samples: **128**

Outcome: **52 sub-models**

EVALUATION

EVALUATION

Code

METRIC
pass@k

BENCHMARKS
HumanEval
MBPP

EVALUATION

Code

METRIC
pass@k

BENCHMARKS
HumanEval
MBPP

CSR

METRIC
0-shot accuracy

BENCHMARKS
6 CSR datasets

EVALUATION

Code

METRIC
pass@k

BENCHMARKS
HumanEval
MBPP

CSR

METRIC
0-shot accuracy

BENCHMARKS
6 CSR datasets

Translation

METRIC
METEOR

BENCHMARKS
wmt14
opus_books

EVALUATION

Code

METRIC
pass@k

BENCHMARKS
HumanEval
MBPP

CSR

METRIC
0-shot accuracy

BENCHMARKS
6 CSR datasets

Translation

METRIC
METEOR

BENCHMARKS
wmt14
opus_books

Math

METRIC
8-shot accuracy

BENCHMARKS
gsm8k

RESULTS AND ANALYSIS

RESULTS

Domain	Calibration Set	Evaluation Metrics			
		Code (pass@10)	CSR (0-shot)	Translation (meteor)	MATH (8-shot)
None	None	~52.9%	~40.57%	~50.41%	~46.4%
Code	py_alpaca	~46.99%	~38.23%	~45.75%	~32.44%
	apps	**~47.07%**	~36.47%	~45.84%	~32.2%
	github_py	~46.47%	~37.34%	~45.23%	~33.6%
CSR	OB_qa	~43.23%	**~38.9%**	~45.63%	~26.9%
	Hellaswag	~40.82%	~38.4%	~44.28%	~26.99%
	Csense_qa	~41.61%	~38.45%	~45.21%	~27.29%
Transl	wmt14	~38.58%	~38.34%	~48.55%	~27.14%
	song_transl	~40.91%	~37.58%	~48.29%	~29.7%
	opus_books	~41.01%	~39.03%	**~48.6%**	~26.8%
Math	math_qa	~44.44%	~37.65%	~45.38%	~35.25%
	math_inst	~45.26%	~37.95%	~45.15%	~35.10%
	math_orca	~45.05%	~37.39%	~45.6%	**~36.16%**
Eng Text	c4	~42.2%	~38.1%	~46.74%	~29.00%

RESULTS

As expected, pruning **reduces accuracy.**

Domain	Calibration Set	Evaluation Metrics			
		Code (pass@10)	CSR (0-shot)	Translation (meteor)	MATH (8-shot)
None	None	~52.9%	~40.57%	~50.41%	~46.4%
Code	py_alpaca	~46.99%	~38.23%	~45.75%	~32.44%
	apps	**~47.07%**	~36.47%	~45.84%	~32.2%
	github_py	~46.47%	~37.34%	~45.23%	~33.6%
CSR	OB_qa	~43.23%	**~38.9%**	~45.63%	~26.9%
	Hellaswag	~40.82%	~38.4%	~44.28%	~26.99%
	Csense_qa	~41.61%	~38.45%	~45.21%	~27.29%
Transl	wmt14	~38.58%	~38.34%	~48.55%	~27.14%
	song_transl	~40.91%	~37.58%	~48.29%	~29.7%
	opus_books	~41.01%	~39.03%	**~48.6%**	~26.8%
Math	math_qa	~44.44%	~37.65%	~45.38%	~35.25%
	math_inst	~45.26%	~37.95%	~45.15%	~35.10%
	math_orca	~45.05%	~37.39%	~45.6%	**~36.16%**
Eng Text	c4	~42.2%	~38.1%	~46.74%	~29.00%

RESULTS

As expected, pruning **reduces accuracy.**

c4 dataset **never** yields the best results.

Domain	Calibration Set	Evaluation Metrics			
		Code (pass@10)	CSR (0-shot)	Translation (meteor)	MATH (8-shot)
None	None	~52.9%	~40.57%	~50.41%	~46.4%
Code	py_alpaca	~46.99%	~38.23%	~45.75%	~32.44%
	apps	**~47.07%**	~36.47%	~45.84%	~32.2%
	github_py	~46.47%	~37.34%	~45.23%	~33.6%
CSR	OB_qa	~43.23%	**~38.9%**	~45.63%	~26.9%
	Hellaswag	~40.82%	~38.4%	~44.28%	~26.99%
	Csense_qa	~41.61%	~38.45%	~45.21%	~27.29%
Transl	wmt14	~38.58%	~38.34%	~48.55%	~27.14%
	song_transl	~40.91%	~37.58%	~48.29%	~29.7%
	opus_books	~41.01%	~39.03%	**~48.6%**	~26.8%
Math	math_qa	~44.44%	~37.65%	~45.38%	~35.25%
	math_inst	~45.26%	~37.95%	~45.15%	~35.10%
	math_orca	~45.05%	~37.39%	~45.6%	**~36.16%**
Eng Text	c4	~42.2%	~38.1%	~46.74%	~29.00%

RESULTS

Domain	Calibration Set	Evaluation Metrics			
		Code (pass@10)	CSR (0-shot)	Translation (meteor)	MATH (8-shot)
None	None	~52.9%	~40.57%	~50.41%	~46.4%
Code	py_alpaca	~46.99%	~38.23%	~45.75%	~32.44%
	apps	**~47.07%**	~36.47%	~45.84%	~32.2%
	github_py	~46.47%	~37.34%	~45.23%	~33.6%
CSR	OB_qa	~43.23%	**~38.9%**	~45.63%	~26.9%
	Hellaswag	~40.82%	~38.4%	~44.28%	~26.99%
	Csense_qa	~41.61%	~38.45%	~45.21%	~27.29%
Transl	wmt14	~38.58%	~38.34%	~48.55%	~27.14%
	song_transl	~40.91%	~37.58%	~48.29%	~29.7%
	opus_books	~41.01%	~39.03%	**~48.6%**	~26.8%
Math	math_qa	~44.44%	~37.65%	~45.38%	~35.25%
	math_inst	~45.26%	~37.95%	~45.15%	~35.10%
	math_orca	~45.05%	~37.39%	~45.6%	**~36.16%**
Eng Text	c4	~42.2%	~38.1%	~46.74%	~29.00%

As expected, pruning **reduces accuracy.**

c4 dataset **never** yields the best results.

Using **task-specific** calibration samples **enhances the performance** of the pruned models on those specific tasks compared to general calibration sets.

RESULTS

Domain	Calibration Set	Evaluation Metrics			
		Code (pass@10)	CSR (0-shot)	Translation (meteor)	MATH (8-shot)
	opus_books	~41.01%	~39.03%	**~48.6%**	~26.8%
Math	math_qa	~44.44%	~37.65%	~45.38%	~35.25%
	math_inst	~45.26%	~37.95%	~45.15%	~35.10%
	math_orca	~45.05%	~37.39%	~45.6%	**~36.16%**
Eng Text	c4	~42.2%	~38.1%	~46.74%	~29.00%

As exp… prunin… in a **lo… accur…**

c4 datas… yields the best results

…k-specific …n samples **…s the …ance** of the …odels on …cific tasks …d to general …n sets.

HYPOTHESIS 1

It is possible to build task-specific sub-models by pruning foundational LLMs using the appropriate dataset.

OBSERVATION: CODE GENERATION

- The **highest accuracy** for code generation tasks is consistently achieved when pruning with a **coding dataset**.

OBSERVATION: CODE GENERATION

- The **highest accuracy** for code generation tasks is consistently achieved when pruning with a **coding dataset**.
- One would expect that CodeLlama would retain the highest accuracy compared to the original model.

OBSERVATION: CODE GENERATION

- The **highest accuracy** for code generation tasks is consistently achieved when pruning with a **coding dataset**.
- One would expect that CodeLlama would retain the highest accuracy compared to the original model.

BUT

CodeLlama

Domain	Calibration Set	Code (pass@10)
None	None	~71.00%
Code	py_alpaca	**~61.00%**
	apps	~60.7%
	github_py	~60.5%

Gemma

Domain	Calibration Set	Code (pass@10)
None	None	~52.9%
Code	py_alpaca	~46.99%
	apps	**~47.07%**
	github_py	~46.47%

OBSERVATION: CODE GENERATION

- The **highest accuracy** for code generation tasks is consistently achieved when pruning with a **coding dataset**.
- One would expect that CodeLlama would retain the highest accuracy compared to the original model.

BUT

CodeLlama

Domain	Calibration Set	Code (pass@10)
None	None	~71.00%
Code	py_alpaca	**~61.00%**
	apps	~60.7%
	github_py	~60.5%

↕ - 10

Gemma

Domain	Calibration Set	Code (pass@10)
None	None	~52.9%
Code	py_alpaca	~46.99%
	apps	**~47.07%**
	github_py	~46.47%

↕ - 6

Gemma is the model with the lowest accuracy loss.

OBSERVATION: CODE GENERATION

- The **highest accuracy** for code generation tasks is consistently achieved when pruning with a **coding dataset**.

Gemma has a higher robustness to pruning due to its broader training data and more generalized learning.

	py_alpaca	**~61%**		py_alpaca	~46.99%
Code	apps	~60.7%	Code	apps	**~47.07%**
	github_py	~60.5%		github_py	~46.47%

Gemma is the model with the lowest accuracy loss.

OBSERVATION: COMMONSENSE REASONING

Domain	CSR (0-shot)	CSR (0-shot)	CSR (0-shot)	CSR (0-shot)
None	~54.7%	~52.3%	~61.4%	~40.57%
Code	~51.3%	~48.41%	~58.25%	~38.23%
	~51.1%	~47.06%	~57.11%	~36.47%
	~51.6%	~48.55%	~57.99%	~37.34%
CSR	~52.3%	~48.4%	~58.2%	**~38.9%**
	~52.5%	**~49.3%**	~57.6%	~38.4%
	~52.17%	~48.2%	~58.2%	~38.45%
Transl	~52.2%	~49.1%	~58.08%	~38.34%
	~51.9%	~48.28%	~57.62%	~37.58%
	~52.05%	~48.06%	~57.77%	~39.03%
Math	~52.05%	~48.32%	~57.68%	~37.65%
	~51.8%	~48.63%	**~58.26%**	~37.95%
	~52.08%	~48.67%	~57.95%	~37.39%
Eng Text	~51.5%	~49.02%	~57.8%	~38.1%

- Commonsense reasoning tasks demonstrate accuracies that are relatively **consistent across different-domain sub-models**.

OBSERVATION: COMMONSENSE REASONING

Domain	CSR (0-shot)	CSR (0-shot)	CSR (0-shot)	CSR (0-shot)
None	~54.7%	~52.3%	~61.4%	~40.57%
Code	~51.3%	~48.41%	~58.25%	~38.23%
	~51.1%	~47.06%	~57.11%	~36.47%
	~51.6%	~48.55%	~57.99%	~37.34%
CSR	~52.3%	~48.4%	~58.2%	**~38.9%**
	~52.5%	**~49.3%**	~57.6%	~38.4%
	~52.17%	~48.2%	~58.2%	~38.45%
Transl	~52.2%	~49.1%	~58.08%	~38.34%
	~51.9%	~48.28%	~57.62%	~37.58%
	~52.05%	~48.06%	~57.77%	~39.03%
Math	~52.05%	~48.32%	~57.68%	~37.65%
	~51.8%	~48.63%	**~58.26%**	~37.95%
	~52.08%	~48.67%	~57.95%	~37.39%
Eng Text	~51.5%	~49.02%	~57.8%	~38.1%

- Commonsense reasoning tasks demonstrate accuracies that are relatively **consistent across different-domain sub-models**.

↓

Sub-models do not derive significant benefits from the usage of specifically commonsense reasoning calibration samples.

OBSERVATION: COMMONSENSE REASONING

Domain	CSR (0-shot)	CSR (0-shot)	CSR (0-shot)	CSR (0-shot)
None	~54.7%	~52.3%	~61.4%	~40.57%
Cod				
CSR				
Transl	~51.9%	~48.28%	~57.62%	~37.58%
	~52.05%	~48.06%	~57.77%	~39.03%
Math	~52.05%	~48.32%	~57.68%	~37.65%
	~51.8%	~48.63%	**~58.26%**	~37.95%
	~52.08%	~48.67%	~57.95%	~37.39%
Eng Text	~51.5%	~49.02%	~57.8%	~38.1%

Natural language is already rich in commonsense reasoning knowledge, so using a specific calibration sample does not have any impact.

- Commonsense reasoning tasks demonstrate **tent across**

Sub-models do not derive significant benefits from the usage of specifically commonsense reasoning calibration samples.

OBSERVATION: TRANSLATION

- **Translation-specific** sub-models have the **highest accuracy** on translation.

OBSERVATION: TRANSLATION

- **Translation-specific** sub-models have the **highest accuracy** on translation.
- With translation there is a significantly **smaller loss in accuracy** compared to code generation sub-models.

Domain	Calibration Set	Code (pass@10)
None	None	~60.5%
Code	py_alpaca	**~47.28%**
	apps	~45.86%
	github_py	~46.87%

↕ - 13.22

Domain	Calibr Set	Translation (meteor)
None	None	~52.95%
Transl	wmt14	**~51.31%**
	song_transl	~50.32%
	opus_books	~50.41%

↕ - 1.64

OBSERVATION: TRANSLATION

- **Translation-specific** sub-models have the **highest accuracy** on translation.
- With translation there is a significantly **smaller loss in accuracy** compared to code generation sub-models.

Domain	Calibration Set	Code (pass@10)
None	None	~60.5%
Code	py_alpaca	**~47.28%**
	apps	~45.86%
	github_py	~46.87%

↕ - 13.22

Domain	Calibr Set	Translation (meteor)
None	None	~52.95%
Transl	wmt14	**~51.31%**
	song_transl	~50.32%
	opus_books	~50.41%

↕ - 1.64

  - Translation tasks involve more **direct mapping** between input and output sequences, while coding tasks are **more complex.**

OBSERVATION: TRANSLATION

- **Translation-specific** sub-models have the **highest accuracy** on translation.
- With translation there is a significantly **smaller loss in accuracy** compared to code generation sub-models.

Unlike translation-specific sub-models, code generation sub-models lose critical information or structure after pruning.

Code	apps	~45.86%	Transl	sum_transl	~50.32%
	github_py	~46.87%		opus_books	~50.41%

- Translation tasks involve more **direct mapping** between input and output sequences, while coding tasks are **more complex.**

OBSERVATION: CODE/MATH SUB-MODELS

- Code and math sub-models experience a relatively modest drop in accuracy when evaluated on their counterpart domains (i.e., math models evaluated on code tasks and vice versa).

OBSERVATION: CODE/MATH SUB-MODELS

- Code and math sub-models experience a relatively modest drop in accuracy when evaluated on their counterpart domains (i.e., math models evaluated on code tasks and vice versa).

Domain	Calibration Set	Code (pass@10)
None	None	~52.9%
Code	py_alpaca	~46.99%
	apps	**~47.07%**
	github_py	~46.47%
Math	math_qa	~44.44%
	math_inst	~45.26%
	math_orca	~45.05%

-5.83

- 7.64

Domain	Calibration Set	MATH (8-shot)
None	None	~46.4%
Code	py_alpaca	~32.44%
	apps	~32.2%
	github_py	~33.6%
Math	math_qa	~35.25%
	math_inst	~35.10%
	math_orca	**~36.16%**

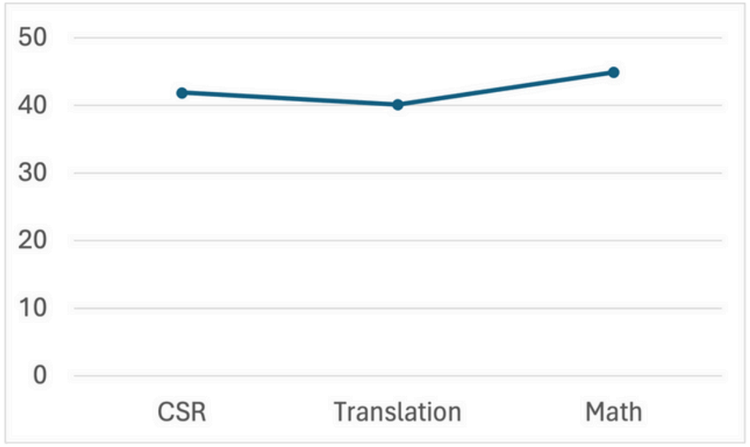
-13.4

-10.24

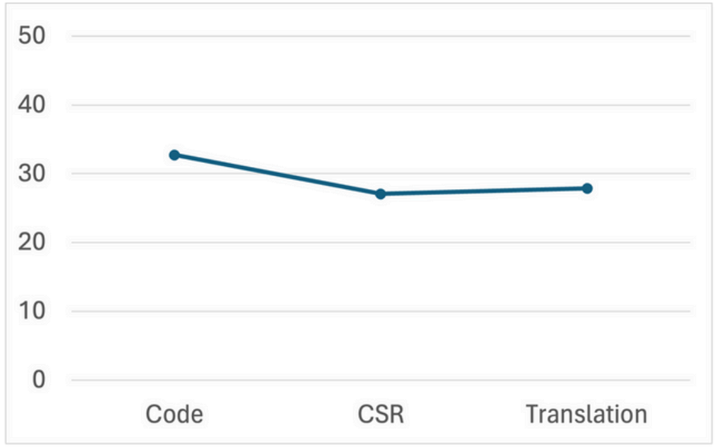
Evaluations made on Gemma sub-models

OBSERVATION: CODE/MATH SUB-MODELS

- Code and math sub-models experience a relatively modest drop in accuracy when evaluated on their counterpart domains (i.e., math models evaluated on code tasks and vice versa).



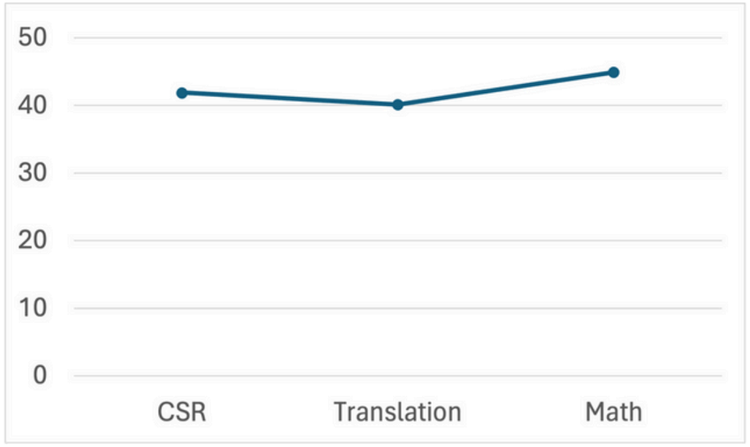
Performance of Gemma sub-models on Code



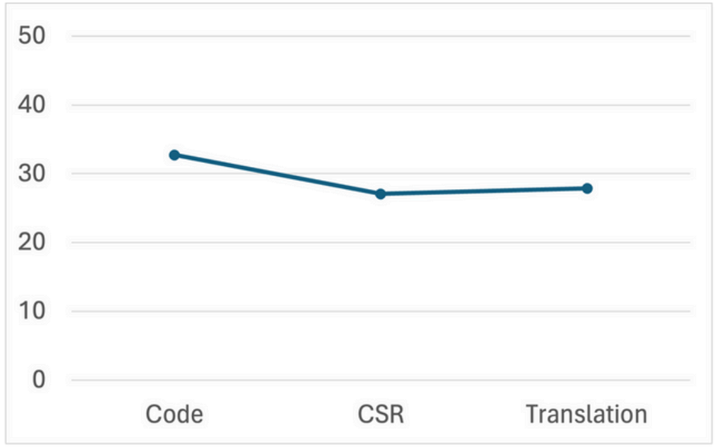
Performance of Gemma sub-models on Math

OBSERVATION: CODE/MATH SUB-MODELS

- Code and math sub-models experience a relatively modest drop in accuracy when evaluated on their counterpart domains (i.e., math models evaluated on code tasks and vice versa).



Performance of Gemma sub-models on Code



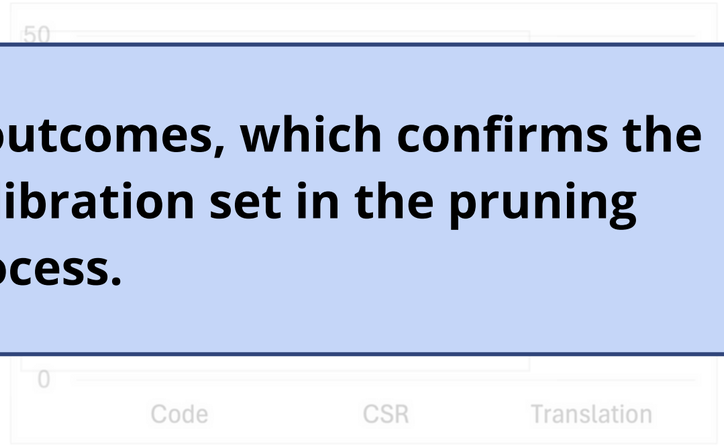
Performance of Gemma sub-models on Math

Both domains require logical thinking, problem-solving abilities, and a strong understanding of numerical concepts.

OBSERVATION: CODE/MATH SUB-MODELS

- Code and math sub-models experience a relatively modest drop in accuracy when evaluated on their counterpart domains (i.e., math models evaluated on code tasks and vice versa).

Performance of Gemma sub-models on Code



Performance of Gemma sub-models on Math

Similar tasks yield similar outcomes, which confirms the important role of the calibration set in the pruning process.

Both domains require logical thinking, problem-solving abilities, and a strong understanding of numerical concepts.

EVALUATION ON CODING SUB-DOMAINS

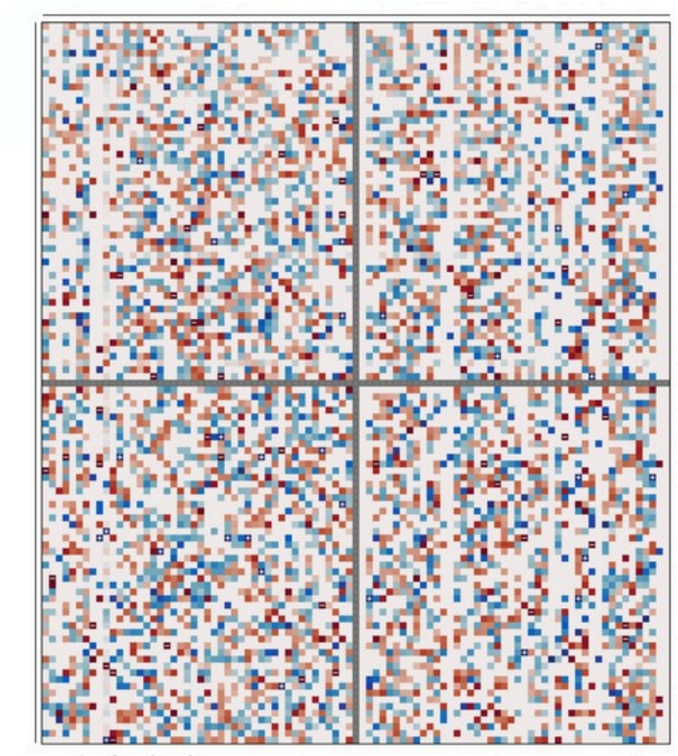
Programming Language	Calibr Set	Evaluation Metrics			
		HE (Python)	HE (Java)	HE (C++)	HE (JavaScript)
None	None	~71%	~68.3%	~64%	~67.7%
Python	py_alpaca	**~61%**	~55%	~54.2%	~59%
Java	code-java	~56.1%	**~60%**	~54.2%	~55.6%
C++	cpp_dataset	~56%	~54%	**~57.1%**	~56.3%
Javascript	code-javascript	~56.7%	~55.7%	~50.9%	**~60.2%**

EVALUATION ON CODING SUB-DOMAINS

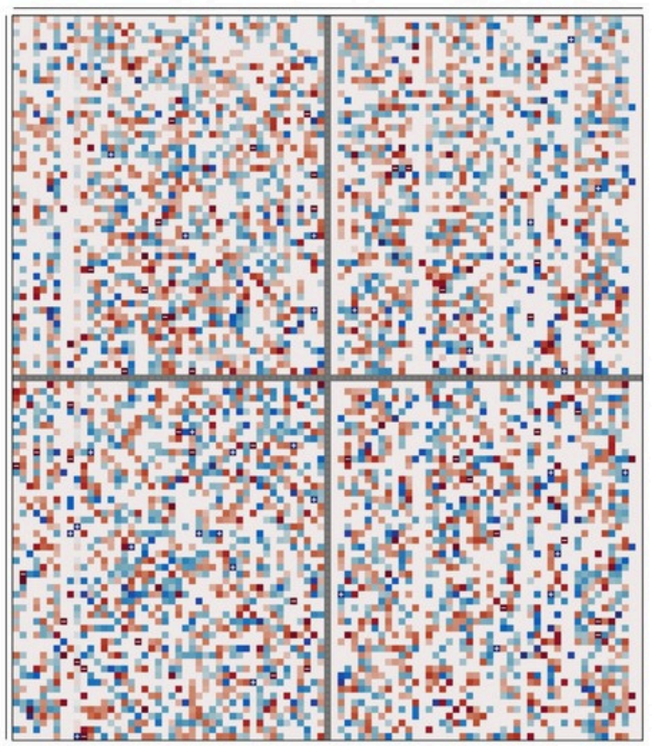
Programming Language	Calibr Set	Evaluation Metrics			
		HE (Python)	HE (Java)	HE (C++)	HE (JavaScript)
None	None	~71%	~68.3%	~64%	~67.7%
Python	py_alpaca	**~61%**	~55%	~54.2%	~59%
Java	code-java	~56.1%	**~60%**	~54.2%	~55.6%
C++	cpp_dataset	~56%	~54%	**~57.1%**	~56.3%
Javascript	code-javascript	~56.7%	~55.7%	~50.9%	**~60.2%**

SUB-MODELS COMPARISON (1)

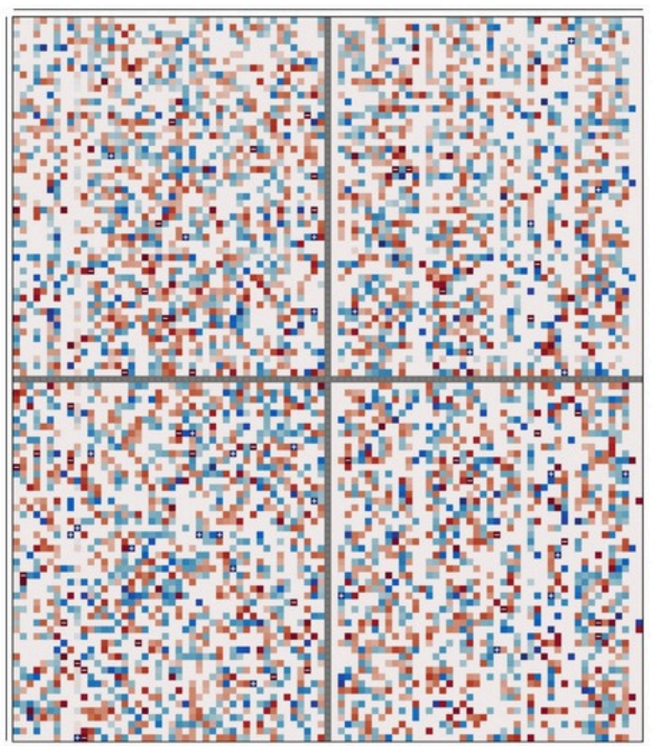
Weight matrices of sub-models



math_orca sub-model

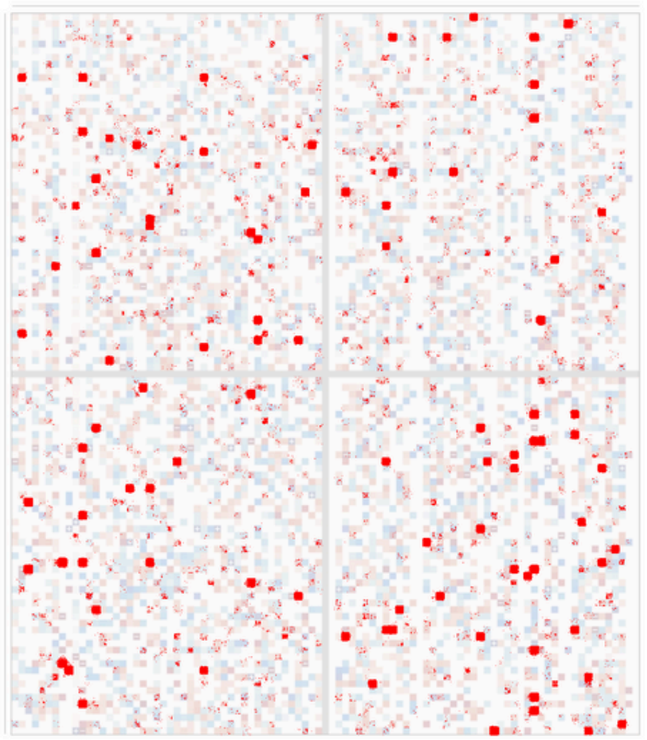


math_qa sub-model

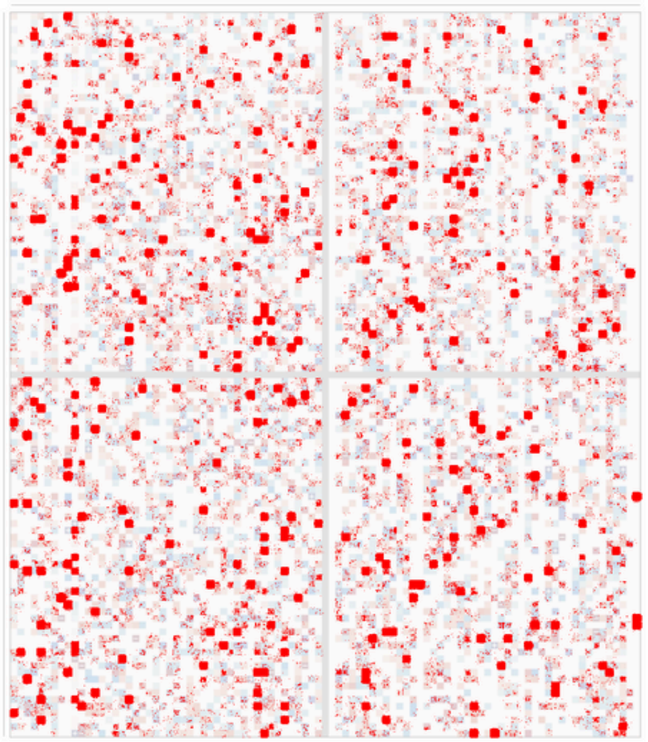


opus_books sub-model

SUB-MODELS COMPARISON (2)

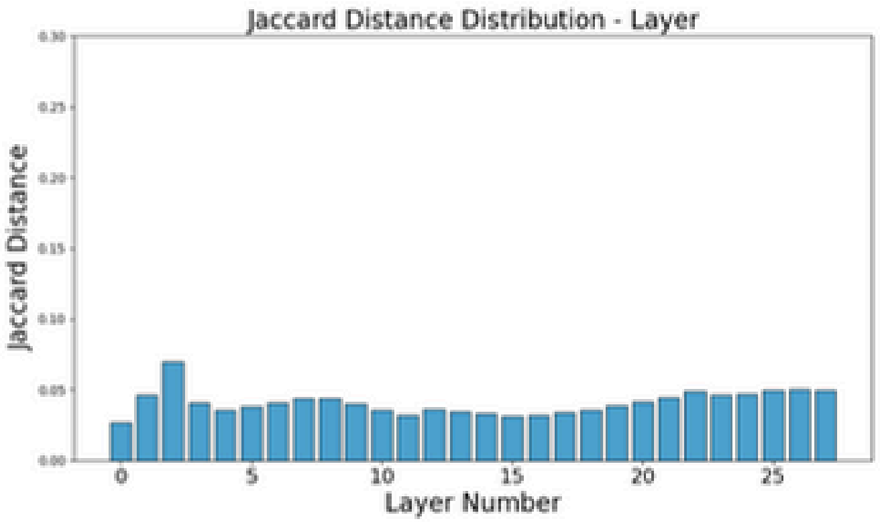


Differences between **same-domain** sub-models

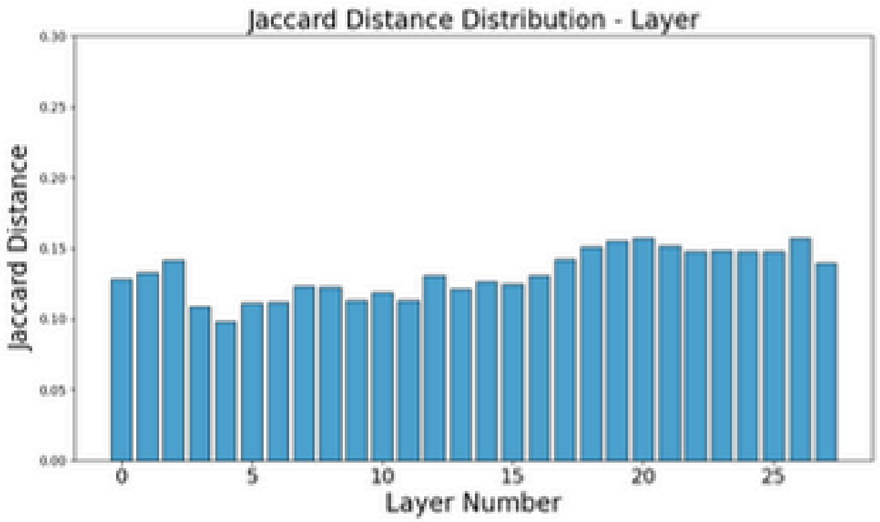


Differences between **different-domain** sub-models

PER-LAYER JACCARD DISTANCE

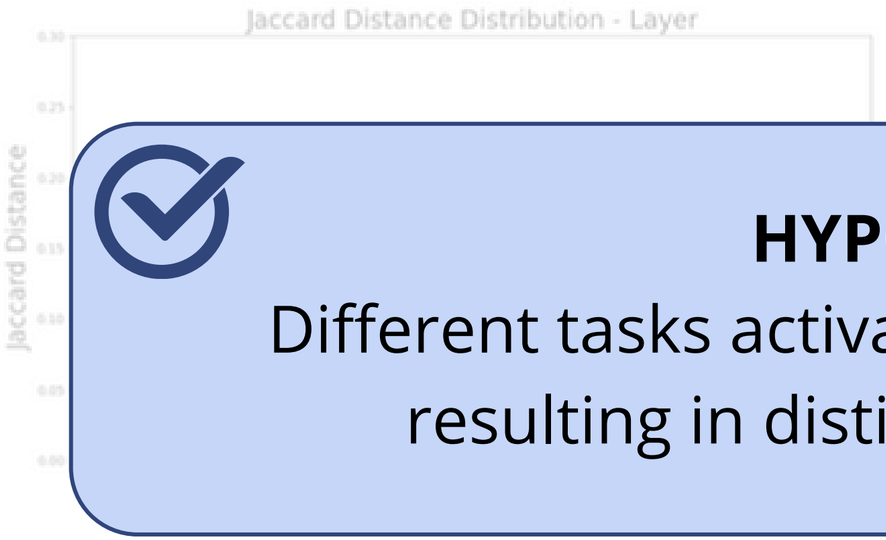


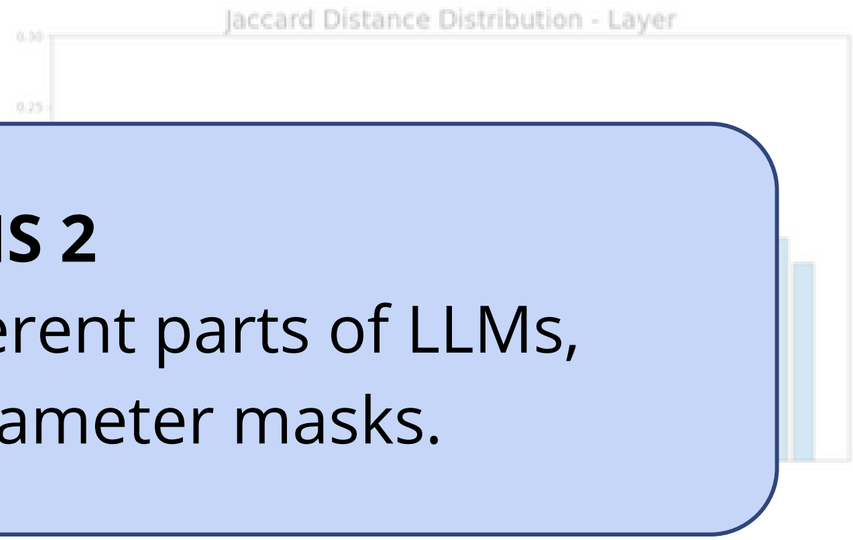
math_orca sub-model
vs
math_qa sub-model



math_orca sub-model
vs
opus_books sub-model

PER-LAYER JACCARD DISTANCE





HYPOTHESIS 2

Different tasks activate different parts of LLMs, resulting in distinct parameter masks.

math_orca sub-model
vs
math_qa sub-model

math_orca sub-model
vs
opus_books sub-model

INFERENCE SPEED-UP

Model	Layer	Dense	Sparse	Speedup
CodeLlama-I-13B	q_proj	0.449	0.329	1.365×
	k_proj	0.450	0.330	1.363×
	v_proj	0.450	0.330	1.361×
	o_proj	0.450	0.331	1.360×
	up_proj	1.014	0.998	1.016×
	gate_proj	1.010	1.000	1.010×
	down_proj	1.061	0.881	1.204×

INFERENCE SPEED-UP

Model	Layer	Dense	Sparse	Speedup
CodeLlama-I-13B	q_proj	0.449	0.329	1.365×
	k_proj	0.450	0.330	1.363×
	v_proj	0.450	0.330	1.361×
	o_proj	0.450	0.331	1.360×
	up_proj	1.014	0.998	1.016×
	gate_proj	1.010	1.000	1.010×
	down_proj	1.061	0.881	1.204×

Pruning models leads to inference speed-up, as expected.

CONCLUSIONS AND FUTURE WORK

KEY TAKEAWAYS

01 This thesis improved the original Wanda pruning technique.

KEY TAKEAWAYS

01 This thesis improved the original Wanda pruning technique.

02 The importance of using appropriate datasets for calibration was demonstrated.

KEY TAKEAWAYS

01 This thesis improved the original Wanda pruning technique.

02 The importance of using appropriate datasets for calibration was demonstrated.

03 Interesting insights into the choice of pruned weights depending on the used calibration set.

KEY TAKEAWAYS

01 This thesis improved the original Wanda pruning technique.

02 The importance of using appropriate datasets for calibration was demonstrated.

03 Interesting insights into the choice of pruned weights depending on the used calibration set.

POSSIBLE FUTURE WORKS

- Try different types of models and datasets.
- Evaluate the impact of sparsity.
- Propose a task-specific inference system.

THANK YOU

Q&A

BACKUP SLIDES

pass@k metric

Evaluates the **functional correctness of code:** a sample of generated code is considered correct if it **passes a set of unit tests.**

pass@k metric

Evaluates the **functional correctness of code:** a sample of generated code is considered correct if it **passes a set of unit tests.**

How it works:

- For a given problem, at least ***k*** code samples are generated by the LLM.

pass@k metric

Evaluates the **functional correctness of code:** a sample of generated code is considered correct if it **passes a set of unit tests.**

How it works:

- For a given problem, at least ***k*** code samples are generated by the LLM.
- The generated samples are assessed against some unit tests, the ***c*** samples that pass all units tests are considered correct.

pass@k metric

Evaluates the **functional correctness of code:** a sample of generated code is considered correct if it **passes a set of unit tests.**

How it works:

- For a given problem, at least ***k*** code samples are generated by the LLM.
- The generated samples are assessed against some unit tests, the ***c*** samples that pass all units tests are considered correct.
- The evaluation score is:

$$\text{pass @ } k := \mathop{\mathbb{E}}_{\text{Problems}} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

METEOR

Unigram matching between the machine-produced translation and human-produced reference translations.

METEOR

Unigram matching between the machine-produced translation and human-produced reference translations.

How evaluations are made:

- Given a test sample, the LLM is asked to translate it **five** times.

METEOR

Unigram matching between the machine-produced translation and human-produced reference translations.

How evaluations are made:

- Given a test sample, the LLM is asked to translate it **five** times.
- **METEOR accuracy** is then calculated independently for each of these five translations by comparing them to the reference translation of the test.

METEOR

Unigram matching between the machine-produced translation and human-produced reference translations.

How evaluations are made:

- Given a test sample, the LLM is asked to translate it **five** times.
- **METEOR accuracy** is then calculated independently for each of these five translations by comparing them to the reference translation of the test.

$$Fmean = \frac{10PR}{R + 9P} \qquad Penalty = 0.5 * \left(\frac{\#chunks}{\#unigrams\_matched} \right)^3$$

$$Score = Fmean * (1 - Penalty)$$

METEOR

Unigram matching between the machine-produced translation and human-produced reference translations.

How evaluations are made:

- Given a test sample, the LLM is asked to translate it **five** times.
- **METEOR accuracy** is then calculated independently for each of these five translations by comparing them to the reference translation of the test.
- The **highest score** among these five translations is retained.

METEOR

Unigram matching between the machine-produced translation and human-produced reference translations.

How evaluations are made:

- Given a test sample, the LLM is asked to translate it **five** times.
- **METEOR accuracy** is then calculated independently for each of these five translations by comparing them to the reference translation of the test.
- The **highest score** among these five translations is retained.
- This procedure is repeated for 400 different test samples, and the **average accuracy** serves as the final accuracy.

METEOR

Unigram matching between the machine-produced translation and human-produced reference translations

How eva

- Given a
- METEOR accuracy is then calculated independently for each of these five translations by comparing them to the reference translation of the test
- The **hig**
- This pr **racy** serves as the final ac

Incorporates both **precision** and **recall**, therefore being more accurate than other metrics.

Supports also matches between words that are **morphological variants** of each other or **synonyms**.

METEOR (example)

Source text: “La volpe marrone veloce salta sopra al cane pigro”

Reference translation: “The quick brown fox jumps over the lazy dog”

Translation attempt 1: “The brown fox jumps over the dog” **[score: 0.783]**
Translation attempt 2: “The brown fox” **[score: 0.350]**
Translation attempt 3: “The quick brown fox and the dog” **[score: 0.680]**

Final score: 0.783

...

Repeat this for every sentence in the evaluation dataset and average the scores.

N-SHOT ACCURACY

The accuracy of a model in correctly answering to new examples after being exposed to **n similar (but not equal) examples**.

N-SHOT ACCURACY

The accuracy of a model in correctly answering to new examples after being exposed to **n similar (but not equal) examples**.

How evaluations are made:

- The query sent to the model is preceded by N similar examples of query + answer.

N-SHOT ACCURACY

The accuracy of a model in correctly answering to new examples after being exposed to **n similar (but not equal) examples**.

How evaluations are made:

- The query sent to the model is preceded by N similar examples of query + answer.
- The model generates the answer

N-SHOT ACCURACY

The accuracy of a model in correctly answering to new examples after being exposed to **n similar (but not equal) examples**.

How evaluations are made:

- The query sent to the model is preceded by N similar examples of query + answer.
- The model generates the answer
- Accuracy is calculated on the answers given:

$$\text{Accuracy} = \frac{\text{\# correct answers}}{\text{\# total answers}}$$

N-SHOT ACCURACY (example)

"Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.

Example Query

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: "

Actual Query

CodeLlama-13B-Instruct

Domain	Calibration Set	Evaluation Metrics			
		Code (pass@10)	CSR (0-shot)	Translation (meteor)	MATH (8-shot)
None	None	~71.00%	~54.7%	~51.47%	~26.76%
Code	py_alpaca	**~61.00%**	~51.3%	~44.01%	~19.2%
	apps	~60.7%	~51.1%	~42.85%	~16.1%
	github_py	~60.5%	~51.6%	~43.1%	~16.9%
CSR	OB_qa	~54.8%	~52.3%	~44.95%	~15.2%
	Hellaswag	~57.15%	**~52.5%**	~44.6%	~15.9%
	Csense_qa	~55.72%	~52.17%	~46.9%	~10.7%
Transl	wmt14	~57.55%	~52.2%	**~49.5%**	~16.6%
	song_transl	~57.2%	~51.9%	~47.7%	~11.6%
	opus_books	~56.25%	~52.05%	~47.85%	~15.8%
Math	math_qa	~59.2%	~52.05%	~43.35%	**~21.68%**
	math_inst	~60.21%	~51.8%	~43.85%	~19.95%
	math_orca	~57.34%	~52.08%	~42.85%	~20.62%
Eng Text	c4	~51.4%	~51.5%	~42.3%	~15.39%

CodeLlama-7B-Instruct

Domain	Calibration Set	Evaluation Metrics			
		Code (pass@10)	CSR (0-shot)	Translation (meteor)	MATH (8-shot)
None	None	~60.5%	~52.3%	~47.48%	~17.81%
Code	py_alpaca	**~53.03%**	~48.41%	~38.39%	~8.7%
	apps	~52.51%	~47.06%	~36.38%	~7.4%
	github_py	~50.7%	~48.55%	~37.79%	~9.1%
CSR	OB_qa	~45.7%	~48.4%	~37.5%	~7.35%
	Hellaswag	~46.99%	**~49.3%**	~38.86%	~7.96%
	Csense_qa	~46.26%	~48.2%	~40.72%	~5.99%
Transl	wmt14	~48.19%	~49.1%	**~44.4%**	~7.2%
	song_transl	~44.11%	~48.28%	~42.6%	~6.22%
	opus_books	~46.95%	~48.06%	~43.5%	~7.05%
Math	math_qa	~51.02%	~48.32%	~38.95%	~10.02%
	math_inst	~49.38%	~48.63%	~37.9%	**~11.00%**
	math_orca	~49.26%	~48.67%	~40.9%	~9.63%
Eng Text	c4	~44.00%	~49.02%	~38.39%	~8.04%

Mistral-7B-Instruct

Domain	Calibration Set	Evaluation Metrics			
		Code (pass@10)	CSR (0-shot)	Translation (meteor)	MATH (8-shot)
None	None	~60.5%	~61.4%	~52.95%	~32.90%
Code	py_alpaca	**~47.28%**	~58.25%	~48.22%	~23.9%
	apps	~45.86%	~57.11%	~47.99%	~24.94%
	github_py	~46.87%	~57.99%	~47.07%	~24.33%
CSR	OB_qa	~46.28%	~58.2%	~48.9%	~21.9%
	Hellaswag	~44.15%	~57.6%	~46.86%	~22.1%
	Csense_qa	~40.4%	~58.2%	~46.93%	~20.54%
Transl	wmt14	~42.92%	~58.08%	**~51.31%**	~20.62%
	song_transl	~42.93%	~57.62%	~50.32%	~19.78%
	opus_books	~43.73%	~57.77%	~50.41%	~20.01%
Math	math_qa	~45.65%	~57.68%	~47.75%	**~27.82%**
	math_inst	~45.26%	**~58.26%**	~47.9%	~27.14%
	math_orca	~47.18%	~57.95%	~48.3%	~26.61%
Eng Text	c4	~42.5%	~57.8%	~47.45%	~22.5%