CodeGen2: Lessons for Training LLMs on Programming and Natural Languages

Erik Nijkamp[*], **Hiroaki Hayashi**[*], **Caiming Xiong**, **Silvio Savarese**, **Yingbo Zhou**

Salesforce Research

Abstract

Large language models (LLMs) have demonstrated remarkable abilities in representation learning for program synthesis and understanding tasks. The quality of the learned representations appears to be dictated by the neural scaling laws as a function of the number of model parameters and observations, while imposing upper bounds on the model performance by the amount of available data and compute, which is costly.

In this study, we attempt to render the training of LLMs for program synthesis more efficient by unifying four key components: (1) model architectures, (2) learning methods, (3) infill sampling, and, (4) data distributions. Specifically, for the model architecture, we attempt to unify encoder and decoder-based models into a single prefix-LM. For learning methods, (i) causal language modeling, (ii) span corruption, (iii) infilling are unified into a simple learning algorithm. For infill sampling, we explore the claim of a ”free lunch” hypothesis. For data distributions, the effect of a mixture distribution of programming and natural languages on model performance is explored.

We conduct a comprehensive series of empirical experiments on 1B LLMs, for which failures and successes of this exploration are distilled into four lessons. We will provide a final recipe for training and release CodeGen2 models in size 1B, 3.7B, 7B, and, 16B parameters, along with the training framework as open-source: `https://github.com/salesforce/CodeGen2`.

1 Introduction

1.1 Motivation: Cost of LLMs

Large language models (LLMs) have demonstrated strong empirical performance in a myriad of tasks across domains. In recent work (Chen et al., 2021b; Nijkamp et al., 2022; Fried et al., 2022; Allal et al., 2023), these findings have been transferred from natural to programming languages and achieved impressive performance in program synthesis and understanding tasks. The appeal of these models stems from three properties: (1) simple - the involved architectures are of low technical complexity due to reliance on self-attention circuits, (2) universal - that is a single model can handle a variety of different tasks rather than n specialized models for n tasks, which dramatically reduces resource and cost requirements, and, (3) scale - that is neural scaling laws dictate the performance of the model as a function of the amount of model parameters, data, compute in the form of power laws, that is, larger models usually yield predictably improved performance on down-stream tasks.

However, these merits obscure unresolved challenges: (1) while the self-attention circuit is technically simple, one has to pick an attention masking scheme to learn either bi-directional representations (encoder) or uni-directional representations (decoder), (2) while transformers appear task-agnostic, synthesis and understanding tasks have not been unified, (3) while improving performance with scale is attractive, even training a small set of models for different tasks incurs a significant cost. For the practitioner, the choices of model architecture, learning algorithm, and data distributions are not obvious. Exploration of these choices induces high monetary cost stemming from compute requirements.

[*] Equal contribution.

1.2 Goals: Reduce Cost by Unification and Open-source

To address the concerns of monetary cost and choice of variants, we attempt to unify (1) model architecture, (2) learning objective, (3) left-to-right and infill sampling, and (4) data distributions into a single recipe, which yields a single universal model with competitive performance on a wide range of synthesis and understanding tasks.

To approach the unification of these aspects in a principled manner, we postulate and evaluate the following hypotheses:

(1) **Model Architecture:** Encoder and decoder representations can be unified into a Prefix-LM (Raffel et al., 2020) for which the bi-directional self-attention is beneficial for harder few-shot tasks without degradation in performance over standard causal-decoders.

(2) **Learning Algorithm:** A mixture of objectives for causal language modeling and span-corruption yields efficient information transport for zero-shot learning (decoder) and understanding tasks (encoder).

(3) **Sampling Procedure:** Equipping a model with both left-to-right and infill sampling, under the assumption of the ”free lunch” hypothesis, does not increase computational cost.

(4) **Data Distributions:** A mixture of natural and programming languages simultaneously benefits tasks in both domains without affecting performance within a single modality.

The goals of this work are (i) to share lessons and provide a unified recipe to train such a universal model, (ii) to open-source implementation of the training procedure, and, (iii) to open-source a family of well-trained models.

1.3 Findings: A Mixed Bag of Results

In our attempt to achieve these goals, our method is to attempt full unification across all aspects and collect evidence to guide the ablation of features. We attempt to provide evidence to reject or not reject these hypotheses in an extensive study across a large set of experiments on 1B LLMs. The findings for our postulated hypotheses are summarized as follows:

(1) **Model Architecture:** We failed to provide evidence to quantify any benefits of Prefix-LM over the causal-decoder baseline within our set of evaluation tasks.

(2) **Learning Algorithm:** We succeeded in a simple mixture of objective functions while maintaining zero-shot performance.

(3) **Sampling Procedure:** We failed to provide evidence for the ”free lunch” hypothesis of equipping models with infill sampling without incurring an additional cost in compute.

(4) **Data Distributions:** We show promising evidence of mixing natural and programming languages into a single model.

While we did not achieve full unification, we gained valuable findings and trained competitive open-source models on permissive data.

1.4 Contributions: Lessons, Recipe, and Open-Source

We share these findings distilled with the following contributions:

- **Four lessons:** Distillation of results on (1) Prefix-LM as an architecture, (2) Free-lunch hypothesis of infill sampling, (3) Choice of objective functions, and, (4) Data mixture of natural and programming language,
- **Simple mixture objective:** We propose a simple, unified mixture of uncorrupted and within-file span-corruption sequences with next-token-prediction which yields competitive performance for both left-to-right and fill-in-the-middle auto-regressive sampling,
- **Open-Source implementation:** We will provide a well-engineered and tested reference implementation for LLM training of the final recipe,
- **Open-Source models:** We will open-source the CodeGen2 family of infill-capable models trained solely on permissive data once training for larger LLMs has converged.

1.5 Related Work

LLMs on Code Transformers capture dependency among sequence elements through attention mechanism (Bahdanau et al., 2014) and are highly scalable, as shown in natural language processing (Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2020). Several efforts explore these models for program synthesis (Chen et al., 2021a; Austin et al., 2021; Li et al., 2022; Fried et al., 2022; Nijkamp et al., 2022; Allal et al., 2023) and its effectiveness (Vaithilingam et al., 2022).

Ablation Studies Raffel et al. (2020) introduce the concept of non-causal decoder in the form of a Prefix-LM with favorable performance over causal decoders after fine-tuning on down-stream tasks. The performance in few-shot generative tasks was not evaluated. Wang et al. (2022) conduct an extensive ablation study over architectures and objectives with the conclusion that decoder-only models with causal language modeling exhibit the strongest zero-shot generalization. Therefore, we limit our investigation to causal and non-causal decoders. Tay et al. (2022a) compare encoder-decoder, decoder-only, and Prefix-LM architectures and report the beneficial performance of encoder-decoder models, while zero-shot generation tasks are not evaluated. The authors later adopt Prefix-LM instead of encoder-decoder in (Tay et al., 2022b).

Data Mixtures LaMDA (Thoppilan et al., 2022) was trained on a mixture of various data sources including dialogues, code documents, Q&A data, tutorials, and, Wikipedia. However, the impact of this mixture and the specific sources are unclear. Xie et al. (2023) introduces a data selection method based on importance resampling which allows to mix datasets of various sizes, however, the evaluation only covers encoder-only models.

2 Method: From Unification to Ablation

In this Section, requirements for our goals are defined along with relevant components for ablation.

2.1 Requirement: Performance on a Variety of Tasks

We aim to render the training of LLMs for program synthesis more efficient by providing a unification of both learning methods and model architectures, while maintaining (or improving) performance of individual tasks under identical compute budget. The set of tasks is as follows:

(1) **Program Synthesis with left-to-right sampling (zero-shot)** The prompt as an intent specification is in the form of a function signature and doc-string. A program is conditionally sampled (or completed) based on the prompt in left-to-right auto-regressive fashion. The HumanEval (Chen et al., 2021b) is recruited to evaluate the quality of synthesized programs. Specifically, for prompt (a), we sample $b \sim P(b|a)$.

(2) **Program Synthesis with infill sampling (zero-shot)** The prompt includes both past and future tokens for which the tokens "in-between" are supposed to be sampled. For instance, the prompt includes the first and last few lines of a function definition, while the body of the function is to be filled in. Specifically, for prompt (a, c), we sample $b \sim P(b|a, c)$. The HumanEval-Infill (Fried et al., 2022) benchmark is recruited for evaluation.

(3) **In-context Learning from examples (few-shot)** A task is defined given a set of examples (or "shots"). Specially, for n few-shot examples $((x_1, y_1), \ldots, (x_n, y_n))$ with code x and label y, we sample label $y_{n+1} \sim P(y|(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}))$. The XSum benchmark (Narayan et al., 2018) is recruited for evaluation.

(4) **Program Understanding with bi-directional representations (fine-tune)** Causally-masked decoder models are constrained to auto-regressive sampling in left-to-right fashion. For understanding tasks, such as defect detection (Lu et al., 2021), removing this constraint over time such that the representations can be a function of all input tokens simultaneously seems desirable. Such representations are obtained from bi-directional language models without causal masking. The CodeXGLUE (Lu et al., 2021) and SuperGLUE (Wang et al., 2019) benchmarks are recruited for evaluation.

2.2 Components: Architecture, Objective, Sampling, Data

Model Architecture In representation learning with transformers (Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2020), two schemes of modeling are prevalent which differ in their attention masks for the contextualization of hidden vectors. For a sequence $x = (x_1, \ldots, x_n)$ of n vectors, we differ: (1) bi-directional encoder-based representations in which each token vector x_i can attend all other tokens $\{x_j : i = 1, \ldots, n\}$, (2) uni-directional decoder-based representations in which each token vector x_i can only attend previous tokens $\{x_j : j \leq i\}$. While encoder-based representations for which each hidden vector can contextualize with all other vectors may be desirable for understanding tasks, decoder-based representations with temporal causal masking are required for language modeling for which the joint density is factorized as the product of conditionals over time steps. To unify both schemes, we adopt the notion of prefix-based language modeling (Prefix-LM) (Raffel et al., 2020). For a prefix, we decompose the input sequence x into a prefix p and a context c. For the prefix $p = (x_1, \ldots, x_m)$ where $m < n$, each token can attend over all other tokens in the prefix, which amounts to bi-directional representations. For the context $c = (x_{m+1}, \ldots, x_n)$, each token can only attend to previous tokens, which amounts to uni-directional decoder representations. This unifies bi-directional attention over the prefix with the requirement of causal masking to factorize the joint density over time. The hope is to achieve competitive auto-regressive sampling for synthesis tasks, while learning strong bi-directional representations for understanding tasks.

Learning Algorithm The choice of encoder or decoder-based model architectures typically guides the selection of learning algorithms for language modeling. Encoder-based models may be trained with the task of masked language modeling in the form of denoising span corruptions (Devlin et al., 2019; Raffel et al., 2020). Decoder-based models may be trained as density language modeling in the form of a next-token-prediction task (Radford et al., 2018). For encoder-based models, the prevalent algorithms are variants of token reconstructions or denoising task for which spans of tokens undergo a corruption or perturbation. For a sequence $x = (x_1, \ldots, x_n)$ of n tokens, a perturbation $\tilde{x} = (x_1, m_1, x_5, x_6, m_2, x_7, \ldots, x_n)$ replaces spans of tokens with special mask tokens $(m_1, m_2, \ldots)$. The learning task is to recover the original sequence x from the perturbation $\tilde{x}$. Denoising-based learning objectives have been shown to be highly efficient for language understanding tasks. For decoder-based models, the prevalent algorithm is maximum likelihood-based learning of causal language modeling in the form of a next token prediction task (Radford et al., 2018). For a sequence $x = (x_1, \ldots, x_n)$ of n tokens, the task is to predict the token x_i given previous tokens $(x_j : j < i)$. In this work, we explore a learning algorithm as a mixture of both causal language modeling objectives and span corruption. We postulate for such a mixture the task-specific prior information should be minimized to avoid over-fitting to specific tasks. That is, ideally, the distributions over mixture ratio of tasks, length of the prefixes, and length of spans are uniform.

Sampling Procedure Program synthesis in the form of auto-regressive sampling from a language model has been established as a predominant method. While left-to-right sampling can only take previous tokens into account, often when editing existing files of code, conditioning the sampling on context before and after the current position within a file is desirable. Several variants (Du et al., 2022; Fried et al., 2022; Bavarian et al., 2022) which rearrange a sequence (a, b, c) into $(a, \textit{<mask>}, c, \textit{<end>}, b)$ such that the infilling b given (a, c) can be learned by standard next-token-prediction objectives in causally masked decoders. Bavarian et al. (2022) claim under the "free lunch" hypothesis that training LLMs with such modified training observation does not incur any additional cost in compute or degradation in performance on zero-shot generation tasks.

Data Distribution In maximum likelihood learning, a model is learned by minimizing some measure of divergence between the data and model distribution. Surprisingly, when increasing the number of observations and model parameters, few-shot abilities emerge when sampling from the fitted densities (Brown et al., 2020; Wei et al., 2022). For program synthesis, Nijkamp et al. (2022) demonstrate sampling executable code in a multi-turn conversational scheme, similar to (Ouyang et al., 2022), but without explicit instruction fine-tuning. It is hypothesized that these abilities emerge from weak supervision in the data. Programs often include functions with accompanying English descriptions. We attempt to increase the amount of natural language to further improve this ability and explicitly create a mixture of natural and programming languages. Further, the resulting model may be competitive on down-stream tasks in both domains.

3 Results: Lessons and Recipe

In this Section, we present the empirical results and conclusions of the attempt toward unification and ablation study for which the findings are distilled into the following four lessons.

3.1 Lesson 1: Prefix-LM's benefit is questionable

As discussed, Prefix-LM behaves as an encoder with bi-directional attention, and as an autoregressive decoder with a causal mask. However, it is unclear if this unification of the architecture leads to competitive performance on both ends. We evaluate this question from three angles: data, representation, and objective. In the following, we refer to the first half of the sequence that is covered by bi-directional attention as *non-causal* part, and the rest of the sequence as *causal* part.

3.1.1 Data

Context and Hypothesis When training a Prefix-LM with next token prediction, the loss for the non-causal part is masked because the prediction is informed by the future tokens that are used when encoding. Depending on the length of the non-causal part in each sequence, this means the reduction of the effective number of tokens responsible for the gradient update, which raises the question of whether or not the prefix mask negatively impacts the learning of Prefix-LM. Specifically, we hypothesize that the lack of gradient on the non-causal part results in worse results on the code generation task due to the slower rate of information transport via NTP.

Results and Findings To evaluate this hypothesis, we first train a Prefix-LM on BigPython (Nijkamp et al., 2022) with causal language modeling and compare the model performance on HumanEval against a causal decoder baseline trained for the same number of steps. The length of the non-causal part is set to rN, where $r \sim [0, 0.9]$ and N is the sequence length. Surprisingly, except for the first 100,000 steps, we observe that the HumanEval scores for Prefix-LM are on-par with the causal decoder for the majority of pretraining, not exhibiting a tendency of slower learning. In addition, Prefix-LM trained with the combination of causal language modeling and span corruption yields competitive infill capability, relative to InCoder (Fried et al., 2022).

Next, we repeat the same experiment on the Stack (Kocetkov et al., 2022), with Python being 9.4% of all tokens in the dataset. We refer to Table 2 for HumanEval and HumanEval-Infill results. Contrary to the promising result above, however, we observed strictly worse pass@k throughout the pretraining, resulting in 2 points worse than the causal decoder baseline.

Based on these conflicting observations, we speculate that a negative effect due to the lack of enough gradient update for the target language (*e.g.* Python) exists in Prefix-LM, which did not surface earlier thanks to training exclusively on Python for a large number of steps. It still remains a question as to whether this effect surfaces at a certain threshold or gradually.

3.1.2 Representation

Context and Hypothesis Prefix-LM is appealing due to the bi-directional attention, which allows for the model to contextualize hidden states with both past and future tokens. Controlled by only the existence of an attention mask, the ease of switching the role as an encoder or decoder allows for trivial unification, if the resulting model is competitive. Based on the recent success with Prefix-LM (Tay et al., 2022b), we hypothesize that the trained Prefix-LM yields a competitive encoder representation that can be used for a range of discriminative tasks, while retaining generative capability as the decoder. To test this hypothesis, we train Prefix-LM with a mixture of causal language modeling and span corruption, given that denoising yields strong representations (Raffel et al., 2020; Tay et al., 2022a). Aiming for wider coverage of tasks, we train the model on both programming and natural language and evaluate each model on the defect detection task from CodeXGLUE (Lu et al., 2021) and 4 tasks from SuperGLUE (Wang et al., 2019), respectively. We attach a classification head on top of the hidden state at a certain time step (first or last token) to adapt to each task. In addition to the finetuning-based discriminative tasks, we examine if the in-context learning ability is enhanced by bi-directional attention. We follow UL2 and evaluate the models trained on natural language on XSum as a few-shot generation task. We examine the change in performance by increasing the number of examples fed into the model as context.

Model	Size	HumanEval	HumanEval-Infill	
			SingleLine	MultiLine
Incoder	1.3B	9.79	52.56	23.85
	6.7B	15.20	66.69	34.62
CodeGen2	1B	10.27	53.41	23.41
	3.7B	14.88	65.72	33.45
	7B	19.09	68.74	38.88

Table 1: Pass@1 on HumanEval. For HumanEval-Infill, only end-of-mask truncation is used.

Results and Findings In Table 2, Prefix-LM generally achieves better results over causal decoder trained with the same objective on our SuperGLUE tasks, which partially answers our question that bi-directional attention may yield informative representations. However, our 1B parameter model could not compete with much smaller encoder-only pre-trained models such as CodeBERT on CodeXGLUE or RoBERTa-large on SuperGLUE, which leads us to conclude that the representations are not sufficiently informative to justify the substitution of smaller scale encoder-only pre-trained models with one Prefix-LM. For few-shot XSum, we did not observe meaningful differences between the two models, regardless of the number of exemplars in the non-causal part.

3.1.3 OBJECTIVE

Context and Hypothesis As discussed, Prefix-LM appears to be effective when trained with UL2 (Tay et al., 2022a), which proposes to combine causal language modeling (CLM) and denoising with 6 different hyperparameters. The objective achieves significant gain over CLM and T5-style span corruption on a range of tasks under different model scales. In the hope of achieving similarly competitive results in the programming language domain, we train Prefix-LM with the UL2 objective (including the task tokens), but find that the trained model is significantly worse on HumanEval by 9 points on pass@1 compared to the CLM baseline. We hypothesize that this unexpectedly under-performing result is due to the small ratio of non-corrupted sequences used by one of the UL2 objectives, S-denoiser, which on average treats 75% of a sequence as the prefix.

Results and Findings We verify the effect of the prefix length by simplifying UL2's denoiser hyperparameters such that (1) the percentage of S-denoiser is higher (14% to 50%) and (2) the average prefix length for the S-denoiser is shorter (75% to 50%), and observe consistent improvement on HumanEval over the original UL2 hyperparameters. In general, we conclude that causal language modeling on non-corrupted sequences is aligned with the zero-shot generation task in HumanEval, which requires sampling at the end of prefix, and therefore we maximize the number of tokens in the sequence used for gradient updates to learn a competitive model.

3.2 LESSON 2: INFILL IS NOT A FREE LUNCH

Context and Hypothesis The process of writing code involves editing tokens within the file, not only at the end. Among several works that have attempted to equip the model with this infill ability, Bavarian et al. (2022) claims to achieve infill with no degradation in left-to-right sampling at the end of context by carefully permuting a portion of sequences from training data, getting the infill ability "for free." On the other hand, concurrent work reports that infill is "cheap", *i.e.* there is a consistent performance drop in left-to-right sampling in HumanEval, if the model is trained on the infilling objective (Allal et al., 2023). While conflicting results have been reported, we hypothesize that indeed infill is for free, if trained following the methods by Bavarian et al. (2022), and attempt to reproduce infill learning with our models.

Results and Findings To verify the free-lunch hypothesis, we train a causal decoder with a mixture of CLM and PSM (Prefix, Suffix, Middle sequence reordering) infilling objective, following the experimental setting in Bavarian et al. (2022). However, we observed about 1 point decrease in pass@1 in HumanEval performance compared to the causal decoder baseline trained only with causal language modeling. Thus, our observation follows that of (Allal et al., 2023) in that infill is not for free, and we leave careful reimplementation to reproduce "infill for free" a future work.

Model	HumanEval pass@1	Infill-Single pass@1	Infill-Multi pass@1	XSum R-L	SuperGLUE BoolQ (Acc)	CodeXGLUE Defect (Acc)
Decoder	7.99	43.37	17.40	10.28	0.774	0.635
Prefix-LM	6.71	39.11	17.25	9.78	0.806	0.640

Table 2: Comparison of causal decoder and Prefix-LM on various tasks. HumanEval, Infill-Single, and Infill-Multi are evaluated as zero-shot code generation, XSum is evaluated as 1-shot summarization, SuperGLUE (BoolQ) and CodeXGLUE (defect detection) are evaluated by fine-tuning.

3.3 Lesson 3: Objective can be Simple, yet Needs to be Carefully Chosen

Context and Hypothesis So far, we learn from the lessons above that (1) Prefix-LM is not an ideal architecture due to poor multilingual performance and sub-par utility of representations, (2) infill is not for free, but HumanEval performance is close to that of a non-infill model. To train our causal decoder model with competitive left-to-right and infill sampling capability, we align the sequence format (e.g., no task tokens), remove complexity, and assume uniform distributions over task mixture and span lengths to avoid bias relative to (Tay et al., 2022a). We choose span corruption as the base infill objective following InCoder (Fried et al., 2022). However, we take a different approach in selecting the spans for corruption: (1) we first sample a dynamic ratio of sequence to mask out, (2) we then sample the span length and mask out locations such that the total number of tokens match the ratio of the original sequence determined earlier. Additionally, we introduce two changes below:

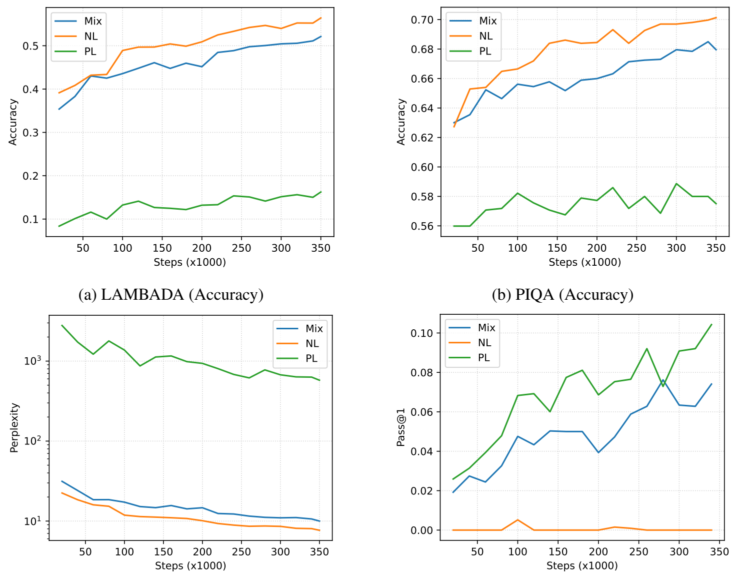
- **Mixed Objective.** While InCoder learns to predict the next tokens within subsequences appearing in the span-corrupted sequence, we incorporate a dedicated causal language modeling objective for a portion of sequences. For each sample, the choice of span corruption or causal language modeling objective is decided with $p = 0.5$. We do not prefix the sequence with a task token as proposed by UL2, since no notable differences are observed with the presence of task tokens.
- **File-level Corruption.** Span corruption may accidentally mask the document boundaries, which leads to a malformed context that confuses the model. To avoid this, we apply span corruption at file-level, *i.e.*, if a sequence can be split into two subsequences because of a document boundary, apply the span corruption for each subsequence, and concatenate them together. This results in some sequences having multiple pairs of denoising instances.

Results and Findings We examine our recipe on four model sizes: 1B, 3.7B, 7B, and 16B, and refer to them as **CodeGen2**.[1] For training a subset of the Stack v1.1 (Kocetkov et al., 2022), filtered with a stronger permissive license guideline, is used. The models are evaluated on HumanEval (Chen et al., 2021a) and HumanEval-Infill (Fried et al., 2022; Bavarian et al., 2022), where we follow their truncation strategies. However, we note the discrepancy in experimental results for InCoder from the original paper, as we only use the end-of-mask token (`<eom>`) instead of heuristics to determine the end of infill sequence to measure the line-agnostic ability to infill and stop.

3.4 Lesson 4: Multi-modal Data Mixing

Context and Hypothesis With the growing interest in cross-modal applications between natural and programming languages due to the rise of code-aware conversational LLMs, there is a demand for tested training recipes on mixed textual modalities. To some extent, an implicit mixture of modalities exists in programming language data in the form of comments and documents, and in natural language data as snippets of code for online QA forums, which is a sufficient supervised learning signal for a large-scale LLM for code to exhibit an ability to generate code from text (Nijkamp et al., 2022). However, our preliminary experiment shows that such LLMs for code exhibit poor performance on natural language tasks, despite the implicit mixture. It is our hypothesis that the poor performance can be mitigated with a sufficient learning signal from a distribution of natural language. We test the hypothesis by training a causal decoder with the combined causal language modeling and span corruption objective on a mixture of natural and programming languages (**Mix**). Examples in a batch are sampled from the Pile (Gao et al., 2020) and the Stack data equally likely.

[1]As of submission, the 16B-parameter model is still under training. We will revise the manuscript and open-source the model once the training converges.



(a) LAMBADA (Accuracy)

(b) PIQA (Accuracy)

(c) LAMBADA (Perplexity)

(d) HumanEval (Pass@1)

Figure 1: Results on LAMBADA and PIQA for NL and HumanEval for PL over number of training steps.

Results and Findings The model is evaluated in a zero-shot manner on both programming and natural language downstream tasks over the training steps to assess the effect of data mixing on each modality; specifically, we report pass@1 for HumanEval, accuracy and perplexity for LAMBADA (Paperno et al., 2016), and accuracy for PIQA (Bisk et al., 2020). For comparison, we alter the training data with (1) only the Pile (Gao et al., 2020) (**NL**) or (2) only the Stack (Kocetkov et al., 2022) (**PL**), and trained for the same number of steps. Fig 1 depicts the results for these tasks. Consistently over the training steps, we observe that:

- **Mix** does not outperform other baselines for the evaluated tasks given the same compute budget. Since the exposure to each domain reduces by 50% under **Mix**, the model reasonably performs worse than the domain-matched counterparts (*e.g.*, **PL** for HumanEval).
- **Mix** performs closely to the domain-matched models. From early in the training, **Mix** substantially improves performance over the domain-mismatched baselines (*e.g.*, **PL** for LAMBADA), which suggests that one should mix natural and programming languages if (a) the compute budget is constrained and (b) the resulting model is used for both domains.

Nonetheless, we find the recipe promising that even a simple mixture of training data allows for efficient learning on both domains, which might suggest that a slightly longer training with **Mix** can yield one competitive model for both domains.

4 Conclusion: Lessons and Open-Source

The training of LLMs is costly and involves a myriad of design choices. Our goal was to address this challenge by unification across architecture, objectives, sampling procedures, and data distributions. We formed hypotheses for each of these aspects and distilled the positive and negative findings into four lessons. While we did not achieve satisfactory unification, the findings of this exploration and our final training recipe may be valuable for practitioners.

In summary, for our hypotheses, we believe (1) the Prefix-LM architecture does not yield any measurable improvements on our set of tasks, (2) training a model with infill sampling is not a free lunch, (3) a simple mixture of causal language modeling and span-corruption limited to within-file spans is sufficient, and (4) a mixture distribution of programming and natural languages looks promising.

To facilitate future research, we will open-source the training implementation and the resulting family of CodeGen2 models with the sizes of 1B, 3.7B, 7B, and 16B parameters for the community.

REFERENCES

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, et al. Santacoder: don't reach for the stars! *arXiv preprint arXiv:2301.03988*, 2023.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 320–335, 2022.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*, 2022.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, 2020.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode, Feb 2022.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*, 2021.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL `https://aclanthology.org/P16-1144`.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Neil Houlsby, and Donald Metzler. Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022a.

Yi Tay, Jason Wei, Hyung Won Chung, Vinh Q Tran, David R So, Siamak Shakeri, Xavier Garcia, Huaixiu Steven Zheng, Jinfeng Rao, Aakanksha Chowdhery, et al. Transcending scaling laws with 0.1% extra compute. *arXiv preprint arXiv:2210.11399*, 2022b.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pp. 1–7, 2022.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective works best for zero-shot generalization? In *International Conference on Machine Learning*, pp. 22964–22984. PMLR, 2022.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. *arXiv preprint arXiv:2302.03169*, 2023.