OUTRAGEOUSLY LARGE NEURAL NETWORKS: THE SPARSELY-GATED MIXTURE-OF-EXPERTS LAYER

Noam Shazeer[1], Azalia Mirhoseini*†[1], Krzysztof Maziarz*[2], Andy Davis[1], Quoc Le[1], Geoffrey Hinton[1] and Jeff Dean[1]

[1]Google Brain, {noam,azalia,andydavis,qvl,geoffhinton,jeff}@google.com
[2]Jagiellonian University, Cracow, krzysztof.maziarz@student.uj.edu.pl

ABSTRACT

The capacity of a neural network to absorb information is limited by its number of parameters. Conditional computation, where parts of the network are active on a per-example basis, has been proposed in theory as a way of dramatically increasing model capacity without a proportional increase in computation. In practice, however, there are significant algorithmic and performance challenges. In this work, we address these challenges and finally realize the promise of conditional computation, achieving greater than 1000x improvements in model capacity with only minor losses in computational efficiency on modern GPU clusters. We introduce a Sparsely-Gated Mixture-of-Experts layer (MoE), consisting of up to thousands of feed-forward sub-networks. A trainable gating network determines a sparse combination of these experts to use for each example. We apply the MoE to the tasks of language modeling and machine translation, where model capacity is critical for absorbing the vast quantities of knowledge available in the training corpora. We present model architectures in which a MoE with up to 137 billion parameters is applied convolutionally between stacked LSTM layers. On large language modeling and machine translation benchmarks, these models achieve significantly better results than state-of-the-art at lower computational cost.

1 INTRODUCTION AND RELATED WORK

1.1 CONDITIONAL COMPUTATION

Exploiting scale in both training data and model size has been central to the success of deep learning. When datasets are sufficiently large, increasing the capacity (number of parameters) of neural networks can give much better prediction accuracy. This has been shown in domains such as text (Sutskever et al., 2014; Bahdanau et al., 2014; Jozefowicz et al., 2016; Wu et al., 2016), images (Krizhevsky et al., 2012; Le et al., 2012), and audio (Hinton et al., 2012; Amodei et al., 2015). For typical deep learning models, where the entire model is activated for every example, this leads to a roughly quadratic blow-up in training costs, as both the model size and the number of training examples increase. Unfortunately, the advances in computing power and distributed computation fall short of meeting such demand.

Various forms of conditional computation have been proposed as a way to increase model capacity without a proportional increase in computational costs (Davis & Arel, 2013; Bengio et al., 2013; Eigen et al., 2013; Ludovic Denoyer, 2014; Cho & Bengio, 2014; Bengio et al., 2015; Almahairi et al., 2015). In these schemes, large parts of a network are active or inactive on a per-example basis. The gating decisions may be binary or sparse and continuous, stochastic or deterministic. Various forms of reinforcement learning and back-propagation are proposed for trarining the gating decisions.

*Equally major contributors
†Work done as a member of the Google Brain Residency program (g.co/brainresidency)

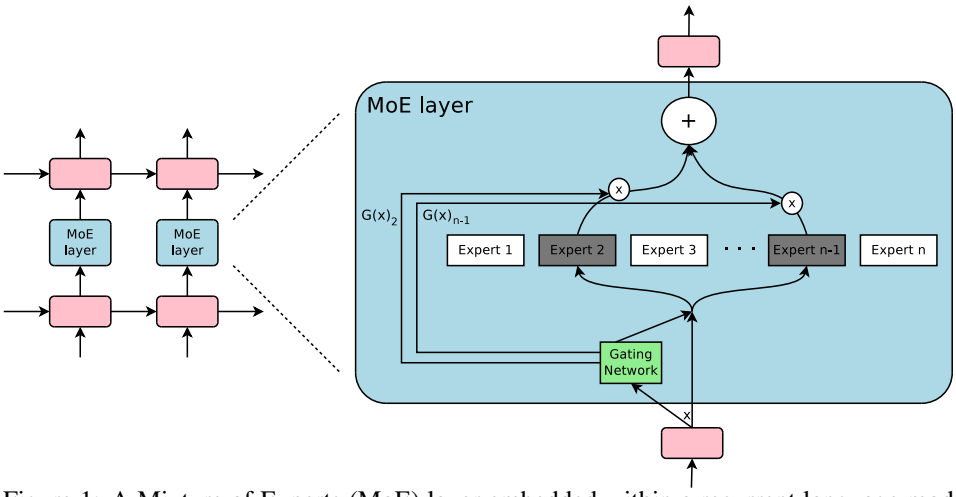

Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

While these ideas are promising in theory, no work to date has yet demonstrated massive improvements in model capacity, training time, or model quality. We blame this on a combination of the following challenges:

- Modern computing devices, especially GPUs, are much faster at arithmetic than at branching. Most of the works above recognize this and propose turning on/off large chunks of the network with each gating decision.
- Large batch sizes are critical for performance, as they amortize the costs of parameter transfers and updates. Conditional computation reduces the batch sizes for the conditionally active chunks of the network.
- Network bandwidth can be a bottleneck. A cluster of GPUs may have computational power thousands of times greater than the aggregate inter-device network bandwidth. To be computationally efficient, the relative computational versus network demands of an algorithm must exceed this ratio. Embedding layers, which can be seen as a form of conditional computation, are handicapped by this very problem. Since the embeddings generally need to be sent across the network, the number of (example, parameter) interactions is limited by network bandwidth instead of computational capacity.
- Depending on the scheme, loss terms may be necessary to achieve the desired level of sparsity per-chunk and/or per example. Bengio et al. (2015) use three such terms. These issues can affect both model quality and load-balancing.
- Model capacity is most critical for very large data sets. The existing literature on conditional computation deals with relatively small image recognition data sets consisting of up to 600,000 images. It is hard to imagine that the labels of these images provide a sufficient signal to adequately train a model with millions, let alone billions of parameters.

In this work, we for the first time address all of the above challenges and finally realize the promise of conditional computation. We obtain greater than 1000x improvements in model capacity with only minor losses in computational efficiency and significantly advance the state-of-the-art results on public language modeling and translation data sets.

1.2 Our Approach: The Sparsely-Gated Mixture-of-Experts Layer

Our approach to conditional computation is to introduce a new type of general purpose neural network component: a Sparsely-Gated Mixture-of-Experts Layer (MoE). The MoE consists of a number of experts, each a simple feed-forward neural network, and a trainable gating network which selects a sparse combination of the experts to process each input (see Figure 1). All parts of the network are trained jointly by back-propagation.

While the introduced technique is generic, in this paper we focus on language modeling and machine translation tasks, which are known to benefit from very large models. In particular, we apply a MoE convolutionally between stacked LSTM layers (Hochreiter & Schmidhuber, 1997), as in Figure 1. The MoE is called once for each position in the text, selecting a potentially different combination of experts at each position. The different experts tend to become highly specialized based on syntax and semantics (see Appendix E Table 9). On both language modeling and machine translation benchmarks, we improve on best published results at a fraction of the computational cost.

1.3 Related work on Mixtures of Experts

Since its introduction more than two decades ago (Jacobs et al., 1991; Jordan & Jacobs, 1994), the mixture-of-experts approach has been the subject of much research. Different types of expert architectures hae been proposed such as SVMs (Collobert et al., 2002), Gaussian Processes (Tresp, 2001; Theis & Bethge, 2015; Deisenroth & Ng, 2015), Dirichlet Processes (Shahbaba & Neal, 2009), and deep networks. Other work has focused on different expert configurations such as a hierarchical structure (Yao et al., 2009), infinite numbers of experts (Rasmussen & Ghahramani, 2002), and adding experts sequentially (Aljundi et al., 2016). Garmash & Monz (2016) suggest an ensemble model in the format of mixture of experts for machine translation. The gating network is trained on a pre-trained ensemble NMT model.

The works above concern top-level mixtures of experts. The mixture of experts is the whole model. Eigen et al. (2013) introduce the idea of using multiple MoEs with their own gating networks as parts of a deep model. It is intuitive that the latter approach is more powerful, since complex problems may contain many sub-problems each requiring different experts. They also allude in their conclusion to the potential to introduce sparsity, turning MoEs into a vehicle for computational computation.

Our work builds on this use of MoEs as a general purpose neural network component. While Eigen et al. (2013) uses two stacked MoEs allowing for two sets of gating decisions, our convolutional application of the MoE allows for different gating decisions at each position in the text. We also realize sparse gating and demonstrate its use as a practical way to massively increase model capacity.

2 The Structure of the Mixture-of-Experts layer

The Mixture-of-Experts (MoE) layer consists of a set of n "expert networks" $E_1, \cdots, E_n$, and a "gating network" G whose output is a sparse n-dimensional vector. Figure 1 shows an overview of the MoE module. The experts are themselves neural networks, each with their own parameters. Although in principle we only require that the experts accept the same sized inputs and produce the same-sized outputs, in our initial investigations in this paper, we restrict ourselves to the case where the models are feed-forward networks with identical architectures, but with separate parameters.

Let us denote by $G(x)$ and $E_i(x)$ the output of the gating network and the output of the i-th expert network for a given input x. The output y of the MoE module can be written as follows:

$$y = \sum_{i=1}^{n} G(x)_i E_i(x) \tag{1}$$

We save computation based on the sparsity of the output of $G(x)$. Wherever $G(x)_i = 0$, we need not compute $E_i(x)$. In our experiments, we have up to thousands of experts, but only need to evaluate a handful of them for every example. If the number of experts is very large, we can reduce the branching factor by using a two-level hierarchical MoE. In a hierarchical MoE, a primary gating network chooses a sparse weighted combination of "experts", each of which is itself a secondary mixture-of-experts with its own gating network. In the following we focus on ordinary MoEs. We provide more details on hierarchical MoEs in Appendix B.

Our implementation is related to other models of conditional computation. A MoE whose experts are simple weight matrices is similar to the parameterized weight matrix proposed in (Cho & Bengio, 2014). A MoE whose experts have one hidden layer is similar to the block-wise dropout described in (Bengio et al., 2015), where the dropped-out layer is sandwiched between fully-activated layers.

2.1 GATING NETWORK

Softmax Gating: A simple choice of non-sparse gating function (Jordan & Jacobs, 1994) is to multiply the input by a trainable weight matrix W_g and then apply the $Softmax$ function.

$$G_\sigma(x) = Softmax(x \cdot W_g) \tag{2}$$

Noisy Top-K Gating: We add two components to the Softmax gating network: sparsity and noise. Before taking the softmax function, we add tunable Gaussian noise, then keep only the top k values, setting the rest to $-\infty$ (which causes the corresponding gate values to equal 0). The sparsity serves to save computation, as described above. While this form of sparsity creates some theoretically scary discontinuities in the output of gating function, we have not yet observed this to be a problem in practice. The noise term helps with load balancing, as will be discussed in Appendix A. The amount of noise per component is controlled by a second trainable weight matrix W_{noise}.

$$G(x) = Softmax(KeepTopK(H(x), k)) \tag{3}$$

$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i) \tag{4}$$

$$KeepTopK(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases} \tag{5}$$

Training the Gating Network We train the gating network by simple back-propagation, along with the rest of the model. If we choose $k > 1$, the gate values for the top k experts have nonzero derivatives with respect to the weights of the gating network. This type of occasionally-sensitive behavior is described in (Bengio et al., 2013) with respect to noisy rectifiers. Gradients also back-propagate through the gating network to its inputs. Our method differs here from (Bengio et al., 2015) who use boolean gates and a REINFORCE-style approach to train the gating network.

3 ADDRESSING PERFORMANCE CHALLENGES

3.1 THE SHRINKING BATCH PROBLEM

On modern CPUs and GPUs, large batch sizes are necessary for computational efficiency, so as to amortize the overhead of parameter loads and updates. If the gating network chooses k out of n experts for each example, then for a batch of b examples, each expert receives a much smaller batch of approximately $\frac{kb}{n} \ll b$ examples. This causes a naive MoE implementation to become very inefficient as the number of experts increases. The solution to this shrinking batch problem is to make the original batch size as large as possible. However, batch size tends to be limited by the memory necessary to store activations between the forwards and backwards passes. We propose the following techniques for increasing the batch size:

Mixing Data Parallelism and Model Parallelism: In a conventional distributed training setting, multiple copies of the model on different devices asynchronously process distinct batches of data, and parameters are synchronized through a set of parameter servers. In our technique, these different batches run synchronously so that they can be combined for the MoE layer. We distribute the standard layers of the model and the gating network according to conventional data-parallel schemes, but keep only one shared copy of each expert. Each expert in the MoE layer receives a combined batch consisting of the relevant examples from all of the data-parallel input batches. The same set of devices function as data-parallel replicas (for the standard layers and the gating networks) and as model-parallel shards (each hosting a subset of the experts). If the model is distributed over d devices, and each device processes a batch of size b, each expert receives a batch of approximately $\frac{kbd}{n}$ examples. Thus, we achieve a factor of d improvement in expert batch size.

In the case of a hierarchical MoE (Section B), the primary gating network employs data parallelism, and the secondary MoEs employ model parallelism. Each secondary MoE resides on one device.

This technique allows us to increase the number of experts (and hence the number of parameters) by proportionally increasing the number of devices in the training cluster. The total batch size increases, keeping the batch size per expert constant. The memory and bandwidth requirements per device also remain constant, as do the step times, as does the amount of time necessary to process a number of training examples equal to the number of parameters in the model. It is our goal to train a trillion-parameter model on a trillion-word corpus. We have not scaled our systems this far as of the writing of this paper, but it should be possible by adding more hardware.

Taking Advantage of Convolutionality: In our language models, we apply the same MoE to each time step of the previous layer. If we wait for the previous layer to finish, we can apply the MoE to all the time steps together as one big batch. Doing so increases the size of the input batch to the MoE layer by a factor of the number of unrolled time steps.

Increasing Batch Size for a Recurrent MoE: We suspect that even more powerful models may involve applying a MoE recurrently. For example, the weight matrices of a LSTM or other RNN could be replaced by a MoE. Sadly, such models break the convolutional trick from the last paragraph, since the input to the MoE at one timestep depends on the output of the MoE at the previous timestep. Gruslys et al. (2016) describe a technique for drastically reducing the number of stored activations in an unrolled RNN, at the cost of recomputing forward activations. This would allow for a large increase in batch size.

3.2 Network Bandwidth

Another major performance concern in distributed computing is network bandwidth. Since the experts are stationary (see above) and the number of gating parameters is small, most of the communication involves sending the inputs and outputs of the experts across the network. To maintain computational efficiency, the ratio of an expert's computation to the size of its input and output must exceed the ratio of computational to network capacity of the computing device. For GPUs, this may be thousands to one. In our experiments, we use experts with one hidden layer containing thousands of RELU-activated units. Since the weight matrices in the expert have sizes $input\_size \times hidden\_size$ and $hidden\_size \times output\_size$, the ratio of computation to input and output is equal to the size of the hidden layer. Conveniently, we can increase computational efficiency simply by using a larger hidden layer, or more hidden layers.

4 Balancing Expert Utilization

We have observed that the gating network tends to converge to a state where it always produces large weights for the same few experts. This imbalance is self-reinforcing, as the favored experts are trained more rapidly and thus are selected even more by the gating network. Eigen et al. (2013) describe the same phenomenon, and use a hard constraint at the beginning of training to avoid this local minimum. Bengio et al. (2015) include a soft constraint on the batch-wise average of each gate.[1]

We take a soft constraint approach. We define the importance of an expert relative to a batch of training examples to be the batchwise sum of the gate values for that expert. We define an additional loss $L_{importance}$, which is added to the overall loss function for the model. This loss is equal to the square of the coefficient of variation of the set of importance values, multiplied by a hand-tuned scaling factor $w_{importance}$. This additional loss encourages all experts to have equal importance.

$$Importance(X) = \sum_{x \in X} G(x) \tag{6}$$

$$L_{importance}(X) = w_{importance} \cdot CV(Importance(X))^2 \tag{7}$$

[1] Bengio et al. (2015) also include two additional losses. One controls per-example sparsity, which we do not need since it is enforced by the fixed value of k. A third loss encourages diversity of gate values. In our experiments, we find that the gate values naturally diversify as the experts specialize (in a virtuous cycle), and we do not need to enforce diversity of gate values.

While this loss function can ensure equal importance, experts may still receive very different numbers of examples. For example, one expert may receive a few examples with large weights, and another may receive many examples with small weights. This can cause memory and performance problems on distributed hardware. To solve this problem, we introduce a second loss function, L_{load} , which ensures balanced loads. Appendix A contains the definition of this function, along with experimental results.

5 EXPERIMENTS

5.1 1 BILLION WORD LANGUAGE MODELING BENCHMARK

Dataset: This dataset, introduced by (Chelba et al., 2013) consists of shuffled unique sentences from news articles, totaling approximately 829 million words, with a vocabulary of 793,471 words.

Previous State-of-the-Art: The best previously published results (Jozefowicz et al., 2016) use models consisting of one or more stacked Long Short-Term Memory (LSTM) layers (Hochreiter & Schmidhuber, 1997; Gers et al., 2000). The number of parameters in the LSTM layers of these models vary from 2 million to 151 million. Quality increases greatly with parameter count, as do computational costs. Results for these models form the top line of Figure 2-right.

MoE Models: Our models consist of two stacked LSTM layers with a MoE layer between them (see Figure 1). We vary the sizes of the layers and the number of experts. For full details on model architecture, training regimen, additional baselines and results, see Appendix C.

Low Computation, Varied Capacity: To investigate the effects of adding capacity, we trained a series of MoE models all with roughly equal computational costs: about 8 million multiply-and-adds per training example per timestep in the forwards pass, excluding the softmax layer. We call this metric (ops/timestep). We trained models with flat MoEs containing 4, 32, and 256 experts, and models with hierarchical MoEs containing 256, 1024, and 4096 experts. Each expert had about 1 million parameters. For all the MoE layers, 4 experts were active per input.

The results of these models are shown in Figure 2-left. The model with 4 always-active experts performed (unsurprisingly) similarly to the computationally-matched baseline models, while the largest of the models (4096 experts) achieved an impressive 24% lower perplexity on the test set.

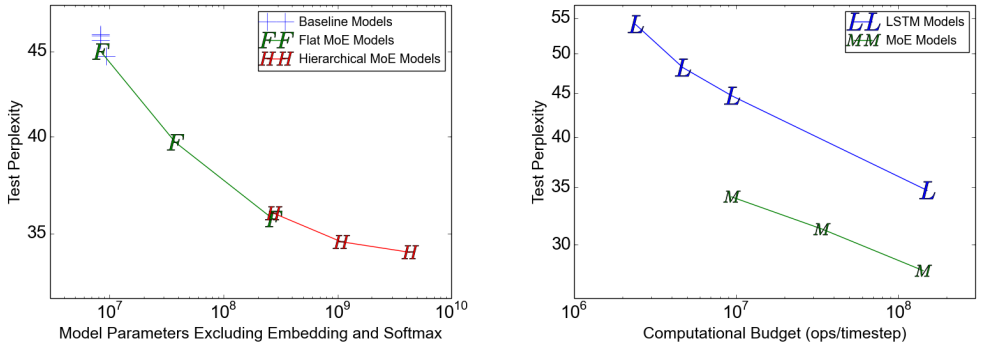

Figure 2: Model comparison on 1-Billion-Word Language-Modeling Benchmark. On the left, we plot test perplexity as a function of model capacity for models with similar computational budgets of approximately 8-million-ops-per-timestep. On the right, we plot test perplexity as a function of computational budget. The top line represents the LSTM models from (Jozefowicz et al., 2016). The bottom line represents 4-billion parameter MoE models with different computational budgets.

Varied Computation, High Capacity: In addition to the largest model from the previous section, we trained two more MoE models with similarly high capacity (4 billion parameters), but higher computation budgets. These models had larger LSTMs, and fewer but larger and experts. Details

Table 1: Summary of high-capacity MoE-augmented models with varying computational budgets, vs. best previously published results (Jozefowicz et al., 2016). Details in Appendix C.

	Test Perplexity 10 epochs	Test Perplexity 100 epochs	#Parameters excluding embedding and softmax layers	ops/timestep	Training Time 10 epochs	TFLOPS /GPU
Best Published Results	34.7	30.6	151 million	151 million	59 hours, 32 k40s	1.09
Low-Budget MoE Model	34.1		4303 million	8.9 million	15 hours, 16 k40s	0.74
Medium-Budget MoE Model	31.3		4313 million	33.8 million	17 hours, 32 k40s	1.22
High-Budget MoE Model	**28.0**		4371 million	142.7 million	47 hours, 32 k40s	**1.56**

can be found in Appendix C.2. Results of these three models form the bottom line of Figure 2-right. Table 1 compares the results of these models to the best previously-published result on this dataset . Even the fastest of these models beats the best published result (when controlling for the number of training epochs), despite requiring only 6% of the computation.

Computational Efficiency: We trained our models using TensorFlow (Abadi et al., 2016) on clusters containing 16-32 Tesla K40 GPUs. For each of our models, we determine computational efficiency in TFLOPS/GPU by dividing the number of floating point operations required to process one training batch by the observed step time and the number of GPUs in the cluster. The operation counts used here are higher than the ones we report in our ops/timestep numbers in that we include the backwards pass, we include the importance-sampling-based training of the softmax layer, and we count a multiply-and-add as two separate operations. For all of our MoE models, the floating point operations involved in the experts represent between 37% and 46% of the total.

For our baseline models wtih no MoE, observed computational efficiency ranged from 1.07-1.29 TFLOPS/GPU. For our low-computation MoE models, computation efficiency ranged from 0.74-0.90 TFLOPS/GPU, except for the 4-expert model which did not make full use of the available parallelism. Our highest-computation MoE model was more efficient at 1.56 TFLOPS/GPU, likely due to the larger matrices. These numbers represent a significant fraction of the theoretical maximum of 4.29 TFLOPS/GPU claimed by NVIDIA. Detailed results are in Appendix C, Table 7.

5.2 100 Billion Word Google News Corpus

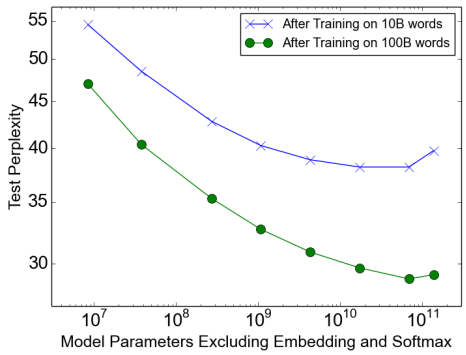

Figure 3: Language modeling on a 100 billion word corpus. Models have similar computational budgets (8 million ops/timestep).

On the 1-billion-word corpus, adding additional capacity seems to produce diminishing returns as the number of parameters in the MoE layer exceeds 1 billion, as can be seen in Figure 2-left. We hypothesized that for a larger training set, even higher capacities would produce significant quality improvements.

We constructed a similar training set consisting of shuffled unique sentences from Google's internal news corpus, totalling roughly 100 billion words. Similarly to the previous section, we tested a series of models with similar computational costs of about 8 million ops/timestep. In addition to a baseline LSTM model, we trained models augmented with MoE layers containing 32, 256, 1024,

4096, 16384, 65536, and 131072 experts. This corresponds to up to 137 billion parameters in the MoE layer. Details on architecture, training, and results are given in Appendix D.

Results: Figure 3 shows test perplexity as a function of capacity after training on 10 billion words (top line) and 100 billion words (bottom line). When training over the full 100 billion words, test perplexity improves significantly up to 65536 experts (68 billion parameters), dropping 39% lower than the computationally matched baseline, but degrades at 131072 experts, possibly a result of too much sparsity. The widening gap between the two lines demonstrates (unsurprisingly) that increased model capacity helps more on larger training sets.

Even at 65536 experts (99.994% layer sparsity), computational efficiency for the model stays at a respectable 0.72 TFLOPS/GPU.

5.3 Machine Translation (Single Language Pair)

Model Architecture: Our model was a modified version of the GNMT model described in (Wu et al., 2016). To reduce computation, we decreased the number of LSTM layers in the encoder and decoder from 9 and 8 to 3 and 2 respectively. We inserted MoE layers in both the encoder (between layers 2 and 3) and the decoder (between layers 1 and 2). Each MoE layer contained up to 2048 experts each with about two million parameters, adding a total of about 8 billion parameters to the models. Further details on model architecture, testing procedure and results can be found in Appendix E.

Datasets: We benchmarked our method on the WMT'14 En→Fr and En→De corpora, whose training sets have 36M sentence pairs and 5M sentence pairs, respectively. The experimental protocols were also similar to those in (Wu et al., 2016): newstest2014 was used as the test set to compare against previous work (Luong et al., 2015a; Zhou et al., 2016; Wu et al., 2016), while the combination of newstest2012 and newstest2013 was used as the development set. We also tested the same model on a Google's Production English to French data.

Table 2: Results on WMT'14 En→ Fr newstest2014 (bold values represent best results).

Model	Test Perplexity	Test BLEU	ops/timenstep	Total #Parameters	Training Time
MoE with 2048 Experts	2.69	40.35	85M	8.7B	3 days/64 k40s
MoE with 2048 Experts (longer training)	**2.63**	**40.56**	85M	8.7B	6 days/64 k40s
GNMT (Wu et al., 2016)	2.79	39.22	214M	278M	6 days/96 k80s
GNMT+RL (Wu et al., 2016)	2.96	39.92	214M	278M	6 days/96 k80s
PBMT (Durrani et al., 2014)		37.0			
LSTM (6-layer) (Luong et al., 2015b)		31.5			
LSTM (6-layer+PosUnk) (Luong et al., 2015b)		33.1			
DeepAtt (Zhou et al., 2016)		37.7			
DeepAtt+PosUnk (Zhou et al., 2016)		39.2			

Table 3: Results on WMT'14 En → De newstest2014 (bold values represent best results).

Model	Test Perplexity	Test BLEU	ops/timestep	Total #Parameters	Training Time
MoE with 2048 Experts	**4.64**	**26.03**	85M	8.7B	1 day/64 k40s
GNMT (Wu et al., 2016)	5.25	24.91	214M	278M	1 day/96 k80s
GNMT +RL (Wu et al., 2016)	8.08	24.66	214M	278M	1 day/96 k80s
PBMT (Durrani et al., 2014)		20.7			
DeepAtt (Zhou et al., 2016)		20.6			

Table 4: Results on the Google Production En→ Fr dataset (bold values represent best results).

Model	Eval Perplexity	Eval BLEU	Test Perplexity	Test BLEU	ops/timestep	Total #Parameters	Training Time
MoE with 2048 Experts	**2.60**	**37.27**	**2.69**	**36.57**	85M	8.7B	1 day/64 k40s
GNMT (Wu et al., 2016)	2.78	35.80	2.87	35.56	214M	278M	6 days/96 k80s

Results: Tables 2, 3, and 4 show the results of our largest models, compared with published results. Our approach achieved BLEU scores of 40.56 and 26.03 on the WMT'14 En→Fr and En→De benchmarks. As our models did not use RL refinement, these results constitute significant gains of 1.34 and 1.12 BLEU score on top of the strong baselines in (Wu et al., 2016). The perplexity scores are also better.[2] On the Google Production dataset, our model achieved 1.01 higher test BLEU score even after training for only one sixth of the time.

5.4 Multilingual Machine Translation

Dataset: (Johnson et al., 2016) train a single GNMT (Wu et al., 2016) model on a very large combined dataset of twelve language pairs. Results are somewhat worse than those for 12 separately trained single-pair GNMT models. This is not surprising, given that the twelve models have 12 times the capacity and twelve times the aggregate training of the one model. We repeat this experiment with a single MoE-augmented model. See Appendix E for details on model architecture. We train our model on the same dataset as (Johnson et al., 2016) and process the same number of training examples (about 3 billion sentence pairs). Our training time was shorter due to the lower computational budget of our model.

Results: Results for the single-pair GNMT models, the multilingual GNMT model and the multilingual MoE model are given in Table 5. The MoE model achieves 19% lower perplexity on the dev set than the multilingual GNMT model. On BLEU score, the MoE model significantly beats the multilingual GNMT model on 11 of the 12 language pairs (by as much as 5.84 points), and even beats the monolingual GNMT models on 8 of 12 language pairs. The poor performance on English → Korean seems to be a result of severe overtraining, as for the rarer language pairs a small number of real examples were highly oversampled in the training corpus.

Table 5: Multilingual Machine Translation (bold values represent best results).

	GNMT-Mono	GNMT-Multi	MoE-Multi	MoE-Multi vs. GNMT-Multi
Parameters	278M / model	278M	8.7B	
ops/timestep	212M	212M	102M	
training time, hardware	various	21 days, 96 k20s	**12 days, 64 k40s**	
Perplexity (dev)		4.14	**3.35**	-19%
French → English Test BLEU	36.47	34.40	**37.46**	+3.06
German → English Test BLEU	31.77	31.17	**34.80**	+3.63
Japanese → English Test BLEU	23.41	21.62	**25.91**	+4.29
Korean → English Test BLEU	25.42	22.87	**28.71**	+5.84
Portuguese → English Test BLEU	44.40	42.53	**46.13**	+3.60
Spanish → English Test BLEU	38.00	36.04	**39.39**	+3.35
English → French Test BLEU	35.37	34.00	**36.59**	+2.59
English → German Test BLEU	**26.43**	23.15	24.53	+1.38
English → Japanese Test BLEU	**23.66**	21.10	22.78	+1.68
English → Korean Test BLEU	**19.75**	18.41	16.62	-1.79
English → Portuguese Test BLEU	**38.40**	37.35	37.90	+0.55
English → Spanish Test BLEU	34.50	34.25	**36.21**	+1.96

6 Conclusion

This work is the first to demonstrate major wins from conditional computation in deep networks. We carefully identified the design considerations and challenges of conditional computing and addressed them with a combination of algorithmic and engineering solutions. While we focused on text, conditional computation may help in other domains as well, provided sufficiently large training sets. We look forward to seeing many novel implementations and applications of conditional computation in the years to come.

Acknowledgments

We would like to thank all of the members of the Google Brain and Google Translate teams who helped us with this project, in particular Zhifeng Chen, Yonghui Wu, and Melvin Johnson. Thanks also to our anonymous ICLR reviewers for the helpful suggestions on making this paper better.

[2]Reported perplexities relative to the tokenization used by both our models and GNMT.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016. URL `http://arxiv.org/abs/1603.04467`.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. *CoRR*, abs/1611.06194, 2016. URL `http://arxiv.org/abs/1611.06194`.

A. Almahairi, N. Ballas, T. Cooijmans, Y. Zheng, H. Larochelle, and A. Courville. Dynamic Capacity Networks. *ArXiv e-prints*, November 2015.

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

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

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

K. Cho and Y. Bengio. Exponentially Increasing the Capacity-to-Computation Ratio for Conditional Computation in Deep Learning. *ArXiv e-prints*, June 2014.

Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computing*, 2002.

Andrew Davis and Itamar Arel. Low-rank approximations for conditional feedforward computation in deep neural networks. *arXiv preprint arXiv:1312.4461*, 2013.

Marc Peter Deisenroth and Jun Wei Ng. Distributed Gaussian processes. In *ICML*, 2015.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization, 2010.

Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. Edinburgh's phrase-based machine translation systems for wmt-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2014.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

Ekaterina Garmash and Christof Monz. Ensemble learning for multi-source neural machine translation. In *staff.science.uva.nl/c.monz*, 2016.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 2000.

Audrunas Gruslys, Rémi Munos, Ivo Danihelka, Marc Lanctot, and Alex Graves. Memory-efficient backpropagation through time. *CoRR*, abs/1606.03401, 2016. URL `http://arxiv.org/abs/1606.03401`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computing*, 1991.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558, 2016. URL `http://arxiv.org/abs/1611.04558`.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computing*, 1994.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Reinhard Kneser and Hermann. Ney. Improved backingoff for m-gram language modeling., 1995.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeffrey Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

Patrick Gallinari Ludovic Denoyer. Deep sequential neural network. *arXiv preprint arXiv:1410.0510*, 2014.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *EMNLP*, 2015a.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *ACL*, 2015b.

Carl Edward Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. *NIPS*, 2002.

Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pp. 338–342, 2014.

Mike Schuster and Kaisuke Nakajima. Japanese and Korean voice search. *ICASSP*, 2012.

Babak Shahbaba and Radford Neal. Nonlinear models using dirichlet process mixtures. *JMLR*, 2009.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Lucas Theis and Matthias Bethge. Generative image modeling using spatial LSTMs. In *NIPS*, 2015.

Volker Tresp. Mixtures of Gaussian Processes. In *NIPS*, 2001.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Bangpeng Yao, Dirk Walther, Diane Beck, and Li Fei-fei. Hierarchical mixture of classification experts uncovers interactions between brain regions. In *NIPS*. 2009.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199*, 2016.

APPENDICES

A LOAD-BALANCING LOSS

As discussed in section 4, for load-balancing purposes, we want to define an additional loss function to encourage experts to receive roughly equal numbers of training examples. Unfortunately, the number of examples received by an expert is a discrete quantity, so it can not be used in back-propagation. Instead, we define a smooth estimator $Load(X)$ of the number of examples assigned to each expert for a batch X of inputs. The smoothness allows us to back-propagate gradients through the estimator. This is the purpose of the noise term in the gating function. We define $P(x,i)$ as the probability that $G(x)_i$ is nonzero, given a new random choice of noise on element i, but keeping the already-sampled choices of noise on the other elements. To compute $P(x,i)$, we note that the $G(x)_i$ is nonzero if and only if $H(x)_i$ is greater than the k^{th}-greatest element of $H(x)$ excluding itself. The probability works out to be:

$$P(x,i) = Pr\Big((x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i) \\ > kth\_excluding(H(x),k,i)\Big) \tag{8}$$

Where $kth\_excluding(v,k,i)$ means the kth highest component of v, excluding component i. Simplifying, we get:

$$P(x,i) = \Phi\Big(\frac{(x \cdot W_g)_i - kth\_excluding(H(x),k,i)}{Softplus((x \cdot W_{noise})_i)}\Big) \tag{9}$$

Where Φ is the CDF of the standard normal distribution.

$$Load(X)_i = \sum_{x \in X} P(x,i) \tag{10}$$

We can now define the load loss to be the square of the coefficient of variation of the load vector, multiplied by a hand-tuned scaling factor w_{load}.

$$L_{load}(X) = w_{load} \cdot CV(Load(X))^2 \tag{11}$$

Initial Load Imbalance: To avoid out-of-memory errors, we need to initialize the network in a state of approximately equal expert load (since the soft constraints need some time to work). To accomplish this, we initialize the matrices W_g and W_{noise} to all zeros, which yields no signal and some noise.

Experiments: We trained a set of models with identical architecture (the MoE-256 model described in Appendix C), using different values of $w_{importance}$ and w_{load}. We trained each model for 10 epochs, then measured perplexity on the test set. We also measured the coefficients of variation in $Importance$ and $Load$, as well as ratio of the load on the most overloaded expert to the average load. This last value is significant for load balancing purposes on distributed hardware. All of these metrics were averaged over several training batches.

Table 6: Experiments with different combinations of losses.

$w_{importance}$	w_{load}	Test Perplexity	$CV(Importance(X))$	$CV(Load(X))$	$\frac{max(Load(X))}{mean(Load(X))}$
0.0	0.0	39.8	3.04	3.01	17.80
0.2	0.0	**35.6**	0.06	0.17	1.47
0.0	0.2	35.7	0.22	0.04	1.15
0.1	0.1	**35.6**	0.06	0.05	1.14
0.01	0.01	35.7	0.48	0.11	1.37
1.0	1.0	35.7	0.03	0.02	**1.07**

Results: Results are reported in Table 6. All the combinations containing at least one the two losses led to very similar model quality, where having no loss was much worse. Models with higher values of w_{load} had lower loads on the most overloaded expert.

B HIERACHICAL MIXTURE OF EXPERTS

If the number of experts is very large, we can reduce the branching factor by using a two-level hierarchical MoE. In a hierarchical MoE, a primary gating network chooses a sparse weighted combination of "experts", each of which is itself a secondary mixture-of-experts with its own gating network.[3] If the hierarchical MoE consists of a groups of b experts each, we denote the primary gating network by $G_{primary}$, the secondary gating networks by $(G_1, G_2..G_a)$, and the expert networks by $(E_{0,0}, E_{0,1}..E_{a,b})$. The output of the MoE is given by:

$$y_H = \sum_{i=1}^{a}\sum_{j=1}^{b} G_{primary}(x)_i \cdot G_i(x)_j \cdot E_{i,j}(x) \tag{12}$$

Our metrics of expert utilization change to the following:

$$Importance_H(X)_{i,j} = \sum_{x \in X} G_{primary}(x)_i \cdot G_i(x)_j \tag{13}$$

$$Load_H(X)_{i,j} = \frac{Load_{primary}(X)_i \cdot Load_i(X^{(i)})_j}{|X^{(i)}|} \tag{14}$$

$Load_{primary}$ and $Load_i$ deonte the $Load$ functions for the primary gating network and i^{th} secondary gating network respectively. $X^{(i)}$ denotes the subset of X for which $G_{primary}(x)_i > 0$.

It would seem simpler to let $Load_H(X)_{i,j} = Load_i(X_i)_j$, but this would not have a gradient with respect to the primary gating network, so we use the formulation above.

C 1 BILLION WORD LANGUAGE MODELING BENCHMARK - EXPERIMENTAL DETAILS

C.1 8-MILLION-OPERATIONS-PER-TIMESTEP MODELS

Model Architecture: Our model consists of five layers: a word embedding layer, a recurrent Long Short-Term Memory (LSTM) layer (Hochreiter & Schmidhuber, 1997; Gers et al., 2000), a MoE layer, a second LSTM layer, and a softmax layer. The dimensionality of the embedding layer, the number of units in each LSTM layer, and the input and output dimensionality of the MoE layer are all equal to 512. For every layer other than the softmax, we apply drouput (Zaremba et al., 2014) to the layer output, dropping each activation with probability $DropProb$, otherwise dividing by $(1 - DropProb)$. After dropout, the output of the previous layer is added to the layer output. This residual connection encourages gradient flow (He et al., 2015).

MoE Layer Architecture: Each expert in the MoE layer is a feed forward network with one ReLU-activated hidden layer of size 1024 and an output layer of size 512. Thus, each expert contains $[512 * 1024] + [1024 * 512] = 1M$ parameters. The output of the MoE layer is passed through a sigmoid function before dropout. We varied the number of experts between models, using ordinary MoE layers with 4, 32 and 256 experts and hierarchical MoE layers with 256, 1024 and 4096 experts. We call the resulting models MoE-4, MoE-32, MoE-256, MoE-256-h, MoE-1024-h and MoE-4096-h. For the hierarchical MoE layers, the first level branching factor was 16, corresponding to the number of GPUs in our cluster. We use Noisy-Top-K Gating (see Section 2.1) with $k = 4$ for the ordinary MoE layers and $k = 2$ at each level of the hierarchical MoE layers. Thus, each example is processed by exactly 4 experts for a total of 4M ops/timestep. The two LSTM layers contribute 2M ops/timestep each for the desired total of 8M.

[3] We have not found the need for deeper hierarchies.

Computationally-Matched Baselines: The MoE-4 model does not employ sparsity, since all 4 experts are always used. In addition, we trained four more computationally-matched baseline models with no sparsity:

- MoE-1-Wide: The MoE layer consists of a single "expert" containing one ReLU-activated hidden layer of size 4096.
- MoE-1-Deep: The MoE layer consists of a single "expert" containing four ReLU-activated hidden layers, each with size 1024.
- 4xLSTM-512: We replace the MoE layer with two additional 512-unit LSTM layers.
- LSTM-2048-512: The model contains one 2048-unit LSTM layer (and no MoE). The output of the LSTM is projected down to 512 dimensions (Sak et al., 2014). The next timestep of the LSTM receives the projected output. This is identical to one of the models published in (Jozefowicz et al., 2016). We re-ran it to account for differences in training regimen, and obtained results very similar to the published ones.

Training: The models were trained on a cluster of 16 K40 GPUs using the synchronous method described in Section 3. Each batch consisted of a set of sentences totaling roughly 300,000 words. In the interest of time, we limited training to 10 epochs, (27,000 steps). Training took 12-16 hours for all models, except for MoE-4, which took 18 hours (since all the expert computation was performed on only 4 of 16 GPUs). We used the Adam optimizer (Kingma & Ba, 2015). The base learning rate was increased linearly for the first 1000 training steps, and decreased after that so as to be proportional to the inverse square root of the step number. The Softmax output layer was trained efficiently using importance sampling similarly to the models in (Jozefowicz et al., 2016). For each model, we performed a hyper-parmeter search to find the best dropout probability, in increments of 0.1.

To ensure balanced expert utilization we set $w_{importance} = 0.1$ and $w_{load} = 0.1$, as described in Section 4 and Appendix A.

Results: We evaluate our model using perplexity on the holdout dataset, used by (Chelba et al., 2013; Jozefowicz et al., 2016). We follow the standard procedure and sum over all the words including the end of sentence symbol. Results are reported in Table 7. For each model, we report the test perplexity, the computational budget, the parameter counts, the value of $DropProb$, and the computational efficiency.

Table 7: Model comparison on 1 Billion Word Language Modeling Benchmark. Models marked with * are from (Jozefowicz et al., 2016).

Model	Test Perplexity 10 epochs	Test Perplexity (final)	ops/timestep (millions)	#Params excluding embed. & softmax (millions)	Total #Params (billions)	$Drop-Prob$	TFLOPS per GPU (observed)
Kneser-Ney 5-gram*		67.6	0.00001		1.8		
LSTM-512-512*		54.1	2.4	2.4	0.8	0.1	
LSTM-1024-512*		48.2	4.7	4.7	0.8	0.1	
LSTM-2048-512*	45.0	43.7	9.4	9.4	0.8	0.1	0.61
LSTM-2048-512	44.7		9.4	9.4	0.8	0.1	1.21
4xLSTM-512	46.0		8.4	8.4	0.8	0.1	1.07
MoE-1-Wide	46.1		8.4	8.4	0.8	0.1	1.29
MoE-1-Deep	45.7		8.4	8.4	0.8	0.1	1.29
MoE-4	45.0		8.4	8.4	0.8	0.1	0.52
MoE-32	39.7		8.4	37.8	0.9	0.1	0.87
MoE-256	35.7		8.6	272.9	1.1	0.1	0.81
MoE-256-h	36.0		8.4	272.9	1.1	0.1	0.89
MoE-1024-h	34.6		8.5	1079.0	1.9	0.2	0.90
MoE-4096-h	34.1		8.9	4303.4	5.1	0.2	0.74
2xLSTM-8192-1024*	34.7	30.6	151.0	151.0	1.8	0.25	1.09
MoE-34M	31.3		33.8	4313.9	6.0	0.3	1.22
MoE-143M	**28.0**		142.7	4371.1	6.0	0.4	**1.56**

C.2 More Expensive Models

We ran two additional models (MoE-34M and MoE-143M) to investigate the effects of adding more computation in the presence of a large MoE layer. These models have computation budgets of 34M and 143M ops/timestep. Similar to the models above, these models use a MoE layer between two LSTM layers. The dimensionality of the embedding layer, and the input and output dimensionality of the MoE layer are set to 1024 instead of 512. For MoE-34M, the LSTM layers have 1024 units. For MoE-143M, the LSTM layers have 4096 units and an output projection of size 1024 (Sak et al., 2014). MoE-34M uses a hierarchical MoE layer with 1024 experts, each with a hidden layer of size 2048. MoE-143M uses a hierarchical MoE layer with 256 experts, each with a hidden layer of size 8192. Both models have 4B parameters in the MoE layers. We searched for the best $DropProb$ for each model, and trained each model for 10 epochs.

The two models achieved test perplexity of 31.3 and 28.0 respectively, showing that even in the presence of a large MoE, more computation is still useful. Results are reported at the bottom of Table 7. The larger of the two models has a similar computational budget to the best published model from the literature, and training times are similar. Comparing after 10 epochs, our model has a lower test perplexity by 18%.

D 100 Billion Word Google News Corpus - Experimental Details

Model Architecture: The models are similar in structure to the 8-million-operations-per-timestep models described in the previous section. We vary the number of experts between models, using an ordinary MoE layer with 32 experts and hierarchical MoE layers with 256, 1024, 4096, 16384, 65536 and 131072 experts. For the hierarchical MoE layers, the first level branching factors are 32, 32, 64, 128, 256 and 256, respectively.

Training: Models are trained on a cluster of 32 Tesla K40 GPUs, except for the last two models, which are trained on clusters of 64 and 128 GPUs so as to have enough memory for all the parameters. For all models, training batch sizes are approximately 2.5 million words. Models are trained once-through over about 100 billion words.

We implement several memory optimizations in order to fit up to 1 billion parameters per GPU. First, we do not store the activations of the hidden layers of the experts, but instead recompute them on the backwards pass. Secondly, we modify the optimizer on the expert parameters to require less auxiliary storage:

The Adam optimizer (Kingma & Ba, 2015) keeps first and second moment estimates of the per-parameter gradients. This triples the required memory. To avoid keeping a first-moment estimator, we set $\beta_1 = 0$. To reduce the size of the second moment estimator, we replace it with a factored approximation. For a matrix of parameters, instead of maintaining a full matrix of second-moment estimators, we maintain vectors of row-wise and column-wise averages of that matrix. At each step, the matrix of estimators is taken to be the outer product of those two vectors divided by the mean of either one. This technique could similarly be applied to Adagrad (Duchi et al., 2010).

Table 8: Model comparison on 100 Billion Word Google News Dataset

Model	Test Perplexity .1 epochs	Test Perplexity 1 epoch	ops/timestep (millions)	#Params excluding embed. & softmax (millions)	Total #Params (billions)	TFLOPS per GPU (observed)
Kneser-Ney 5-gram	67.1	45.3	0.00001		76.0	
4xLSTM-512	54.5	47.0	8.4	8.4	0.1	**1.23**
MoE-32	48.5	40.4	8.4	37.8	0.1	0.83
MoE-256-h	42.8	35.3	8.4	272.9	0.4	1.11
MoE-1024-h	40.3	32.7	8.5	1079.0	1.2	1.14
MoE-4096-h	38.9	30.9	8.6	4303.4	4.4	1.07
MoE-16384-h	**38.2**	29.7	8.8	17201.0	17.3	0.96
MoE-65536-h	**38.2**	**28.9**	9.2	68791.0	68.9	0.72
MoE-131072-h	39.8	29.2	9.7	137577.6	137.7	0.30

Results: We evaluate our model using perplexity on a holdout dataset. Results are reported in Table 8. Perplexity after 100 billion training words is 39% lower for the 68-billion-parameter MoE

model than for the baseline model. It is notable that the measured computational efficiency of the largest model (0.30 TFLOPS/GPU) is very low compared to the other models. This is likely a result of the fact that, for purposes of comparison to the other models, we did not increase the training batch size proportionally to the number of GPUs. For comparison, we include results for a computationally matched baseline model consisting of 4 LSTMs, and for an unpruned 5-gram model with Kneser-Ney smoothing (Kneser & Ney, 1995).[4]

E MACHINE TRANSLATION - EXPERIMENTAL DETAILS

Model Architecture for Single Language Pair MoE Models: Our model is a modified version of the GNMT model described in (Wu et al., 2016). To reduce computation, we decrease the number of LSTM layers in the encoder and decoder from 9 and 8 to 3 and 2 respectively. We insert MoE layers in both the encoder (between layers 2 and 3) and the decoder (between layers 1 and 2). We use an attention mechanism between the encoder and decoder, with the first decoder LSTM receiving output from and providing input for the attention [5]. All of the layers in our model have input and output dimensionality of 512. Our LSTM layers have 2048 hidden units, with a 512-dimensional output projection. We add residual connections around all LSTM and MoE layers to encourage gradient flow (He et al., 2015). Similar to GNMT, to effectively deal with rare words, we used sub-word units (also known as "wordpieces") (Schuster & Nakajima, 2012) for inputs and outputs in our system.

We use a shared source and target vocabulary of 32K wordpieces. We also used the same beam search technique as proposed in (Wu et al., 2016).

We train models with different numbers of experts in the MoE layers. In addition to a baseline model with no MoE layers, we train models with flat MoE layers containing 32 experts, and models with hierarchical MoE layers containing 512 and 2048 experts. The flat MoE layers use $k = 4$ and the hierarchical MoE models use $k = 2$ at each level of the gating network. Thus, each input is processed by exactly 4 experts in each MoE layer. Each expert in the MoE layer is a feed forward network with one hidden layer of size 2048 and ReLU activation. Thus, each expert contains $[512 * 2048] + [2048 * 512] = 2M$ parameters. The output of the MoE layer is passed through a sigmoid function. We use the strictly-balanced gating function described in Appendix F.

Model Architecture for Multilingual MoE Model: We used the same model architecture as for the single-language-pair models, with the following exceptions: We used noisy-top-k gating as described in Section 2.1, not the scheme from Appendix F. The MoE layers in the encoder and decoder are non-hierarchical MoEs with $n = 512$ experts, and $k = 2$. Each expert has a larger hidden layer of size 8192. This doubles the amount of computation in the MoE layers, raising the computational budget of the entire model from 85M to 102M ops/timestep.

Training: We trained our networks using the Adam optimizer (Kingma & Ba, 2015). The base learning rate was increased linearly for the first 2000 training steps, held constant for an additional 8000 steps, and decreased after that so as to be proportional to the inverse square root of the step number. For the single-language-pair models, similarly to (Wu et al., 2016), we applied dropout (Zaremba et al., 2014) to the output of all embedding, LSTM and MoE layers, using $DropProb = 0.4$. Training was done synchronously on a cluster of up to 64 GPUs as described in section 3. Each training batch consisted of a set of sentence pairs containing roughly 16000 words per GPU.

To ensure balanced expert utilization we set $w_{importance} = 0.01$ and $w_{load} = 0.01$, as described in Section 4 and Appendix A.

Metrics: We evaluated our models using the perplexity and the standard BLEU score metric. We reported tokenized BLEU score as computed by the multi-bleu.pl script, downloaded from the public implementation of Moses (on Github), which was also used in (Luong et al., 2015a).

[4] While the original size of the corpus was 130 billion words, the neural models were trained for a maximum of 100 billion words. The reported Kneser-Ney 5-gram models were trained over 13 billion and 130 billion words respectively, giving them a slight advantage over the other reported results.

[5] For performance reasons, we use a slightly different attention function from the one described in (Wu et al., 2016) - See Appendix G

Results: Tables 2, 3 and 4 in Section 5.3 show comparisons of our results to other published methods. Figure 4 shows test perplexity as a function of number of words in the (training data's) source sentences processed for models with different numbers of experts. As can be seen from the Figure, as we increased the number of experts to approach 2048, the test perplexity of our model continued to improve.

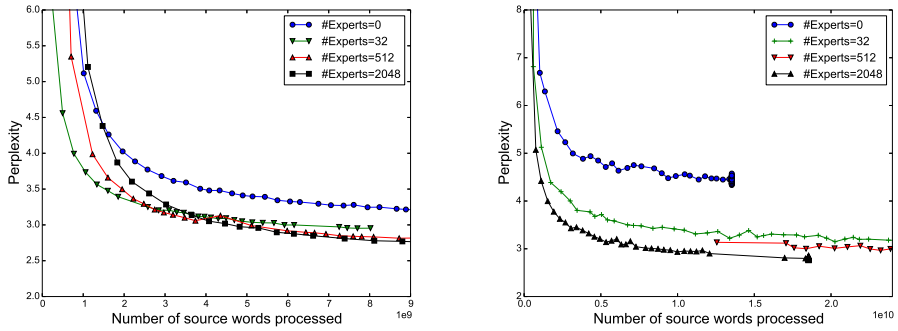

Figure 4: Perplexity on WMT'14 En→ Fr (left) and Google Production En→ Fr (right) datasets as a function of number of words processed. The large differences between models at the beginning of training are due to different batch sizes. All models incur the same computational budget (85M ops/timestep) except the one with no experts.

We found that the experts indeed become highly specialized by syntax and/or semantics, as can be seen in Table 9. For example, one expert is used when the indefinite article "a" introduces the direct object in a verb phrase indicating importance or leadership.

Table 9: Contexts corresponding to a few of the 2048 experts in the MoE layer in the encoder portion of the WMT'14 En→ Fr translation model. For each expert i, we sort the inputs in a training batch in decreasing order of $G(x)_i$, and show the words surrounding the corresponding positions in the input sentences.

Expert 381	Expert 752	Expert 2004
... with **researchers** , ...	... plays **a** core ...	... with **rapidly** growing ...
... to **innovation** .	... plays **a** critical ...	... under **static** conditions ...
... tics **researchers** .	... provides **a** legislative ...	... to **swift** ly ...
... the **generation** of ...	... play **a** leading ...	... to **dras** tically ...
... technology **innovations** is ...	... assume **a** leadership ...	... the **rapid** and ...
... technological **innovations** , ...	... plays **a** central ...	... the **fast** est ...
... support **innovation** throughout ...	... taken **a** leading ...	... the **Quick** Method ...
... role **innovation** will ...	... established **a** reconciliation ...	... rec **urrent**) ...
... research **scienti** st ...	... played **a** vital ...	... provides **quick** access ...
... promoting **innovation** where ...	... have **a** central ...	... of **volatile** organic ...
...	...	...

F STRICTLY BALANCED GATING

Due to some peculiarities in our infrastructure which have since been fixed, at the time we ran some of the machine translation experiments, our models ran faster if every expert received exactly the same batch size. To accommodate this, we used a different gating function which we describe below.

Recall that we define the softmax gating function to be:

$$G_\sigma(x) = Softmax(x \cdot W_g) \tag{15}$$

Sparse Gating (alternate formulation): To obtain a sparse gating vector, we multiply $G_\sigma(x)$ component-wise with a sparse mask $M(G_\sigma(x))$ and normalize the output. The mask itself is a function of $G_\sigma(x)$ and specifies which experts are assigned to each input example:

$$G(x)_i = \frac{G_\sigma(x)_i M(G_\sigma(x))_i}{\sum_{j=1}^{n} G_\sigma(x)_j M(G_\sigma(x))_j} \tag{16}$$

Top-K Mask: To implement top-k gating in this formulation, we would let $M(v) = TopK(v, k)$, where:

$$TopK(v, k)_i = \begin{cases} 1 & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Batchwise Mask: To force each expert to receive the exact same number of examples, we introduce an alternative mask function, $M_{batchwise}(X, m)$, which operates over batches of input vectors. Instead of keeping the top k values per example, we keep the top m values per expert across the training batch, where $m = \frac{k|X|}{n}$, so that each example is sent to an average of k experts.

$$M_{batchwise}(X, m)_{j,i} = \begin{cases} 1 & \text{if } X_{j,i} \text{ is in the top } m \text{ values for to expert } i \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

As our experiments suggest and also observed in (Ioffe & Szegedy, 2015), using a batchwise function during training (such as $M_{batchwise}$) requires modifications to the inference when we may not have a large batch of examples. Our solution to this is to train a vector T of per-expert threshold values to approximate the effects of the batchwise mask. We use the following mask at inference time:

$$M_{threshold}(x, T)_i = \begin{cases} 1 & \text{if } x_i > T_i \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

To learn the threshold values, we apply an additional loss at training time which is minimized when the batchwise mask and the threshold mask are identical.

$$L_{batchwise}(X, T, m) = \sum_{j=1}^{|X|} \sum_{i=1}^{n} (M_{threshold}(x, T)_i - M_{batchwise}(X, m)_{j,i})(X_{j,i} - T_i) \tag{20}$$

G Attention Function

The attention mechanism described in GNMT (Wu et al., 2016) involves a learned "Attention Function" $A(x_i, y_j)$ which takes a "source vector" x_i and a "target vector" y_j, and must be computed for every source time step i and target time step j. In GNMT, the attention function is implemented as a feed forward neural network with a hidden layer of size n. It can be expressed as:

$$A_{GNMT}(x_i, y_j) = \sum_{d=1}^{n} V_d tanh((x_i U)_d + (y_j W)_d) \tag{21}$$

Where U and W are trainable weight matrices and V is a trainable weight vector.

For performance reasons, in our models, we used a slightly different attention function:

$$A(x_i, y_j) = \sum_{d=1}^{n} V_d tanh((x_i U)_d) tanh((y_j W)_d) \tag{22}$$

With our attention function, we can simultaneously compute the attention function on multiple source time steps and multiple target time steps using optimized matrix multiplications. We found little difference in quality between the two functions.

Language Models are Few-Shot Learners

Tom B. Brown* **Benjamin Mann*** **Nick Ryder*** **Melanie Subbiah***

Jared Kaplan† **Prafulla Dhariwal** **Arvind Neelakantan** **Pranav Shyam** **Girish Sastry**

Amanda Askell **Sandhini Agarwal** **Ariel Herbert-Voss** **Gretchen Krueger** **Tom Henighan**

Rewon Child **Aditya Ramesh** **Daniel M. Ziegler** **Jeffrey Wu** **Clemens Winter**

Christopher Hesse **Mark Chen** **Eric Sigler** **Mateusz Litwin** **Scott Gray**

Benjamin Chess **Jack Clark** **Christopher Berner**

Sam McCandlish **Alec Radford** **Ilya Sutskever** **Dario Amodei**

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

*Equal contribution

†Johns Hopkins University, OpenAI

Author contributions listed at end of paper.

Contents

1 Introduction

Recent years have featured a trend towards pre-trained language representations in NLP systems, applied in increasingly flexible and task-agnostic ways for downstream transfer. First, single-layer representations were learned using word vectors [MCCD13, PSM14] and fed to task-specific architectures, then RNNs with multiple layers of representations and contextual state were used to form stronger representations [DL15, MBXS17, PNZtY18] (though still applied to task-specific architectures), and more recently pre-trained recurrent or transformer language models [VSP+17] have been directly fine-tuned, entirely removing the need for task-specific architectures [RNSS18, DCLT18, HR18].

This last paradigm has led to substantial progress on many challenging NLP tasks such as reading comprehension, question answering, textual entailment, and many others, and has continued to advance based on new architectures and algorithms [RSR+19, LOG+19, YDY+19, LCG+19]. However, a major limitation to this approach is that while the architecture is task-agnostic, there is still a need for task-specific datasets and task-specific fine-tuning: to achieve strong performance on a desired task typically requires fine-tuning on a dataset of thousands to hundreds of thousands of examples specific to that task. Removing this limitation would be desirable, for several reasons.

First, from a practical perspective, the need for a large dataset of labeled examples for every new task limits the applicability of language models. There exists a very wide range of possible useful language tasks, encompassing anything from correcting grammar, to generating examples of an abstract concept, to critiquing a short story. For many of these tasks it is difficult to collect a large supervised training dataset, especially when the process must be repeated for every new task.

Second, the potential to exploit spurious correlations in training data fundamentally grows with the expressiveness of the model and the narrowness of the training distribution. This can create problems for the pre-training plus fine-tuning paradigm, where models are designed to be large to absorb information during pre-training, but are then fine-tuned on very narrow task distributions. For instance [HLW+20] observe that larger models do not necessarily generalize better out-of-distribution. There is evidence that suggests that the generalization achieved under this paradigm can be poor because the model is overly specific to the training distribution and does not generalize well outside it [YdC+19, MPL19]. Thus, the performance of fine-tuned models on specific benchmarks, even when it is nominally at human-level, may exaggerate actual performance on the underlying task [GSL+18, NK19].

Third, humans do not require large supervised datasets to learn most language tasks – a brief directive in natural language (e.g. "please tell me if this sentence describes something happy or something sad") or at most a tiny number of demonstrations (e.g. "here are two examples of people acting brave; please give a third example of bravery") is often

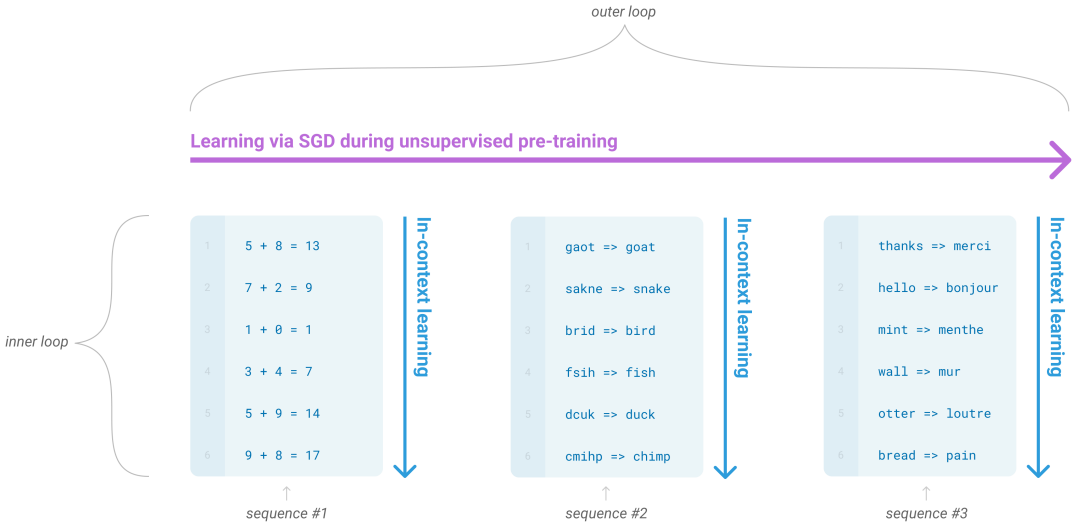

Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

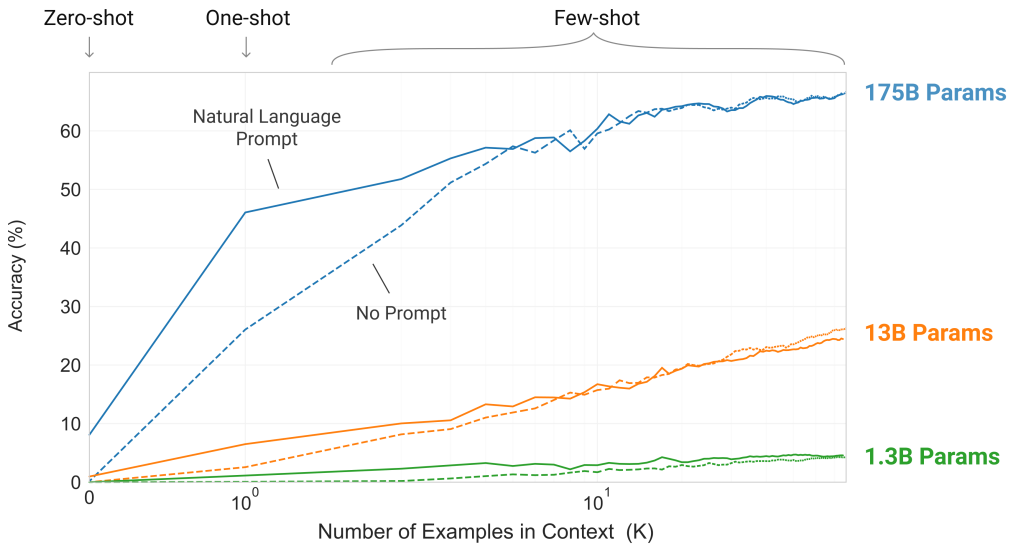

Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

sufficient to enable a human to perform a new task to at least a reasonable degree of competence. Aside from pointing to a conceptual limitation in our current NLP techniques, this adaptability has practical advantages – it allows humans to seamlessly mix together or switch between many tasks and skills, for example performing addition during a lengthy dialogue. To be broadly useful, we would someday like our NLP systems to have this same fluidity and generality.

One potential route towards addressing these issues is meta-learning[1] – which in the context of language models means the model develops a broad set of skills and pattern recognition abilities at training time, and then uses those abilities at inference time to rapidly adapt to or recognize the desired task (illustrated in Figure 1.1). Recent work [RWC+19] attempts to do this via what we call "in-context learning", using the text input of a pretrained language model as a form of task specification: the model is conditioned on a natural language instruction and/or a few demonstrations of the task and is then expected to complete further instances of the task simply by predicting what comes next.

While it has shown some initial promise, this approach still achieves results far inferior to fine-tuning – for example [RWC+19] achieves only 4% on Natural Questions, and even its 55 F1 CoQa result is now more than 35 points behind the state of the art. Meta-learning clearly requires substantial improvement in order to be viable as a practical method of solving language tasks.

Another recent trend in language modeling may offer a way forward. In recent years the capacity of transformer language models has increased substantially, from 100 million parameters [RNSS18], to 300 million parameters [DCLT18], to 1.5 billion parameters [RWC+19], to 8 billion parameters [SPP+19], 11 billion parameters [RSR+19], and finally 17 billion parameters [Tur20]. Each increase has brought improvements in text synthesis and/or downstream NLP tasks, and there is evidence suggesting that log loss, which correlates well with many downstream tasks, follows a smooth trend of improvement with scale [KMH+20]. Since in-context learning involves absorbing many skills and tasks within the parameters of the model, it is plausible that in-context learning abilities might show similarly strong gains with scale.

[1] In the context of language models this has sometimes been called "zero-shot transfer", but this term is potentially ambiguous: the method is "zero-shot" in the sense that no gradient updates are performed, but it often involves providing inference-time demonstrations to the model, so is not truly learning from zero examples. To avoid this confusion, we use the term "meta-learning" to capture the inner-loop / outer-loop structure of the general method, and the term "in context-learning" to refer to the inner loop of meta-learning. We further specialize the description to "zero-shot", "one-shot", or "few-shot" depending on how many demonstrations are provided at inference time. These terms are intended to remain agnostic on the question of whether the model learns new tasks from scratch at inference time or simply recognizes patterns seen during training – this is an important issue which we discuss later in the paper, but "meta-learning" is intended to encompass both possibilities, and simply describes the inner-outer loop structure.

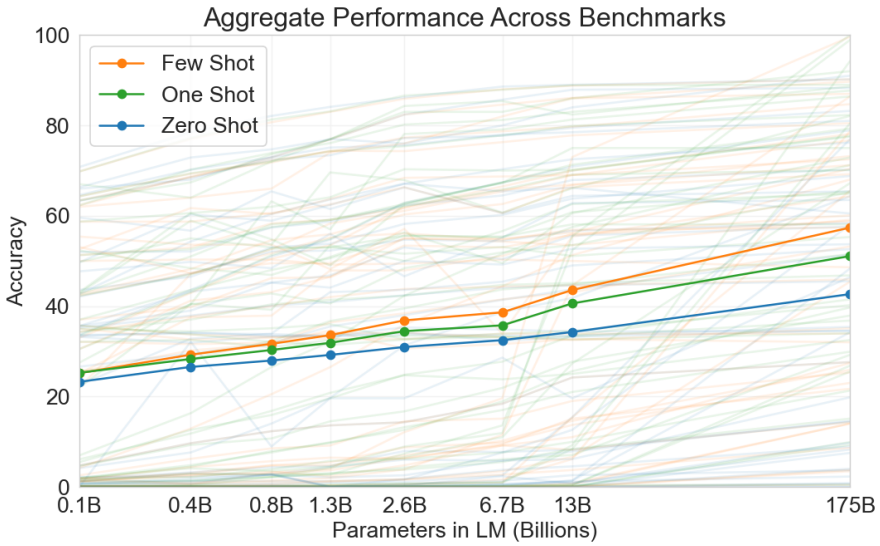

Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

In this paper, we test this hypothesis by training a 175 billion parameter autoregressive language model, which we call GPT-3, and measuring its in-context learning abilities. Specifically, we evaluate GPT-3 on over two dozen NLP datasets, as well as several novel tasks designed to test rapid adaptation to tasks unlikely to be directly contained in the training set. For each task, we evaluate GPT-3 under 3 conditions: (a) "few-shot learning", or in-context learning where we allow as many demonstrations as will fit into the model's context window (typically 10 to 100), (b) "one-shot learning", where we allow only one demonstration, and (c) "zero-shot" learning, where no demonstrations are allowed and only an instruction in natural language is given to the model. GPT-3 could also in principle be evaluated in the traditional fine-tuning setting, but we leave this to future work.

Figure 1.2 illustrates the conditions we study, and shows few-shot learning of a simple task requiring the model to remove extraneous symbols from a word. Model performance improves with the addition of a natural language task description, and with the number of examples in the model's context, K. Few-shot learning also improves dramatically with model size. Though the results in this case are particularly striking, the general trends with both model size and number of examples in-context hold for most tasks we study. We emphasize that these "learning" curves involve no gradient updates or fine-tuning, just increasing numbers of demonstrations given as conditioning.

Broadly, on NLP tasks GPT-3 achieves promising results in the zero-shot and one-shot settings, and in the the few-shot setting is sometimes competitive with or even occasionally surpasses state-of-the-art (despite state-of-the-art being held by fine-tuned models). For example, GPT-3 achieves 81.5 F1 on CoQA in the zero-shot setting, 84.0 F1 on CoQA in the one-shot setting, 85.0 F1 in the few-shot setting. Similarly, GPT-3 achieves 64.3% accuracy on TriviaQA in the zero-shot setting, 68.0% in the one-shot setting, and 71.2% in the few-shot setting, the last of which is state-of-the-art relative to fine-tuned models operating in the same closed-book setting.

GPT-3 also displays one-shot and few-shot proficiency at tasks designed to test rapid adaption or on-the-fly reasoning, which include unscrambling words, performing arithmetic, and using novel words in a sentence after seeing them defined only once. We also show that in the few-shot setting, GPT-3 can generate synthetic news articles which human evaluators have difficulty distinguishing from human-generated articles.

At the same time, we also find some tasks on which few-shot performance struggles, even at the scale of GPT-3. This includes natural language inference tasks like the ANLI dataset, and some reading comprehension datasets like RACE or QuAC. By presenting a broad characterization of GPT-3's strengths and weaknesses, including these limitations, we hope to stimulate study of few-shot learning in language models and draw attention to where progress is most needed.

A heuristic sense of the overall results can be seen in Figure 1.3, which aggregates the various tasks (though it should not be seen as a rigorous or meaningful benchmark in itself).

We also undertake a systematic study of "data contamination" – a growing problem when training high capacity models on datasets such as Common Crawl, which can potentially include content from test datasets simply because such content often exists on the web. In this paper we develop systematic tools to measure data contamination and quantify its distorting effects. Although we find that data contamination has a minimal effect on GPT-3's performance on most datasets, we do identify a few datasets where it could be inflating results, and we either do not report results on these datasets or we note them with an asterisk, depending on the severity.

In addition to all the above, we also train a series of smaller models (ranging from 125 million parameters to 13 billion parameters) in order to compare their performance to GPT-3 in the zero, one and few-shot settings. Broadly, for most tasks we find relatively smooth scaling with model capacity in all three settings; one notable pattern is that the gap between zero-, one-, and few-shot performance often grows with model capacity, perhaps suggesting that larger models are more proficient meta-learners.

Finally, given the broad spectrum of capabilities displayed by GPT-3, we discuss concerns about bias, fairness, and broader societal impacts, and attempt a preliminary analysis of GPT-3's characteristics in this regard.

The remainder of this paper is organized as follows. In Section 2, we describe our approach and methods for training GPT-3 and evaluating it. Section 3 presents results on the full range of tasks in the zero-, one- and few-shot settings. Section 4 addresses questions of data contamination (train-test overlap). Section 5 discusses limitations of GPT-3. Section 6 discusses broader impacts. Section 7 reviews related work and Section 8 concludes.

2 Approach

Our basic pre-training approach, including model, data, and training, is similar to the process described in [RWC+19], with relatively straightforward scaling up of the model size, dataset size and diversity, and length of training. Our use of in-context learning is also similar to [RWC+19], but in this work we systematically explore different settings for learning within the context. Therefore, we start this section by explicitly defining and contrasting the different settings that we will be evaluating GPT-3 on or could in principle evaluate GPT-3 on. These settings can be seen as lying on a spectrum of how much task-specific data they tend to rely on. Specifically, we can identify at least four points on this spectrum (see Figure 2.1 for an illustration):

- **Fine-Tuning (FT)** has been the most common approach in recent years, and involves updating the weights of a pre-trained model by training on a supervised dataset specific to the desired task. Typically thousands to hundreds of thousands of labeled examples are used. The main advantage of fine-tuning is strong performance on many benchmarks. The main disadvantages are the need for a new large dataset for every task, the potential for poor generalization out-of-distribution [MPL19], and the potential to exploit spurious features of the training data [GSL+18, NK19], potentially resulting in an unfair comparison with human performance. In this work we do not fine-tune GPT-3 because our focus is on task-agnostic performance, but GPT-3 can be fine-tuned in principle and this is a promising direction for future work.

- **Few-Shot (FS)** is the term we will use in this work to refer to the setting where the model is given a few demonstrations of the task at inference time as conditioning [RWC+19], but no weight updates are allowed. As shown in Figure 2.1, for a typical dataset an example has a context and a desired completion (for example an English sentence and the French translation), and few-shot works by giving K examples of context and completion, and then one final example of context, with the model expected to provide the completion. We typically set K in the range of 10 to 100 as this is how many examples can fit in the model's context window ($n_{\text{ctx}} = 2048$). The main advantages of few-shot are a major reduction in the need for task-specific data and reduced potential to learn an overly narrow distribution from a large but narrow fine-tuning dataset. The main disadvantage is that results from this method have so far been much worse than state-of-the-art fine-tuned models. Also, a small amount of task specific data is still required. As indicated by the name, few-shot learning as described here for language models is related to few-shot learning as used in other contexts in ML [HYC01, VBL+16] – both involve learning based on a broad distribution of tasks (in this case implicit in the pre-training data) and then rapidly adapting to a new task.

- **One-Shot (1S)** is the same as few-shot except that only one demonstration is allowed, in addition to a natural language description of the task, as shown in Figure 1. The reason to distinguish one-shot from few-shot and zero-shot (below) is that it most closely matches the way in which some tasks are communicated to humans. For example, when asking humans to generate a dataset on a human worker service (for example Mechanical Turk), it is common to give one demonstration of the task. By contrast it is sometimes difficult to communicate the content or format of a task if no examples are given.

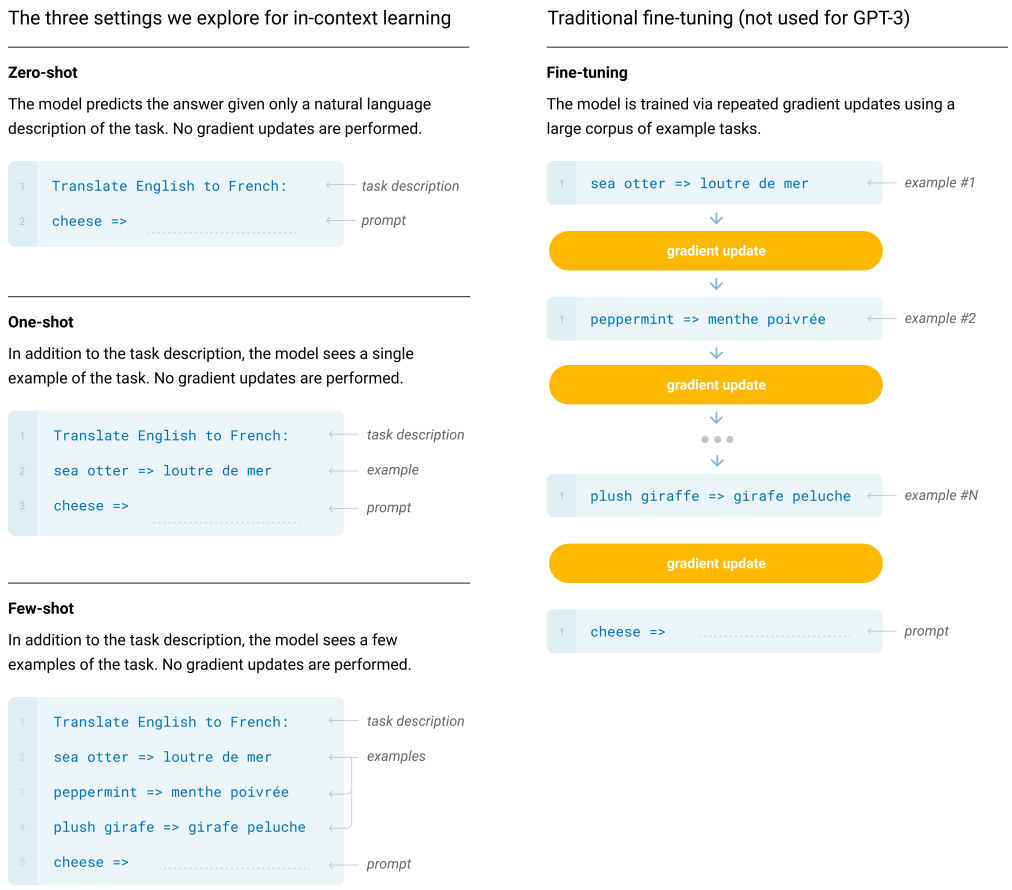

Figure 2.1: Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning. The panels above show four methods for performing a task with a language model – fine-tuning is the traditional method, whereas zero-, one-, and few-shot, which we study in this work, require the model to perform the task with only forward passes at test time. We typically present the model with a few dozen examples in the few shot setting. Exact phrasings for all task descriptions, examples and prompts can be found in Appendix G.

- **Zero-Shot (0S)** is the same as one-shot except that no demonstrations are allowed, and the model is only given a natural language instruction describing the task. This method provides maximum convenience, potential for robustness, and avoidance of spurious correlations (unless they occur very broadly across the large corpus of pre-training data), but is also the most challenging setting. In some cases it may even be difficult for humans to understand the format of the task without prior examples, so this setting is in some cases "unfairly hard". For example, if someone is asked to "make a table of world records for the 200m dash", this request can be ambiguous, as it may not be clear exactly what format the table should have or what should be included (and even with careful clarification, understanding precisely what is desired can be difficult). Nevertheless, for at least some settings zero-shot is closest to how humans perform tasks – for example, in the translation example in Figure 2.1, a human would likely know what to do from just the text instruction.

Figure 2.1 shows the four methods using the example of translating English to French. In this paper we focus on zero-shot, one-shot and few-shot, with the aim of comparing them not as competing alternatives, but as different problem settings which offer a varying trade-off between performance on specific benchmarks and sample efficiency. We especially highlight the few-shot results as many of them are only slightly behind state-of-the-art fine-tuned models. Ultimately, however, one-shot, or even sometimes zero-shot, seem like the fairest comparisons to human performance, and are important targets for future work.

Sections 2.1-2.3 below give details on our models, training data, and training process respectively. Section 2.4 discusses the details of how we do few-shot, one-shot, and zero-shot evaluations.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

2.1 Model and Architectures

We use the same model and architecture as GPT-2 [RWC+19], including the modified initialization, pre-normalization, and reversible tokenization described therein, with the exception that we use alternating dense and locally banded sparse attention patterns in the layers of the transformer, similar to the Sparse Transformer [CGRS19]. To study the dependence of ML performance on model size, we train 8 different sizes of model, ranging over three orders of magnitude from 125 million parameters to 175 billion parameters, with the last being the model we call GPT-3. Previous work [KMH+20] suggests that with enough training data, scaling of validation loss should be approximately a smooth power law as a function of size; training models of many different sizes allows us to test this hypothesis both for validation loss and for downstream language tasks.

Table 2.1 shows the sizes and architectures of our 8 models. Here n_{params} is the total number of trainable parameters, n_{layers} is the total number of layers, d_{model} is the number of units in each bottleneck layer (we always have the feedforward layer four times the size of the bottleneck layer, $d_{\text{ff}} = 4 * d_{\text{model}}$), and d_{head} is the dimension of each attention head. All models use a context window of $n_{\text{ctx}} = 2048$ tokens. We partition the model across GPUs along both the depth and width dimension in order to minimize data-transfer between nodes. The precise architectural parameters for each model are chosen based on computational efficiency and load-balancing in the layout of models across GPU's. Previous work [KMH+20] suggests that validation loss is not strongly sensitive to these parameters within a reasonably broad range.

2.2 Training Dataset

Datasets for language models have rapidly expanded, culminating in the Common Crawl dataset[2] [RSR+19] constituting nearly a trillion words. This size of dataset is sufficient to train our largest models without ever updating on the same sequence twice. However, we have found that unfiltered or lightly filtered versions of Common Crawl tend to have lower quality than more curated datasets. Therefore, we took 3 steps to improve the average quality of our datasets: (1) we downloaded and filtered a version of CommonCrawl based on similarity to a range of high-quality reference corpora, (2) we performed fuzzy deduplication at the document level, within and across datasets, to prevent redundancy and preserve the integrity of our held-out validation set as an accurate measure of overfitting, and (3) we also added known high-quality reference corpora to the training mix to augment CommonCrawl and increase its diversity.

Details of the first two points (processing of Common Crawl) are described in Appendix A. For the third, we added several curated high-quality datasets, including an expanded version of the WebText dataset [RWC+19], collected by scraping links over a longer period of time, and first described in [KMH+20], two internet-based books corpora (Books1 and Books2) and English-language Wikipedia.

Table 2.2 shows the final mixture of datasets that we used in training. The CommonCrawl data was downloaded from 41 shards of monthly CommonCrawl covering 2016 to 2019, constituting 45TB of compressed plaintext before filtering and 570GB after filtering, roughly equivalent to 400 billion byte-pair-encoded tokens. Note that during training, datasets are not sampled in proportion to their size, but rather datasets we view as higher-quality are sampled more frequently, such that CommonCrawl and Books2 datasets are sampled less than once during training, but the other datasets are sampled 2-3 times. This essentially accepts a small amount of overfitting in exchange for higher quality training data.

[2] https://commoncrawl.org/the-data/

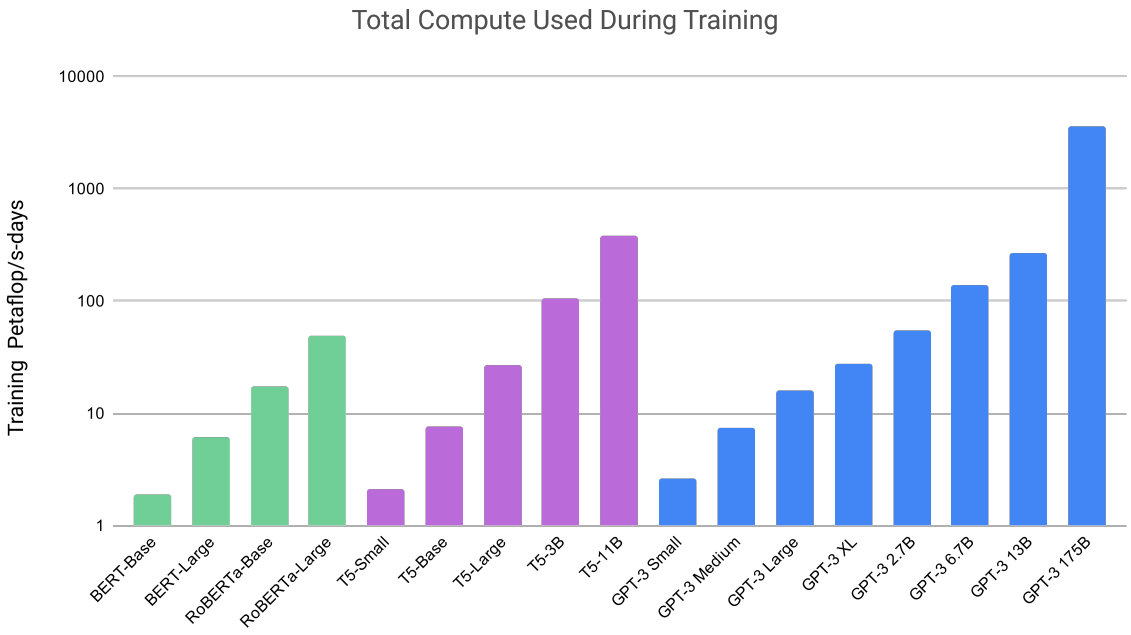

Figure 2.2: Total compute used during training. Based on the analysis in Scaling Laws For Neural Language Models [KMH+20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

A major methodological concern with language models pretrained on a broad swath of internet data, particularly large models with the capacity to memorize vast amounts of content, is potential contamination of downstream tasks by having their test or development sets inadvertently seen during pre-training. To reduce such contamination, we searched for and attempted to remove any overlaps with the development and test sets of all benchmarks studied in this paper. Unfortunately, a bug in the filtering caused us to ignore some overlaps, and due to the cost of training it was not feasible to retrain the model. In Section 4 we characterize the impact of the remaining overlaps, and in future work we will more aggressively remove data contamination.

2.3 Training Process

As found in [KMH+20, MKAT18], larger models can typically use a larger batch size, but require a smaller learning rate. We measure the gradient noise scale during training and use it to guide our choice of batch size [MKAT18]. Table 2.1 shows the parameter settings we used. To train the larger models without running out of memory, we use a mixture of model parallelism within each matrix multiply and model parallelism across the layers of the network. All models were trained on V100 GPU's on part of a high-bandwidth cluster provided by Microsoft. Details of the training process and hyperparameter settings are described in Appendix B.

2.4 Evaluation

For few-shot learning, we evaluate each example in the evaluation set by randomly drawing K examples from that task's training set as conditioning, delimited by 1 or 2 newlines depending on the task. For LAMBADA and Storycloze there is no supervised training set available so we draw conditioning examples from the development set and evaluate on the test set. For Winograd (the original, not SuperGLUE version) there is only one dataset, so we draw conditioning examples directly from it.

K can be any value from 0 to the maximum amount allowed by the model's context window, which is $n_{\text{ctx}} = 2048$ for all models and typically fits 10 to 100 examples. Larger values of K are usually but not always better, so when a separate development and test set are available, we experiment with a few values of K on the development set and then run the best value on the test set. For some tasks (see Appendix G) we also use a natural language prompt in addition to (or for $K = 0$, instead of) demonstrations.

On tasks that involve choosing one correct completion from several options (multiple choice), we provide K examples of context plus correct completion, followed by one example of context only, and compare the LM likelihood of each completion. For most tasks we compare the per-token likelihood (to normalize for length), however on a small number of datasets (ARC, OpenBookQA, and RACE) we gain additional benefit as measured on the development set by normalizing by the unconditional probability of each completion, by computing $\frac{P(\text{completion}|\text{context})}{P(\text{completion}|\text{answer\_context})}$, where answer_context is the string `"Answer: "` or `"A: "` and is used to prompt that the completion should be an answer but is otherwise generic.

On tasks that involve binary classification, we give the options more semantically meaningful names (e.g. "True" or "False" rather than 0 or 1) and then treat the task like multiple choice; we also sometimes frame the task similar to what is done by [RSR+19] (see Appendix G) for details.

On tasks with free-form completion, we use beam search with the same parameters as [RSR+19]: a beam width of 4 and a length penalty of $\alpha = 0.6$. We score the model using F1 similarity score, BLEU, or exact match, depending on what is standard for the dataset at hand.

Final results are reported on the test set when publicly available, for each model size and learning setting (zero-, one-, and few-shot). When the test set is private, our model is often too large to fit on the test server, so we report results on the development set. We do submit to the test server on a small number of datasets (SuperGLUE, TriviaQA, PiQa) where we were able to make submission work, and we submit only the 200B few-shot results, and report development set results for everything else.

3 Results

In Figure 3.1 we display training curves for the 8 models described in Section 2. For this graph we also include 6 additional extra-small models with as few as 100,000 parameters. As observed in [KMH+20], language modeling performance follows a power-law when making efficient use of training compute. After extending this trend by two more orders of magnitude, we observe only a slight (if any) departure from the power-law. One might worry that these improvements in cross-entropy loss come only from modeling spurious details of our training corpus. However, we will see in the following sections that improvements in cross-entropy loss lead to consistent performance gains across a broad spectrum of natural language tasks.

Below, we evaluate the 8 models described in Section 2 (the 175 billion parameter parameter GPT-3 and 7 smaller models) on a wide range of datasets. We group the datasets into 9 categories representing roughly similar tasks.

In Section 3.1 we evaluate on traditional language modeling tasks and tasks that are similar to language modeling, such as Cloze tasks and sentence/paragraph completion tasks. In Section 3.2 we evaluate on "closed book" question answering tasks: tasks which require using the information stored in the model's parameters to answer general knowledge questions. In Section 3.3 we evaluate the model's ability to translate between languages (especially one-shot and few-shot). In Section 3.4 we evaluate the model's performance on Winograd Schema-like tasks. In Section 3.5 we evaluate on datasets that involve commonsense reasoning or question answering. In Section 3.6 we evaluate on reading comprehension tasks, in Section 3.7 we evaluate on the SuperGLUE benchmark suite, and in 3.8 we briefly explore NLI. Finally, in Section 3.9, we invent some additional tasks designed especially to probe in-context learning abilities – these tasks focus on on-the-fly reasoning, adaptation skills, or open-ended text synthesis. We evaluate all tasks in the few-shot, one-shot, and zero-shot settings.

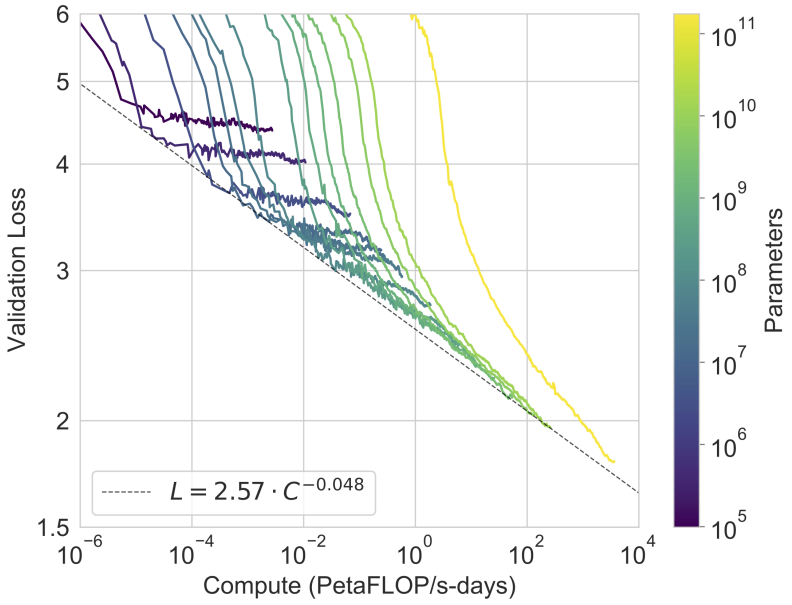

Figure 3.1: Smooth scaling of performance with compute. Performance (measured in terms of cross-entropy validation loss) follows a power-law trend with the amount of compute used for training. The power-law behavior observed in [KMH+20] continues for an additional two orders of magnitude with only small deviations from the predicted curve. For this figure, we exclude embedding parameters from compute and parameter counts.

Setting	PTB
SOTA (Zero-Shot)	35.8[a]
GPT-3 Zero-Shot	**20.5**

Table 3.1: Zero-shot results on PTB language modeling dataset. Many other common language modeling datasets are omitted because they are derived from Wikipedia or other sources which are included in GPT-3's training data. [a][RWC+19]

3.1 Language Modeling, Cloze, and Completion Tasks

In this section we test GPT-3's performance on the traditional task of language modeling, as well as related tasks that involve predicting a single word of interest, completing a sentence or paragraph, or choosing between possible completions of a piece of text.

3.1.1 Language Modeling

We calculate zero-shot perplexity on the Penn Tree Bank (PTB) [MKM+94] dataset measured in [RWC+19]. We omit the 4 Wikipedia-related tasks in that work because they are entirely contained in our training data, and we also omit the one-billion word benchmark due to a high fraction of the dataset being contained in our training set. PTB escapes these issues due to predating the modern internet. Our largest model sets a new SOTA on PTB by a substantial margin of 15 points, achieving a perplexity of 20.50. Note that since PTB is a traditional language modeling dataset it does not have a clear separation of examples to define one-shot or few-shot evaluation around, so we measure only zero-shot.

3.1.2 LAMBADA

The LAMBADA dataset [PKL+16] tests the modeling of long-range dependencies in text – the model is asked to predict the last word of sentences which require reading a paragraph of context. It has recently been suggested that the continued scaling of language models is yielding diminishing returns on this difficult benchmark. [BHT+20] reflect on the small 1.5% improvement achieved by a doubling of model size between two recent state of the art results ([SPP+19]

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

Table 3.2: Performance on cloze and completion tasks. GPT-3 significantly improves SOTA on LAMBADA while achieving respectable performance on two difficult completion prediction datasets. [a][Tur20] [b][RWC+19] [c][LDL19] [d][LCH+20]

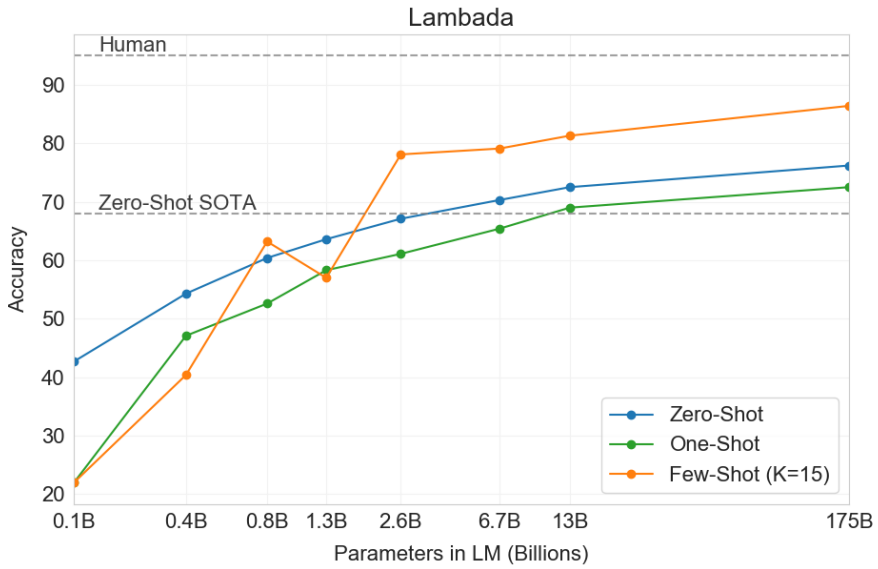

Figure 3.2: On LAMBADA, the few-shot capability of language models results in a strong boost to accuracy. GPT-3 2.7B outperforms the SOTA 17B parameter Turing-NLG [Tur20] in this setting, and GPT-3 175B advances the state of the art by 18%. Note zero-shot uses a different format from one-shot and few-shot as described in the text.

and [Tur20]) and argue that "continuing to expand hardware and data sizes by orders of magnitude is not the path forward". We find that path is still promising and in a zero-shot setting GPT-3 achieves 76% on LAMBADA, a gain of 8% over the previous state of the art.

LAMBADA is also a demonstration of the flexibility of few-shot learning as it provides a way to address a problem that classically occurs with this dataset. Although the completion in LAMBADA is always the last word in a sentence, a standard language model has no way of knowing this detail. It thus assigns probability not only to the correct ending but also to other valid continuations of the paragraph. This problem has been partially addressed in the past with stop-word filters [RWC+19] (which ban "continuation" words). The few-shot setting instead allows us to "frame" the task as a cloze-test and allows the language model to infer from examples that a completion of exactly one word is desired. We use the following fill-in-the-blank format:

Alice was friends with Bob. Alice went to visit her friend _____. → Bob

George bought some baseball equipment, a ball, a glove, and a _____. →

When presented with examples formatted this way, GPT-3 achieves 86.4% accuracy in the few-shot setting, an increase of over 18% from the previous state-of-the-art. We observe that few-shot performance improves strongly with model size. While this setting decreases the performance of the smallest model by almost 20%, for GPT-3 it improves accuracy by 10%. Finally, the fill-in-blank method is not effective one-shot, where it always performs worse than the zero-shot setting. Perhaps this is because all models still require several examples to recognize the pattern.

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

Table 3.3: Results on three Open-Domain QA tasks. GPT-3 is shown in the few-, one-, and zero-shot settings, as compared to prior SOTA results for closed book and open domain settings. TriviaQA few-shot result is evaluated on the wiki split test server.

One note of caution is that an analysis of test set contamination identified that a significant minority of the LAMBADA dataset appears to be present in our training data – however analysis performed in Section 4 suggests negligible impact on performance.

3.1.3 HellaSwag

The HellaSwag dataset [ZHB+19] involves picking the best ending to a story or set of instructions. The examples were adversarially mined to be difficult for language models while remaining easy for humans (who achieve 95.6% accuracy). GPT-3 achieves 78.1% accuracy in the one-shot setting and 79.3% accuracy in the few-shot setting, outperforming the 75.4% accuracy of a fine-tuned 1.5B parameter language model [ZHR+19] but still a fair amount lower than the overall SOTA of 85.6% achieved by the fine-tuned multi-task model ALUM.

3.1.4 StoryCloze

We next evaluate GPT-3 on the StoryCloze 2016 dataset [MCH+16], which involves selecting the correct ending sentence for five-sentence long stories. Here GPT-3 achieves 83.2% in the zero-shot setting and 87.7% in the few-shot setting (with $K = 70$). This is still 4.1% lower than the fine-tuned SOTA using a BERT based model [LDL19] but improves over previous zero-shot results by roughly 10%.

3.2 Closed Book Question Answering

In this section we measure GPT-3's ability to answer questions about broad factual knowledge. Due to the immense amount of possible queries, this task has normally been approached by using an information retrieval system to find relevant text in combination with a model which learns to generate an answer given the question and the retrieved text. Since this setting allows a system to search for and condition on text which potentially contains the answer it is denoted "open-book". [RRS20] recently demonstrated that a large language model can perform surprisingly well directly answering the questions without conditioning on auxilliary information. They denote this more restrictive evaluation setting as "closed-book". Their work suggests that even higher-capacity models could perform even better and we test this hypothesis with GPT-3. We evaluate GPT-3 on the 3 datasets in [RRS20]: Natural Questions [KPR+19], WebQuestions [BCFL13], and TriviaQA [JCWZ17], using the same splits. Note that in addition to all results being in the closed-book setting, our use of few-shot, one-shot, and zero-shot evaluations represent an even stricter setting than previous closed-book QA work: in addition to external content not being allowed, fine-tuning on the Q&A dataset itself is also not permitted.

The results for GPT-3 are shown in Table 3.3. On TriviaQA, we achieve 64.3% in the zero-shot setting, 68.0% in the one-shot setting, and 71.2% in the few-shot setting. The zero-shot result already outperforms the fine-tuned T5-11B by 14.2%, and also outperforms a version with Q&A tailored span prediction during pre-training by 3.8%. The one-shot result improves by 3.7% and matches the SOTA for an open-domain QA system which not only fine-tunes but also makes use of a learned retrieval mechanism over a 15.3B parameter dense vector index of 21M documents [LPP+20]. GPT-3's few-shot result further improves performance another 3.2% beyond this.

On WebQuestions (WebQs), GPT-3 achieves 14.4% in the zero-shot setting, 25.3% in the one-shot setting, and 41.5% in the few-shot setting. This compares to 37.4% for fine-tuned T5-11B, and 44.7% for fine-tuned T5-11B+SSM, which uses a Q&A-specific pre-training procedure. GPT-3 in the few-shot setting approaches the performance of state-of-the-art fine-tuned models. Notably, compared to TriviaQA, WebQS shows a much larger gain from zero-shot to few-shot (and indeed its zero-shot and one-shot performance are poor), perhaps suggesting that the WebQs questions

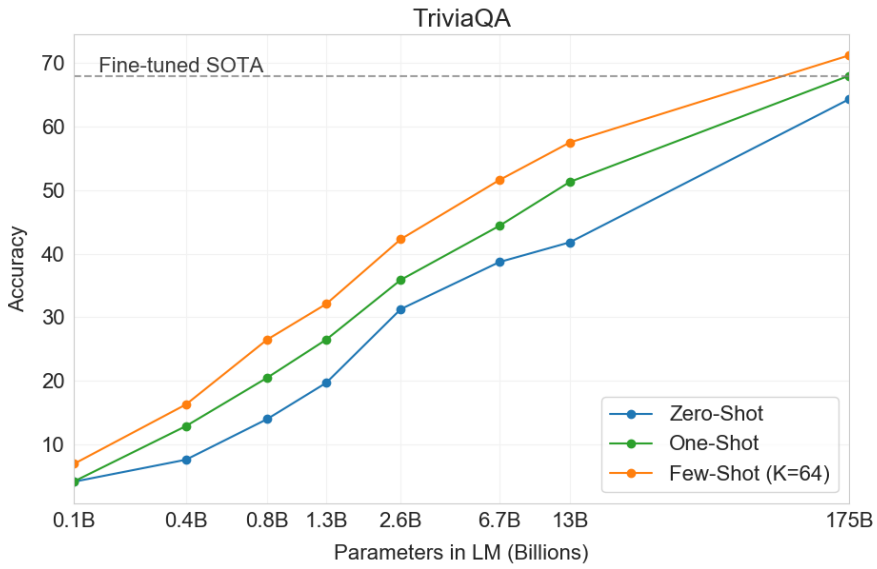

Figure 3.3: On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

and/or the style of their answers are out-of-distribution for GPT-3. Nevertheless, GPT-3 appears able to adapt to this distribution, recovering strong performance in the few-shot setting.

On Natural Questions (NQs) GPT-3 achieves 14.6% in the zero-shot setting, 23.0% in the one-shot setting, and 29.9% in the few-shot setting, compared to 36.6% for fine-tuned T5 11B+SSM. Similar to WebQS, the large gain from zero-shot to few-shot may suggest a distribution shift, and may also explain the less competitive performance compared to TriviaQA and WebQS. In particular, the questions in NQs tend towards very fine-grained knowledge on Wikipedia specifically which could be testing the limits of GPT-3's capacity and broad pretraining distribution.

Overall, on one of the three datasets GPT-3's one-shot matches the open-domain fine-tuning SOTA. On the other two datasets it approaches the performance of the closed-book SOTA despite not using fine-tuning. On all 3 datasets, we find that performance scales very smoothly with model size (Figure 3.3 and Appendix H Figure H.7), possibly reflecting the idea that model capacity translates directly to more 'knowledge' absorbed in the parameters of the model.

3.3 Translation

For GPT-2 a filter was used on a multilingual collection of documents to produce an English only dataset due to capacity concerns. Even with this filtering GPT-2 showed some evidence of multilingual capability and performed non-trivially when translating between French and English despite only training on 10 megabytes of remaining French text. Since we increase the capacity by over two orders of magnitude from GPT-2 to GPT-3, we also expand the scope of the training dataset to include more representation of other languages, though this remains an area for further improvement. As discussed in 2.2 the majority of our data is derived from raw Common Crawl with only quality-based filtering. Although GPT-3's training data is still primarily English (93% by word count), it also includes 7% of text in other languages. These languages are documented in the supplemental material. In order to better understand translation capability, we also expand our analysis to include two additional commonly studied languages, German and Romanian.

Existing unsupervised machine translation approaches often combine pretraining on a pair of monolingual datasets with back-translation [SHB15] to bridge the two languages in a controlled way. By contrast, GPT-3 learns from a blend of training data that mixes many languages together in a natural way, combining them on a word, sentence, and document level. GPT-3 also uses a single training objective which is not customized or designed for any task in particular. However, our one / few-shot settings aren't strictly comparable to prior unsupervised work since they make use of a small amount of paired examples (1 or 64). This corresponds to up to a page or two of in-context training data.

Results are shown in Table 3.4. Zero-shot GPT-3, which only receives on a natural language description of the task, still underperforms recent unsupervised NMT results. However, providing only a single example demonstration for

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

Table 3.4: Few-shot GPT-3 outperforms previous unsupervised NMT work by 5 BLEU when translating into English reflecting its strength as an English LM. We report BLEU scores on the WMT'14 Fr↔En, WMT'16 De↔En, and WMT'16 Ro↔En datasets as measured by `multi-bleu.perl` with XLM's tokenization in order to compare most closely with prior unsupervised NMT work. SacreBLEU[f] [Pos18] results reported in Appendix H. Underline indicates an unsupervised or few-shot SOTA, bold indicates supervised SOTA with relative confidence. [a][EOAG18] [b][DHKH14] [c][WXH+18] [d][oR16] [e][LGG+20] [f][SacreBLEU signature: BLEU+case.mixed+numrefs.1+smooth.exp+tok.intl+version.1.2.20]

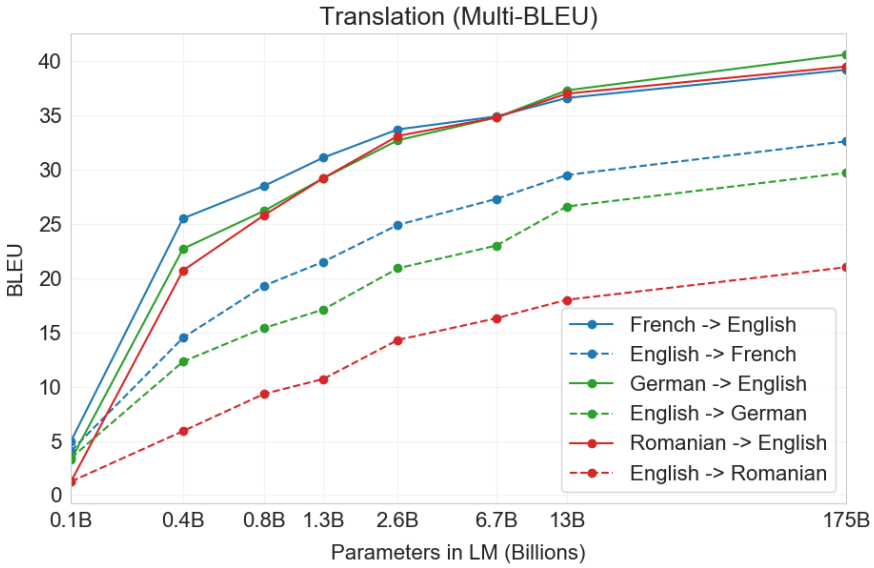

Figure 3.4: Few-shot translation performance on 6 language pairs as model capacity increases. There is a consistent trend of improvement across all datasets as the model scales, and as well as tendency for translation into English to be stronger than translation from English.

Setting	Winograd	Winogrande (XL)
Fine-tuned SOTA	**90.1**[a]	**84.6**[b]
GPT-3 Zero-Shot	88.3*	70.2
GPT-3 One-Shot	89.7*	73.2
GPT-3 Few-Shot	88.6*	77.7

Table 3.5: Results on the WSC273 version of Winograd schemas and the adversarial Winogrande dataset. See Section 4 for details on potential contamination of the Winograd test set. [a][SBBC19] [b][LYN+20]

Figure 3.5: Zero-, one-, and few-shot performance on the adversarial Winogrande dataset as model capacity scales. Scaling is relatively smooth with the gains to few-shot learning increasing with model size, and few-shot GPT-3 175B is competitive with a fine-tuned RoBERTA-large.

each translation task improves performance by over 7 BLEU and nears competitive performance with prior work. GPT-3 in the full few-shot setting further improves another 4 BLEU resulting in similar average performance to prior unsupervised NMT work. GPT-3 has a noticeable skew in its performance depending on language direction. For the three input languages studied, GPT-3 significantly outperforms prior unsupervised NMT work when translating into English but underperforms when translating in the other direction. Performance on En-Ro is a noticeable outlier at over 10 BLEU worse than prior unsupervised NMT work. This could be a weakness due to reusing the byte-level BPE tokenizer of GPT-2 which was developed for an almost entirely English training dataset. For both Fr-En and De-En, few shot GPT-3 outperforms the best supervised result we could find but due to our unfamiliarity with the literature and the appearance that these are un-competitive benchmarks we do not suspect those results represent true state of the art. For Ro-En, few shot GPT-3 performs within 0.5 BLEU of the overall SOTA which is achieved by a combination of unsupervised pretraining, supervised finetuning on 608K labeled examples, and backtranslation [LHCG19b].

Finally, across all language pairs and across all three settings (zero-, one-, and few-shot), there is a smooth trend of improvement with model capacity. This is shown in Figure 3.4 in the case of few-shot results, and scaling for all three settings is shown in Appendix H.

3.4 Winograd-Style Tasks

The Winograd Schemas Challenge [LDM12] is a classical task in NLP that involves determining which word a pronoun refers to, when the pronoun is grammatically ambiguous but semantically unambiguous to a human. Recently fine-tuned language models have achieved near-human performance on the original Winograd dataset, but more difficult versions

Setting	PIQA	ARC (Easy)	ARC (Challenge)	OpenBookQA
Fine-tuned SOTA	79.4	**92.0**[KKS+20]	**78.5**[KKS+20]	**87.2**[KKS+20]
GPT-3 Zero-Shot	**80.5***	68.8	51.4	57.6
GPT-3 One-Shot	**80.5***	71.2	53.2	58.8
GPT-3 Few-Shot	**82.8***	70.1	51.5	65.4

Table 3.6: GPT-3 results on three commonsense reasoning tasks, PIQA, ARC, and OpenBookQA. GPT-3 Few-Shot PIQA result is evaluated on the test server. See Section 4 for details on potential contamination issues on the PIQA test set.

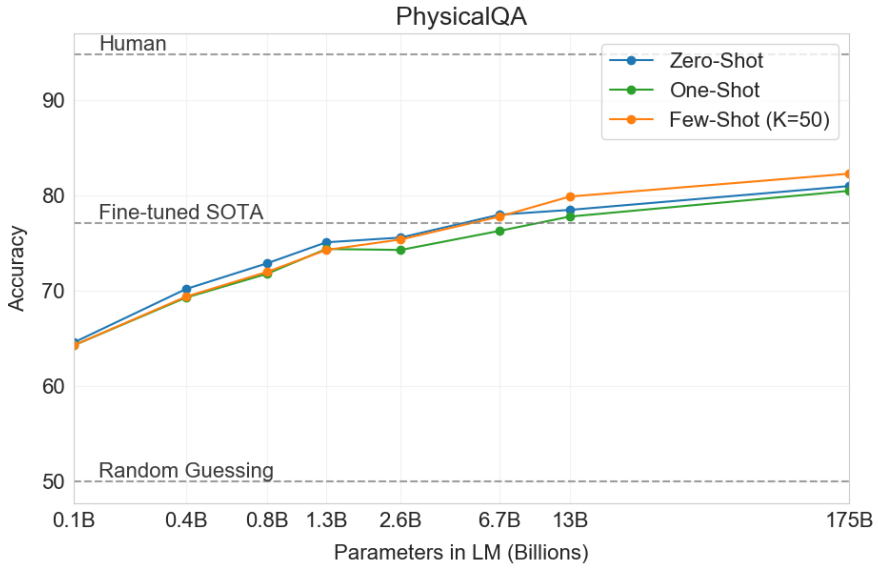

Figure 3.6: GPT-3 results on PIQA in the zero-shot, one-shot, and few-shot settings. The largest model achieves a score on the development set in all three conditions that exceeds the best recorded score on the task.

such as the adversarially-mined Winogrande dataset [SBBC19] still significantly lag human performance. We test GPT-3's performance on both Winograd and Winogrande, as usual in the zero-, one-, and few-shot setting.

On Winograd we test GPT-3 on the original set of 273 Winograd schemas, using the same "partial evaluation" method described in [RWC+19]. Note that this setting differs slightly from the WSC task in the SuperGLUE benchmark, which is presented as binary classification and requires entity extraction to convert to the form described in this section. On Winograd GPT-3 achieves 88.3%, 89.7%, and 88.6% in the zero-shot, one-shot, and few-shot settings, showing no clear in-context learning but in all cases achieving strong results just a few points below state-of-the-art and estimated human performance. We note that contamination analysis found some Winograd schemas in the training data but this appears to have only a small effect on results (see Section 4).

On the more difficult Winogrande dataset, we do find gains to in-context learning: GPT-3 achieves 70.2% in the zero-shot setting, 73.2% in the one-shot setting, and 77.7% in the few-shot setting. For comparison a fine-tuned RoBERTA model achieves 79%, state-of-the-art is 84.6% achieved with a fine-tuned high capacity model (T5), and human performance on the task as reported by [SBBC19] is 94.0%.

3.5 Common Sense Reasoning

Next we consider three datasets which attempt to capture physical or scientific reasoning, as distinct from sentence completion, reading comprehension, or broad knowledge question answering. The first, PhysicalQA (PIQA) [BZB+19], asks common sense questions about how the physical world works and is intended as a probe of grounded understanding of the world. GPT-3 achieves 81.0% accuracy zero-shot, 80.5% accuracy one-shot, and 82.8% accuracy few-shot (the last measured on PIQA's test server). This compares favorably to the 79.4% accuracy prior state-of-the-art of a

Setting	CoQA	DROP	QuAC	SQuADv2	RACE-h	RACE-m
Fine-tuned SOTA	**90.7**[a]	**89.1**[b]	**74.4**[c]	**93.0**[d]	**90.0**[e]	**93.1**[e]
GPT-3 Zero-Shot	81.5	23.6	41.5	59.5	45.5	58.4
GPT-3 One-Shot	84.0	34.3	43.3	65.4	45.9	57.4
GPT-3 Few-Shot	85.0	36.5	44.3	69.8	46.8	58.1

Table 3.7: Results on reading comprehension tasks. All scores are F1 except results for RACE which report accuracy. [a][JZC+19] [b][JN20] [c][AI19] [d][QIA20] [e][SPP+19]

fine-tuned RoBERTa. PIQA shows relatively shallow scaling with model size and is still over 10% worse than human performance, but GPT-3's few-shot and even zero-shot result outperform the current state-of-the-art. Our analysis flagged PIQA for a potential data contamination issue (despite hidden test labels), and we therefore conservatively mark the result with an asterisk. See Section 4 for details.

ARC [CCE+18] is a dataset of multiple-choice questions collected from 3rd to 9th grade science exams. On the "Challenge" version of the dataset which has been filtered to questions which simple statistical or information retrieval methods are unable to correctly answer, GPT-3 achieves 51.4% accuracy in the zero-shot setting, 53.2% in the one-shot setting, and 51.5% in the few-shot setting. This is approaching the performance of a fine-tuned RoBERTa baseline (55.9%) from UnifiedQA [KKS+20]. On the "Easy" version of the dataset (questions which either of the mentioned baseline approaches answered correctly), GPT-3 achieves 68.8%, 71.2%, and 70.1% which slightly exceeds a fine-tuned RoBERTa baseline from [KKS+20]. However, both of these results are still much worse than the overall SOTAs achieved by the UnifiedQA which exceeds GPT-3's few-shot results by 27% on the challenge set and 22% on the easy set.

On OpenBookQA [MCKS18], GPT-3 improves significantly from zero to few shot settings but is still over 20 points short of the overall SOTA. GPT-3's few-shot performance is similar to a fine-tuned BERT Large baseline on the leaderboard.

Overall, in-context learning with GPT-3 shows mixed results on commonsense reasoning tasks, with only small and inconsistent gains observed in the one and few-shot learning settings for both PIQA and ARC, but a significant improvement is observed on OpenBookQA. GPT-3 sets SOTA on the new PIQA dataset in all evaluation settings.

3.6 Reading Comprehension

Next we evaluate GPT-3 on the task of reading comprehension. We use a suite of 5 datasets including abstractive, multiple choice, and span based answer formats in both dialog and single question settings. We observe a wide spread in GPT-3's performance across these datasets suggestive of varying capability with different answer formats. In general we observe GPT-3 is on par with initial baselines and early results trained using contextual representations on each respective dataset.

GPT-3 performs best (within 3 points of the human baseline) on CoQA [RCM19] a free-form conversational dataset and performs worst (13 F1 below an ELMo baseline) on QuAC [CHI+18] a dataset which requires modeling structured dialog acts and answer span selections of teacher-student interactions. On DROP [DWD+19], a dataset testing discrete reasoning and numeracy in the context of reading comprehension, GPT-3 in a few-shot setting outperforms the fine-tuned BERT baseline from the original paper but is still well below both human performance and state-of-the-art approaches which augment neural networks with symbolic systems [RLL+19]. On SQuAD 2.0 [RJL18], GPT-3 demonstrates its few-shot learning capabilities, improving by almost 10 F1 (to 69.8) compared to a zero-shot setting. This allows it to slightly outperform the best fine-tuned result in the original paper. On RACE [LXL+17], a multiple choice dataset of middle school and high school english examinations, GPT-3 performs relatively weakly and is only competitive with the earliest work utilizing contextual representations and is still 45% behind SOTA.

3.7 SuperGLUE

In order to better aggregate results on NLP tasks and compare to popular models such as BERT and RoBERTa in a more systematic way, we also evaluate GPT-3 on a standardized collection of datasets, the SuperGLUE benchmark [WPN+19] [WPN+19] [CLC+19] [DMST19] [RBG11] [KCR+18] [ZLL+18] [DGM06] [BHDD+06] [GMDD07] [BDD+09] [PCC18] [PHR+18]. GPT-3's test-set performance on the SuperGLUE dataset is shown in Table 3.8. In the few-shot setting, we used 32 examples for all tasks, sampled randomly from the training set. For all tasks except WSC

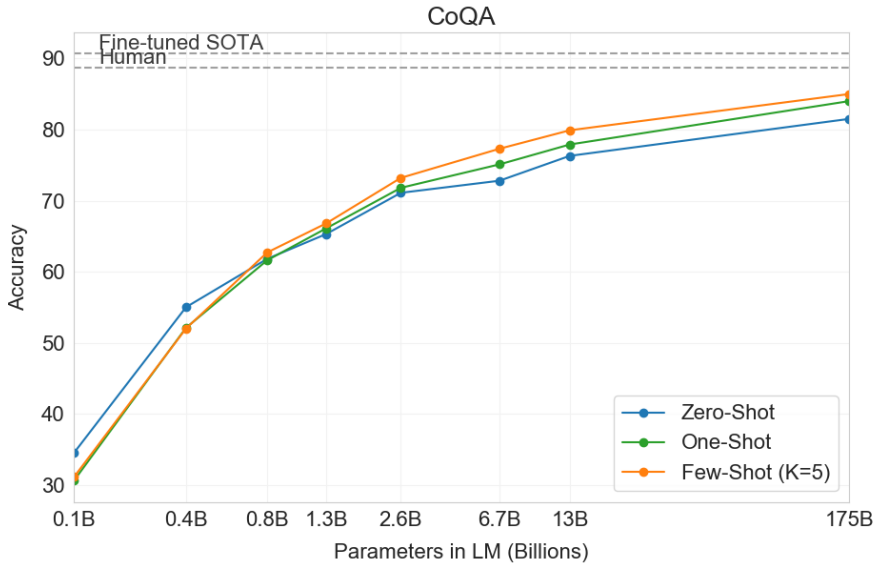

Figure 3.7: GPT-3 results on CoQA reading comprehension task. GPT-3 175B achieves 85 F1 in the few-shot setting, only a few points behind measured human performance and state-of-the-art fine-tuned models. Zero-shot and one-shot performance is a few points behind, with the gains to few-shot being largest for bigger models.

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	**89.0**	**91.0**	**96.9**	**93.9**	**94.8**	**92.5**
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	**76.1**	**93.8**	**62.3**	**88.2**	**92.5**	**93.3**
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

Table 3.8: Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA. All results are reported on the test set. GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

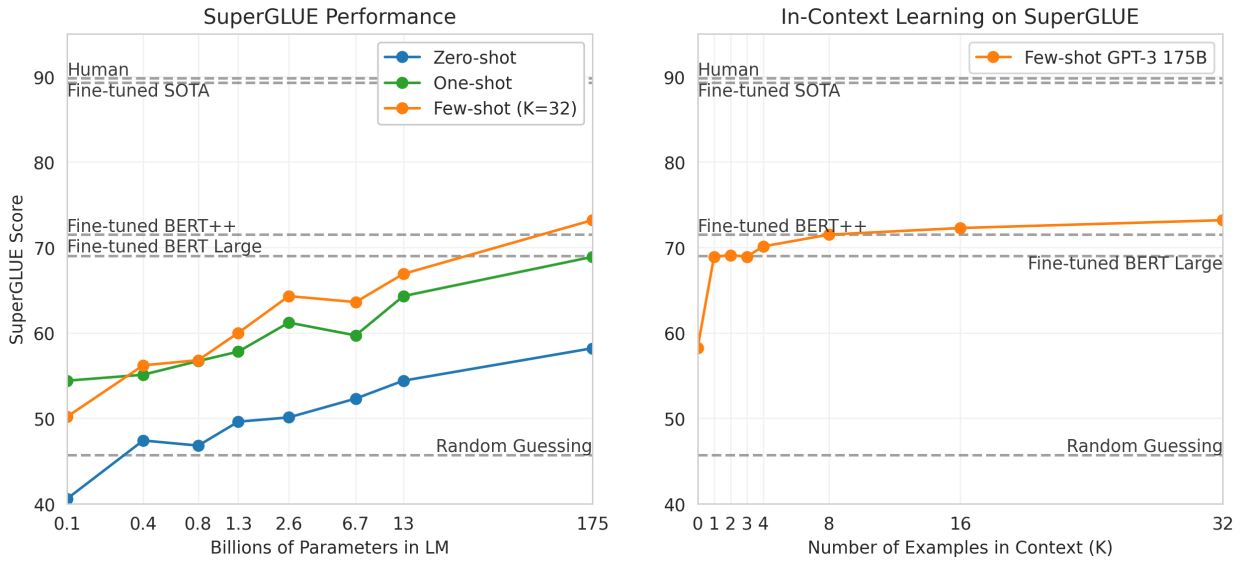

Figure 3.8: Performance on SuperGLUE increases with model size and number of examples in context. A value of $K = 32$ means that our model was shown 32 examples per task, for 256 examples total divided across the 8 tasks in SuperGLUE. We report GPT-3 values on the dev set, so our numbers are not directly comparable to the dotted reference lines (our test set results are in Table 3.8). The BERT-Large reference model was fine-tuned on the SuperGLUE training set (125K examples), whereas BERT++ was first fine-tuned on MultiNLI (392K examples) and SWAG (113K examples) before further fine-tuning on the SuperGLUE training set (for a total of 630K fine-tuning examples). We find the difference in performance between the BERT-Large and BERT++ to be roughly equivalent to the difference between GPT-3 with one example per context versus eight examples per context.

and MultiRC, we sampled a new set of examples to use in the context for each problem. For WSC and MultiRC, we used the same set of randomly drawn examples from the training set as context for all of the problems we evaluated.

We observe a wide range in GPT-3's performance across tasks. On COPA and ReCoRD GPT-3 achieves near-SOTA performance in the one-shot and few-shot settings, with COPA falling only a couple points short and achieving second place on the leaderboard, where first place is held by a fine-tuned 11 billion parameter model (T5). On WSC, performance is still relatively strong, achieving 80.1% in the few-shot setting (note that GPT-3 achieves 88.6% on the original Winograd dataset as described in Section 3.4). On BoolQ, MultiRC, and RTE, performance is reasonable, roughly matching that of a fine-tuned BERT-Large. On CB, we see signs of life at 75.6% in the few-shot setting.

WiC is a notable weak spot with few-shot performance at 49.4% (at random chance). We tried a number of different phrasings and formulations for WiC (which involves determining if a word is being used with the same meaning in two sentences), none of which was able to achieve strong performance. This hints at a phenomenon that will become clearer in the next section (which discusses the ANLI benchmark) – GPT-3 appears to be weak in the few-shot or one-shot setting at some tasks that involve comparing two sentences or snippets, for example whether a word is used the same way in two sentences (WiC), whether one sentence is a paraphrase of another, or whether one sentence implies another. This could also explain the comparatively low scores for RTE and CB, which also follow this format. Despite these weaknesses, GPT-3 still outperforms a fine-tuned BERT-large on four of eight tasks and on two tasks GPT-3 is close to the state-of-the-art held by a fine-tuned 11 billion parameter model.

Finally, we note that the few-shot SuperGLUE score steadily improves with both model size and with number of examples in the context showing increasing benefits from in-context learning (Figure 3.8). We scale K up to 32 examples per task, after which point additional examples will not reliably fit into our context. When sweeping over values of K, we find that GPT-3 requires less than eight total examples per task to outperform a fine-tuned BERT-Large on overall SuperGLUE score.

3.8 NLI

Natural Language Inference (NLI) [Fyo00] concerns the ability to understand the relationship between two sentences. In practice, this task is usually structured as a two or three class classification problem where the model classifies

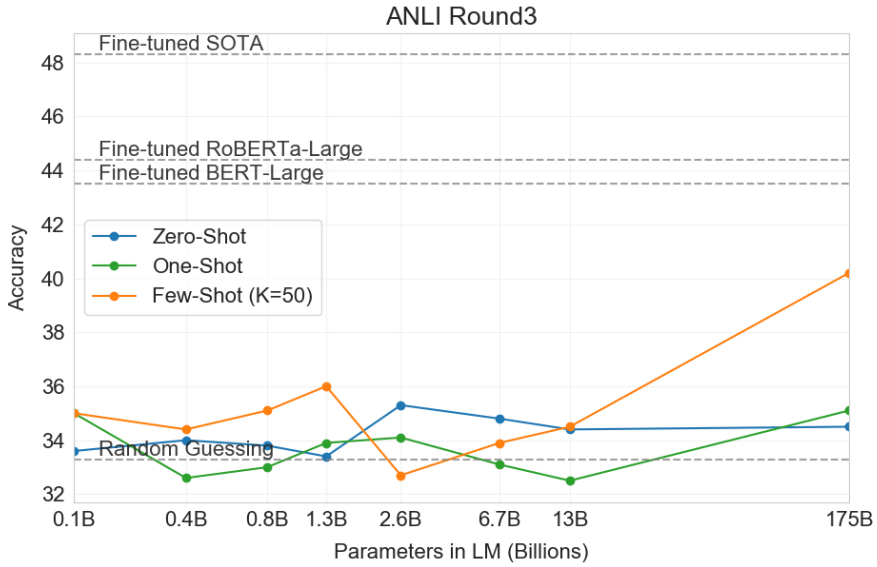

Figure 3.9: Performance of GPT-3 on ANLI Round 3. Results are on the dev-set, which has only 1500 examples and therefore has high variance (we estimate a standard deviation of 1.2%). We find that smaller models hover around random chance, while few-shot GPT-3 175B closes almost half the gap from random chance to SOTA. Results for ANLI rounds 1 and 2 are shown in the appendix.

whether the second sentence logically follows from the first, contradicts the first sentence, or is possibly true (neutral). SuperGLUE includes an NLI dataset, RTE, which evaluates the binary version of the task. On RTE, only the largest version of GPT-3 performs convincingly better than random (56%) in any evaluation setting, but in a few-shot setting GPT-3 performs similarly to a single-task fine-tuned BERT Large. We also evaluate on the recently introduced Adversarial Natural Language Inference (ANLI) dataset [NWD+19]. ANLI is a difficult dataset employing a series of adversarially mined natural language inference questions in three rounds (R1, R2, and R3). Similar to RTE, all of our models smaller than GPT-3 perform at almost exactly random chance on ANLI, even in the few-shot setting ($\sim 33\%$), whereas GPT-3 itself shows signs of life on Round 3. Results for ANLI R3 are highlighted in Figure 3.9 and full results for all rounds can be found in Appendix H. These results on both RTE and ANLI suggest that NLI is still a very difficult task for language models and they are only just beginning to show signs of progress.

3.9 Synthetic and Qualitative Tasks

One way to probe GPT-3's range of abilities in the few-shot (or zero- and one-shot) setting is to give it tasks which require it to perform simple on-the-fly computational reasoning, recognize a novel pattern that is unlikely to have occurred in training, or adapt quickly to an unusual task. We devise several tasks to test this class of abilities. First, we test GPT-3's ability to perform arithmetic. Second, we create several tasks that involve rearranging or unscrambling the letters in a word, tasks which are unlikely to have been exactly seen during training. Third, we test GPT-3's ability to solve SAT-style analogy problems few-shot. Finally, we test GPT-3 on several qualitative tasks, including using new words in a sentence, correcting English grammar, and news article generation. We will release the synthetic datasets with the hope of stimulating further study of test-time behavior of language models.

3.9.1 Arithmetic

To test GPT-3's ability to perform simple arithmetic operations without task-specific training, we developed a small battery of 10 tests that involve asking GPT-3 a simple arithmetic problem in natural language:

- **2 digit addition (2D+)** – The model is asked to add two integers sampled uniformly from $[0, 100)$, phrased in the form of a question, e.g. "Q: What is 48 plus 76? A: 124."
- **2 digit subtraction (2D-)** – The model is asked to subtract two integers sampled uniformly from $[0, 100)$; the answer may be negative. Example: "Q: What is 34 minus 53? A: -19".
- **3 digit addition (3D+)** – Same as 2 digit addition, except numbers are uniformly sampled from $[0, 1000)$.

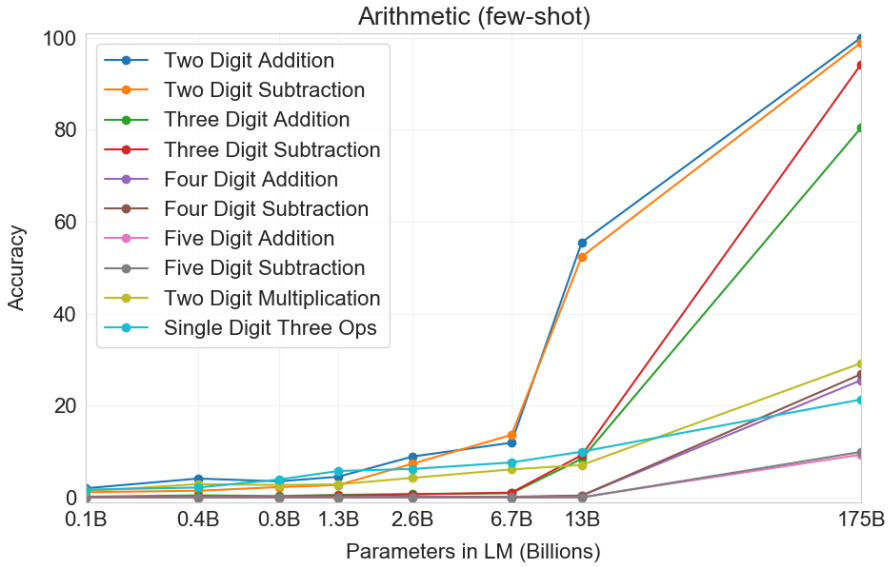

Figure 3.10: Results on all 10 arithmetic tasks in the few-shot settings for models of different sizes. There is a significant jump from the second largest model (GPT-3 13B) to the largest model (GPT-3 175), with the latter being able to reliably accurate 2 digit arithmetic, usually accurate 3 digit arithmetic, and correct answers a significant fraction of the time on 4-5 digit arithmetic, 2 digit multiplication, and compound operations. Results for one-shot and zero-shot are shown in the appendix.

- **3 digit subtraction (3D-)** – Same as 2 digit subtraction, except numbers are uniformly sampled from $[0, 1000)$.
- **4 digit addition (4D+)** – Same as 3 digit addition, except uniformly sampled from $[0, 10000)$.
- **4 digit subtraction (4D-)** – Same as 3 digit subtraction, except uniformly sampled from $[0, 10000)$.
- **5 digit addition (5D+)** – Same as 3 digit addition, except uniformly sampled from $[0, 100000)$.
- **5 digit subtraction (5D-)** – Same as 3 digit subtraction, except uniformly sampled from $[0, 100000)$.
- **2 digit multiplication (2Dx)** – The model is asked to multiply two integers sampled uniformly from $[0, 100)$, e.g. "Q: What is 24 times 42? A: 1008".
- **One-digit composite (1DC)** – The model is asked to perform a composite operation on three 1 digit numbers, with parentheses around the last two. For example, "Q: What is 6+(4*8)? A: 38". The three 1 digit numbers are selected uniformly on $[0, 10)$ and the operations are selected uniformly from {+,-,*}.

In all 10 tasks the model must generate the correct answer exactly. For each task we generate a dataset of 2,000 random instances of the task and evaluate all models on those instances.

First we evaluate GPT-3 in the few-shot setting, for which results are shown in Figure 3.10. On addition and subtraction, GPT-3 displays strong proficiency when the number of digits is small, achieving 100% accuracy on 2 digit addition, 98.9% at 2 digit subtraction, 80.2% at 3 digit addition, and 94.2% at 3-digit subtraction. Performance decreases as the number of digits increases, but GPT-3 still achieves 25-26% accuracy on four digit operations and 9-10% accuracy on five digit operations, suggesting at least some capacity to generalize to larger numbers of digits. GPT-3 also achieves 29.2% accuracy at 2 digit multiplication, an especially computationally intensive operation. Finally, GPT-3 achieves 21.3% accuracy at single digit combined operations (for example, 9*(7+5)), suggesting that it has some robustness beyond just single operations.

As Figure 3.10 makes clear, small models do poorly on all of these tasks – even the 13 billion parameter model (the second largest after the 175 billion full GPT-3) can solve 2 digit addition and subtraction only half the time, and all other operations less than 10% of the time.

One-shot and zero-shot performance are somewhat degraded relative to few-shot performance, suggesting that adaptation to the task (or at the very least recognition of the task) is important to performing these computations correctly. Nevertheless, one-shot performance is still quite strong, and even zero-shot performance of the full GPT-3 significantly

Setting	2D+	2D-	3D+	3D-	4D+	4D-	5D+	5D-	2Dx	1DC
GPT-3 Zero-shot	76.9	58.0	34.2	48.3	4.0	7.5	0.7	0.8	19.8	9.8
GPT-3 One-shot	99.6	86.4	65.5	78.7	14.0	14.0	3.5	3.8	27.4	14.3
GPT-3 Few-shot	100.0	98.9	80.4	94.2	25.5	26.8	9.3	9.9	29.2	21.3

Table 3.9: Results on basic arithmetic tasks for GPT-3 175B. {2,3,4,5}D{+,-} is 2, 3, 4, and 5 digit addition or subtraction, 2Dx is 2 digit multiplication. 1DC is 1 digit composite operations. Results become progressively stronger moving from the zero-shot to one-shot to few-shot setting, but even the zero-shot shows significant arithmetic abilities.

Setting	CL	A1	A2	RI	RW
GPT-3 Zero-shot	3.66	2.28	8.91	8.26	0.09
GPT-3 One-shot	21.7	8.62	25.9	45.4	0.48
GPT-3 Few-shot	37.9	15.1	39.7	67.2	0.44

Table 3.10: GPT-3 175B performance on various word unscrambling and word manipulation tasks, in zero-, one-, and few-shot settings. CL is "cycle letters in word", A1 is anagrams of but the first and last letters, A2 is anagrams of all but the first and last two letters, RI is "Random insertion in word", RW is "reversed words".

outperforms few-shot learning for all smaller models. All three settings for the full GPT-3 are shown in Table 3.9, and model capacity scaling for all three settings is shown in Appendix H.

To spot-check whether the model is simply memorizing specific arithmetic problems, we took the 3-digit arithmetic problems in our test set and searched for them in our training data in both the forms `"<NUM1> + <NUM2> ="` and `"<NUM1> plus <NUM2>"`. Out of 2,000 addition problems we found only 17 matches (0.8%) and out of 2,000 subtraction problems we found only 2 matches (0.1%), suggesting that only a trivial fraction of the correct answers could have been memorized. In addition, inspection of incorrect answers reveals that the model often makes mistakes such as not carrying a "1", suggesting it is actually attempting to perform the relevant computation rather than memorizing a table.

Overall, GPT-3 displays reasonable proficiency at moderately complex arithmetic in few-shot, one-shot, and even zero-shot settings.

3.9.2 Word Scrambling and Manipulation Tasks

To test GPT-3's ability to learn novel symbolic manipulations from a few examples, we designed a small battery of 5 "character manipulation" tasks. Each task involves giving the model a word distorted by some combination of scrambling, addition, or deletion of characters, and asking it to recover the original word. The 5 tasks are:

- **Cycle letters in word (CL)** – The model is given a word with its letters cycled, then the "=" symbol, and is expected to generate the original word. For example, it might be given "lyinevitab" and should output "inevitably".
- **Anagrams of all but first and last characters (A1)** – The model is given a word where every letter except the first and last have been scrambled randomly, and must output the original word. Example: criroptuon = corruption.
- **Anagrams of all but first and last 2 characters (A2)** – The model is given a word where every letter except the first 2 and last 2 have been scrambled randomly, and must recover the original word. Example: opoepnnt $\rightarrow$ opponent.
- **Random insertion in word (RI)** – A random punctuation or space character is inserted between each letter of a word, and the model must output the original word. Example: s.u!c/c!e.s s i/o/n = succession.
- **Reversed words (RW)** – The model is given a word spelled backwards, and must output the original word. Example: stcejbo $\rightarrow$ objects.

For each task we generate 10,000 examples, which we chose to be the top 10,000 most frequent words as measured by [Nor09] of length more than 4 characters and less than 15 characters. The few-shot results are shown in Figure 3.11. Task performance tends to grow smoothly with model size, with the full GPT-3 model achieving 66.9% on removing

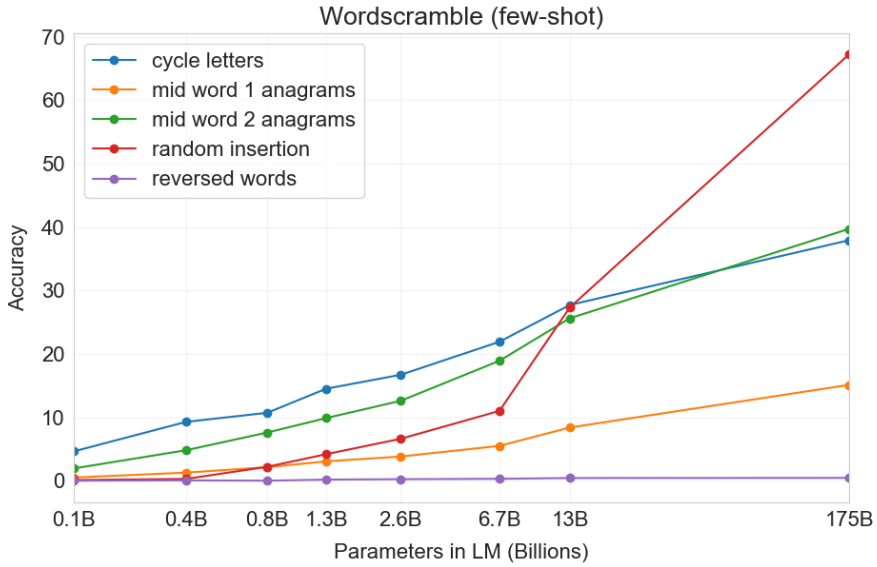

Figure 3.11: Few-shot performance on the five word scrambling tasks for different sizes of model. There is generally smooth improvement with model size although the random insertion task shows an upward slope of improvement with the 175B model solving the task the majority of the time. Scaling of one-shot and zero-shot performance is shown in the appendix. All tasks are done with $K = 100$.

random insertions, 38.6% on cycling letters, 40.2% on the easier anagram task, and 15.1% on the more difficult anagram task (where only the first and last letters are held fixed). None of the models can reverse the letters in a word.

In the one-shot setting, performance is significantly weaker (dropping by half or more), and in the zero-shot setting the model can rarely perform any of the tasks (Table 3.10). This suggests that the model really does appear to learn these tasks at test time, as the model cannot perform them zero-shot and their artificial nature makes them unlikely to appear in the pre-training data (although we cannot confirm this with certainty).

We can further quantify performance by plotting "in-context learning curves", which show task performance as a function of the number of in-context examples. We show in-context learning curves for the Symbol Insertion task in Figure 1.2. We can see that larger models are able to make increasingly effective use of in-context information, including both task examples and natural language task descriptions.

Finally, it is worth adding that solving these tasks requires character-level manipulations, whereas our BPE encoding operates on significant fractions of a word (on average ~ 0.7 words per token), so from the LM's perspective succeeding at these tasks involves not just manipulating BPE tokens but understanding and pulling apart their substructure. Also, CL, A1, and A2 are not bijective (that is, the unscrambled word is not a deterministic function of the scrambled word), requiring the model to perform some search to find the correct unscrambling. Thus, the skills involved appear to require non-trivial pattern-matching and computation.

3.9.3 SAT Analogies

To test GPT-3 on another task that is somewhat unusual relative to the typical distribution of text, we collected a set of 374 "SAT analogy" problems [TLBS03]. Analogies are a style of multiple choice question that constituted a section of the SAT college entrance exam before 2005. A typical example is "audacious is to boldness as (a) sanctimonious is to hypocrisy, (b) anonymous is to identity, (c) remorseful is to misdeed, (d) deleterious is to result, (e) impressionable is to temptation". The student is expected to choose which of the five word pairs has the same relationship as the original word pair; in this example the answer is "sanctimonious is to hypocrisy". On this task GPT-3 achieves 65.2% in the few-shot setting, 59.1% in the one-shot setting, and 53.7% in the zero-shot setting, whereas the average score among college applicants was 57% [TL05] (random guessing yields 20%). As shown in Figure 3.12, the results improve with scale, with the the full 175 billion model improving by over 10% compared to the 13 billion parameter model.

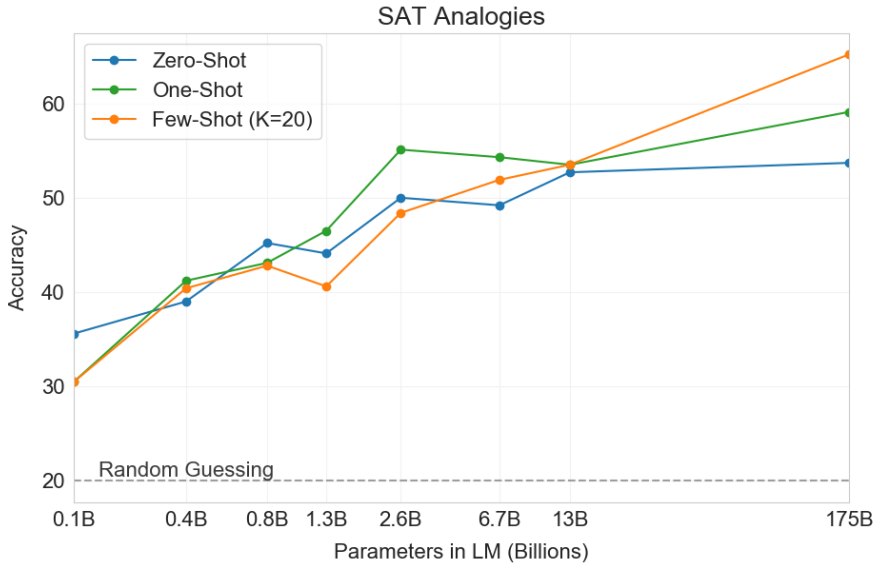

Figure 3.12: Zero-, one-,and few-shot performance on SAT analogy tasks, for different sizes of model. The largest model achieves 65% accuracy in the few-shot setting, and also demonstrates significant gains to in-context learning which are not present in smaller models.

3.9.4 News Article Generation

Previous work on generative language models qualitatively tested their ability to generate synthetic "news articles" by conditional sampling from the model given a human-written prompt consisting of a plausible first sentence for a news story [RWC+19]. Relative to [RWC+19], the dataset used to train GPT-3 is much less weighted towards news articles, so trying to generate news articles via raw unconditional samples is less effective – for example GPT-3 often interprets the proposed first sentence of a "news article" as a tweet and then posts synthetic responses or follow-up tweets. To solve this problem we employed GPT-3's few-shot learning abilities by providing three previous news articles in the model's context to condition it. With the title and subtitle of a proposed next article, the model is able to reliably generate short articles in the "news" genre.

To gauge the quality of news article generation from GPT-3 (which we believe is likely to be correlated with conditional sample generation quality in general), we decided to measure human ability to distinguish GPT-3-generated articles from real ones. Similar work has been carried out by Kreps et al. [KMB20] and Zellers et al. [ZHR+19]. Generative language models are trained to match the distribution of content generated by humans, so the (in)ability of humans to distinguish the two is a potentially important measure of quality.[3]

In order to see how well humans can detect model generated text, we arbitrarily selected 25 article titles and subtitles from the website newser.com (mean length: 215 words). We then generated completions of these titles and subtitles from four language models ranging in size from 125M to 175B (GPT-3) parameters (mean length: 200 words). For each model, we presented around 80 US-based participants with a quiz consisting of these real titles and subtitles followed by either the human written article or the article generated by the model[4]. Participants were asked to select whether the article was "very likely written by a human", "more likely written by a human", "I don't know", "more likely written by a machine", or "very likely written by a machine".

The articles we selected were not in the models' training data and the model outputs were formatted and selected programmatically to prevent human cherry-picking. All models used the same context to condition outputs on and were pre-trained with the same context size and the same article titles and subtitles were used as prompts for each model. However, we also ran an experiment to control for participant effort and attention that followed the same format but involved intentionally bad model generated articles. This was done by generating articles from a "control model": a 160M parameter model with no context and increased output randomness.

[3] This task is also relevant to the potential misuse of language models discussed in Section 6.1.

[4] We wanted to identify how good an average person on the internet is at detecting language model outputs, so we focused on participants drawn from the general US population. See Appendix E for details.

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p-value)	"I don't know" assignments
Control (deliberately bad model)	86%	83%–90%	-	3.6 %
GPT-3 Small	76%	72%–80%	3.9 (2e-4)	4.9%
GPT-3 Medium	61%	58%–65%	10.3 (7e-21)	6.0%
GPT-3 Large	68%	64%–72%	7.3 (3e-11)	8.7%
GPT-3 XL	62%	59%–65%	10.7 (1e-19)	7.5%
GPT-3 2.7B	62%	58%–65%	10.4 (5e-19)	7.1%
GPT-3 6.7B	60%	56%–63%	11.2 (3e-21)	6.2%
GPT-3 13B	55%	52%–58%	15.3 (1e-32)	7.1%
GPT-3 175B	52%	49%–54%	16.9 (1e-34)	7.8%

Table 3.11: Human accuracy in identifying whether short (∼200 word) news articles are model generated. We find that human accuracy (measured by the ratio of correct assignments to non-neutral assignments) ranges from 86% on the control model to 52% on GPT-3 175B. This table compares mean accuracy between five different models, and shows the results of a two-sample T-Test for the difference in mean accuracy between each model and the control model (an unconditional GPT-3 Small model with increased output randomness).

Mean human accuracy (the ratio of correct assignments to non-neutral assignments per participant) at detecting that the intentionally bad articles were model generated was $\sim 86\%$ where 50% is chance level performance. By contrast, mean human accuracy at detecting articles that were produced by the 175B parameter model was barely above chance at $\sim 52\%$ (see Table 3.11).[5] Human abilities to detect model generated text appear to decrease as model size increases: there appears to be a trend towards chance accuracy with model size, and human detection of GPT-3 is close to chance.[6] This is true despite the fact that participants spend more time on each output as model size increases (see Appendix E).

Examples of synthetic articles from GPT-3 are given in Figures 3.14 and 3.15.[7] Much of the text is—as indicated by the evaluations—difficult for humans to distinguish from authentic human content. Factual inaccuracies can be an indicator that an article is model generated since, unlike human authors, the models have no access to the specific facts that the article titles refer to or when the article was written. Other indicators include repetition, non sequiturs, and unusual phrasings, though these are often subtle enough that they are not noticed.

Related work on language model detection by Ippolito et al. [IDCBE19] indicates that automatic discriminators like GROVER [ZHR+19] and GLTR [GSR19] may have greater success at detecting model generated text than human evaluators. Automatic detection of these models may be a promising area of future research.

Ippolito et al. [IDCBE19] also note that human accuracy at detecting model generated text increases as humans observe more tokens. To do a preliminary investigation of how good humans are at detecting longer news articles generated by GPT-3 175B, we selected 12 world news articles from Reuters with an average length of 569 words and generated completions of these articles from GPT-3 with an average length of 498 words (298 words longer than our initial experiments). Following the methodology above, we ran two experiments, each on around 80 US-based participants, to compare human abilities to detect the articles generated by GPT-3 and a control model.

We found that mean human accuracy at detecting the intentionally bad longer articles from the control model was $\sim 88\%$, while mean human accuracy at detecting the longer articles that were produced by GPT-3 175B was still barely above chance at $\sim 52\%$ (see Table 3.12). This indicates that, for news articles that are around 500 words long, GPT-3 continues to produce articles that humans find difficult to distinguish from human written news articles.

3.9.5 Learning and Using Novel Words

A task studied in developmental linguistics [CB78] is the ability to learn and utilize new words, for example using a word in a sentence after seeing it defined only once, or conversely inferring a word's meaning from only one usage. Here we qualitatively test GPT-3's ability to do the former. Specifically, we give GPT-3 the definition of a nonexistent word, such as "Gigamuru", and then ask it to use it in a sentence. We provide one to five previous examples of a (separate)

[5] We use a two-sample Student's T-Test to test for significant difference between the means of the participant accuracies of each model and the control model and report the normalized difference in the means (as the t-statistic) and the p-value.

[6] If a model consistently produces texts that are more impressive than human articles, it is possible that human performance on this task would drop below 50%. Indeed, many individual participants scored below 50% on this task.

[7] Additional non-news samples can be found in Appendix F.

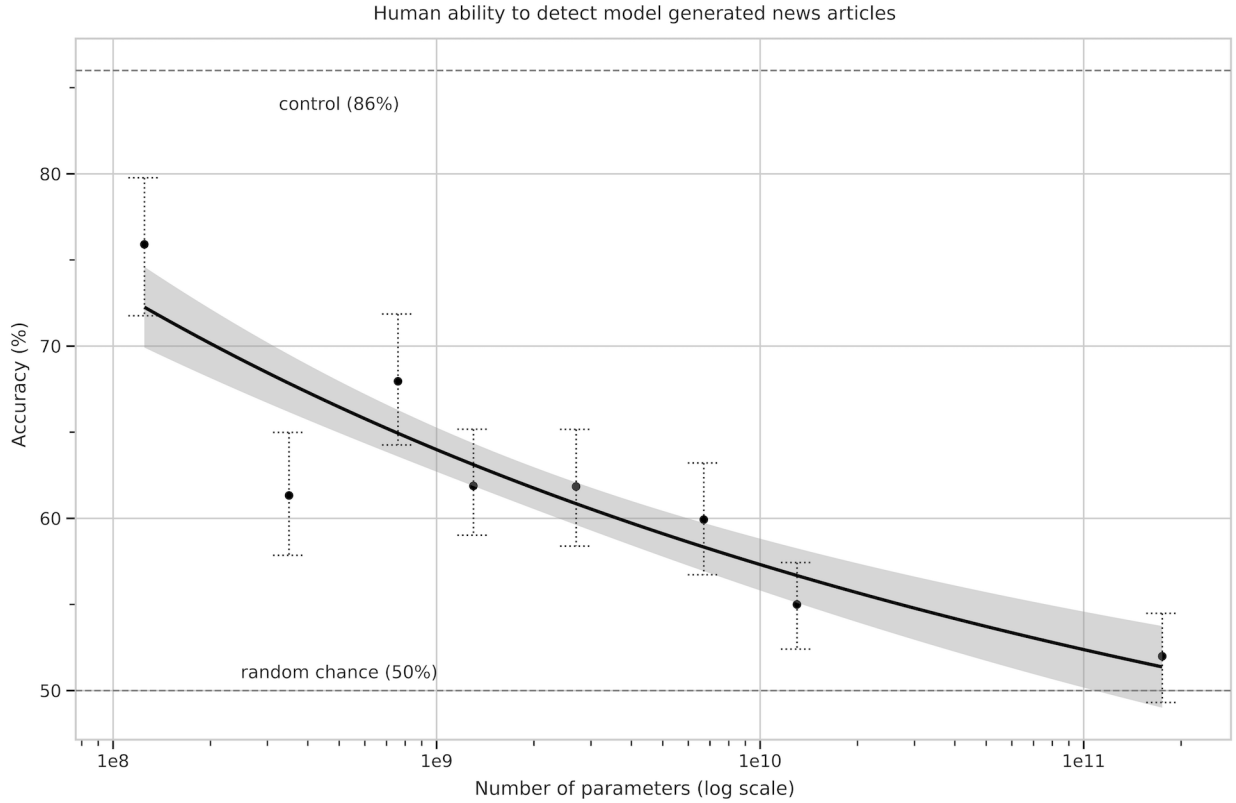

Figure 3.13: People's ability to identify whether news articles are model-generated (measured by the ratio of correct assignments to non-neutral assignments) decreases as model size increases. Accuracy on the outputs on the deliberately-bad control model (an unconditioned GPT-3 Small model with higher output randomness) is indicated with the dashed line at the top, and the random chance (50%) is indicated with the dashed line at the bottom. Line of best fit is a power law with 95% confidence intervals.

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p-value)	"I don't know" assignments
Control	88%	84%–91%	-	2.7%
GPT-3 175B	52%	48%–57%	12.7 (3.2e-23)	10.6%

Table 3.12: People's ability to identify whether ~ 500 word articles are model generated (as measured by the ratio of correct assignments to non-neutral assignments) was 88% on the control model and 52% on GPT-3 175B. This table shows the results of a two-sample T-Test for the difference in mean accuracy between GPT-3 175B and the control model (an unconditional GPT-3 Small model with increased output randomness).

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Figure 3.14: The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%).

Title: Star's Tux Promise Draws Megyn Kelly's Sarcasm
Subtitle: Joaquin Phoenix pledged to not change for each awards event
Article: A year ago, Joaquin Phoenix made headlines when he appeared on the red carpet at the Golden Globes wearing a tuxedo with a paper bag over his head that read, "I am a shape-shifter. I can't change the world. I can only change myself." It was a promise to not change to fit into the Hollywood mold: "I think that's a really special thing, to not change yourself. I think it's a really special thing to say, 'This is what's inside of me, I'm proud of it, and I'm not going to be ashamed because of the way that someone else thinks I should be.'" Now, it's the Oscars, and Phoenix is at it again. But this time, his publicist is saying he'll be wearing a tux no matter what.
Megyn Kelly was not impressed, and she let him have it on The Tonight Show. "You know, I feel like, I feel like you could have worn the tux," she says. "But you're saying you're a shape-shifter. I don't know if you can change your tux, but you can change your mind. You can change your mind. You can change your mind." Phoenix says he did, but it didn't stick. "I was like, 'Okay, I'm going to wear a tuxedo to this thing.' And then I thought, 'I don't want to wear a tuxedo to this thing.'" Kelly goes on to encourage him to change his mind again, but Phoenix says it's too late: "I'm committed to wearing this."

Figure 3.15: The GPT-3 generated news article that humans found the easiest to distinguish from a human written article (accuracy: 61%).

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:
One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduddles.
A "yalubalu" is a type of vegetable that looks like a big pumpkin. An example of a sentence that uses the word yalubalu is:
I was on a trip to Africa and I tried this yalubalu vegetable that was grown in a garden there. It was delicious.

A "Burringo" is a car with very fast acceleration. An example of a sentence that uses the word Burringo is:
In our garage we have a Burringo that my father drives to work every day.

A "Gigamuru" is a type of Japanese musical instrument. An example of a sentence that uses the word Gigamuru is:
I have a Gigamuru that my uncle gave me as a gift. I love to play it at home.

To "screeg" something is to swing a sword at it. An example of a sentence that uses the word screeg is:
We screeghed at each other for several minutes and then we went outside and ate ice cream.

Figure 3.16: Representative GPT-3 completions for the few-shot task of using a new word in a sentence. Boldface is GPT-3's completions, plain text is human prompts. In the first example both the prompt and the completion are provided by a human; this then serves as conditioning for subsequent examples where GPT-3 receives successive additional prompts and provides the completions. Nothing task-specific is provided to GPT-3 other than the conditioning shown here.

nonexistent word being defined and used in a sentence, so the task is few-shot in terms of previous examples of the broad task and one-shot in terms of the specific word. Table 3.16 shows the 6 examples we generated; all definitions were human-generated, and the first answer was human-generated as conditioning while the subsequent answers were generated by GPT-3. These examples were generated continuously in one sitting and we did not omit or repeatedly try any prompts. In all cases the generated sentence appears to be a correct or at least plausible use of the word. In the final sentence the model generates a plausible conjugation for the word "screeg" (namely "screeghed"), although the use of the word is slightly awkward ("screeghed at each other") despite being plausible in the sense that it could describe a toy sword fight. Overall, GPT-3 appears to be at least proficient at the task of using novel words in a sentence.

3.9.6 Correcting English Grammar

Another task well suited for few-shot learning is correcting English grammar. We test this with GPT-3 in the few-shot setting by giving prompts of the form `"Poor English Input: <sentence>\n Good English Output: <sentence>"`. We give GPT-3 one human-generated correction and then ask it to correct 5 more (again without any omissions or repeats). Results are shown in Figure 3.17.

4 Measuring and Preventing Memorization Of Benchmarks

Since our training dataset is sourced from the internet, it is possible that our model was trained on some of our benchmark test sets. Accurately detecting test contamination from internet-scale datasets is a new area of research without established best practices. While it is common practice to train large models without investigating contamination, given the increasing scale of pretraining datasets, we believe this issue is becoming increasingly important to attend to.

This concern is not just hypothetical. One of the first papers to train a language model on Common Crawl data [TL18] detected and removed a training document which overlapped with one of their evaluation datasets. Other work such as GPT-2 [RWC+19] also conducted post-hoc overlap analysis. Their study was relatively encouraging, finding that

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.
Poor English input: I'd be more than happy to work with you in another project.
Good English output: I'd be more than happy to work with you on another project.

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.
Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.

Poor English input: The patient was died.
Good English output: The patient died.

Poor English input: We think that Leslie likes ourselves.
Good English output: We think that Leslie likes us.

Poor English input: Janet broke Bill on the finger.
Good English output: Janet broke Bill's finger.

Poor English input: Mary arranged for, in St. Louis, John to rent a house cheap.
Good English output: Mary arranged for John to rent a house in St. Louis.

Poor English input: Today I have went to the store to to buys some many bottle of water.
Good English output: Today I went to the store to buy some bottles of water.

Poor English input: I have tried to hit ball with bat, but my swing is has miss.
Good English output: I tried to hit the ball with the bat, but my swing missed.

Figure 3.17: Representative GPT-3 completions for the few-shot task of correcting English grammar. Boldface is GPT-3's completions, plain text is human prompts. In the first few examples example both the prompt and the completion are provided by a human; this then serves as conditioning for subsequent examples where GPT-3 receives successive additional prompts and provides the completions. Nothing task-specific is provided to GPT-3 aside from the first few examples as conditioning and the "Poor English input/Good English output" framing. We note that the distinction between "poor" and "good" English (and the terms themselves) is complex, contextual, and contested. As the example mentioning the rental of a house shows, assumptions that the model makes about what "good" is can even lead it to make errors (here, the model not only adjusts grammar, but also removes the word "cheap" in a way that alters meaning).

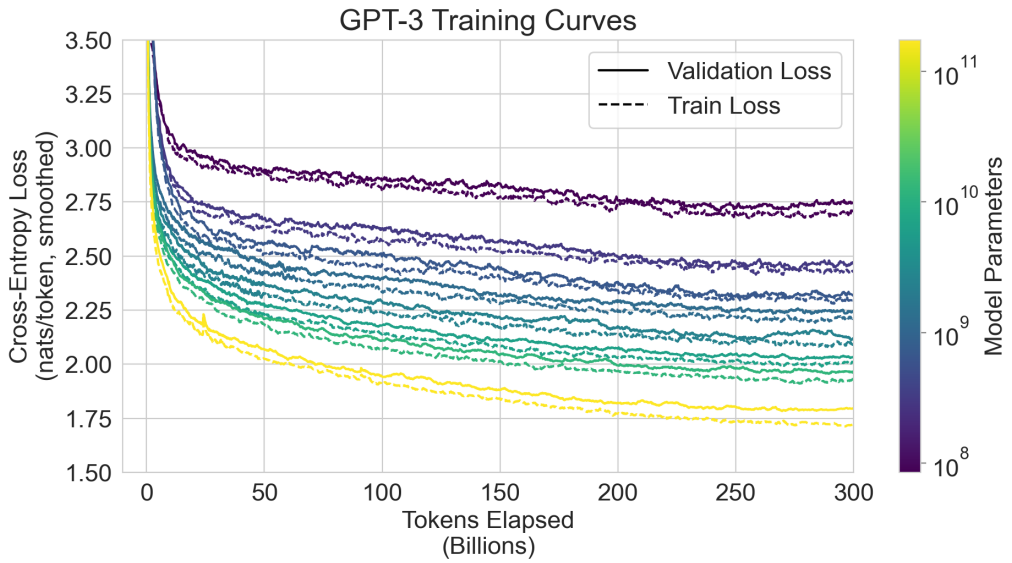

Figure 4.1: GPT-3 Training Curves We measure model performance during training on a deduplicated validation split of our training distribution. Though there is some gap between training and validation performance, the gap grows only minimally with model size and training time, suggesting that most of the gap comes from a difference in difficulty rather than overfitting.

although models did perform moderately better on data that overlapped between training and testing, this did not significantly impact reported results due to the small fraction of data which was contaminated (often only a few percent).

GPT-3 operates in a somewhat different regime. On the one hand, the dataset and model size are about two orders of magnitude larger than those used for GPT-2, and include a large amount of Common Crawl, creating increased potential for contamination and memorization. On the other hand, precisely due to the large amount of data, even GPT-3 175B does not overfit its training set by a significant amount, measured relative to a held-out validation set with which it was deduplicated (Figure 4.1). Thus, we expect that contamination is likely to be frequent, but that its effects may not be as large as feared.

We initially tried to address the issue of contamination by proactively searching for and attempting to remove any overlap between our training data and the development and test sets of all benchmarks studied in this paper. Unfortunately, a bug resulted in only partial removal of all detected overlaps from the training data. Due to the cost of training, it wasn't feasible to retrain the model. To address this, we investigate in detail how the remaining detected overlap impacts results.

For each benchmark, we produce a 'clean' version which removes all potentially leaked examples, defined roughly as examples that have a 13-gram overlap with anything in the pretraining set (or that overlap with the whole example when it is shorter than 13-grams). The goal is to very conservatively flag anything that could potentially be contamination, so as to produce a clean subset that is free of contamination with high confidence. The exact procedure is detailed in Appendix C.

We then evaluate GPT-3 on these clean benchmarks, and compare to the original score. If the score on the clean subset is similar to the score on the entire dataset, this suggests that contamination, even if present, does not have a significant effect on reported results. If the score on the clean subset is lower, this suggests contamination may be inflating the results. The results are summarized in Figure 4.2. Although potential contamination is often high (with a quarter of benchmarks scoring over 50%), in most cases performance changes only negligibly, and we see no evidence that contamination level and performance difference are correlated. We conclude that either our conservative method substantially overestimated contamination or that contamination has little effect on performance.

Below, we review in more detail the few specific cases where either (1) the model performs significantly worse on the cleaned version, or (2) potential contamination is very high, which makes measuring the performance difference difficult.

Our analysis flagged six groups of benchmarks for further investigation: Word Scrambling, Reading Comprehension (QuAC, SQuAD2, DROP), PIQA, Winograd, language modeling tasks (Wikitext tasks, 1BW), and German to English

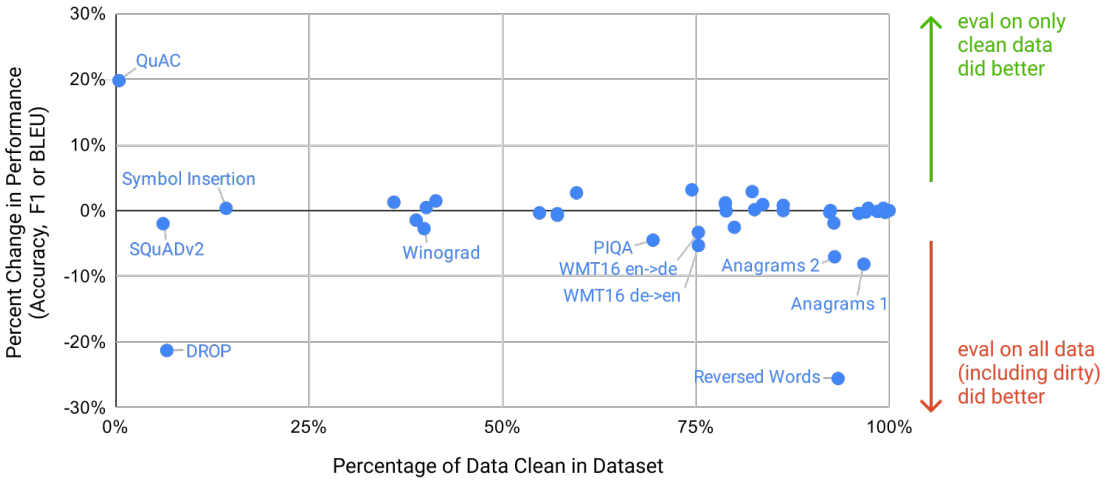

Figure 4.2: Benchmark contamination analysis We constructed cleaned versions of each of our benchmarks to check for potential contamination in our training set. The x-axis is a conservative lower bound for how much of the dataset is known with high confidence to be clean, and the y-axis shows the difference in performance when evaluating only on the verified clean subset. Performance on most benchmarks changed negligibly, but some were flagged for further review. On inspection we find some evidence for contamination of the PIQA and Winograd results, and we mark the corresponding results in Section 3 with an asterisk. We find no evidence that other benchmarks are affected.

translation. Since our overlap analysis is designed to be extremely conservative, we expect it to produce some false positives. We summarize the results for each group of tasks below:

- **Reading Comprehension:** Our initial analysis flagged >90% of task examples from QuAC, SQuAD2, and DROP as potentially contaminated, so large that even measuring the differential on a clean subset was difficult. Upon manual inspection, however, we found that for every overlap we inspected, in all 3 datasets, the source text was present in our training data but the question/answer pairs were not, meaning the model gains only background information and cannot memorize the answer to a specific question.
- **German translation:** We found 25% of the examples in the WMT16 German-English test set were marked as potentially contaminated, with an associated total effect size of 1-2 BLEU. Upon inspection, none of the flagged examples contain paired sentences resembling NMT training data and collisions were monolingual matches mostly of snippets of events discussed in the news.
- **Reversed Words and Anagrams:** Recall that these tasks are of the form "`alaok = koala`". Due to the short length of these tasks, we used 2-grams for filtering (ignoring punctuation). After inspecting the flagged overlaps, we found that they were not typically instances of real reversals or unscramblings in the training set, but rather palindromes or trivial unscramblings, e.g "`kayak = kayak`". The amount of overlap was small, but removing the trivial tasks lead to an increase in difficulty and thus a spurious signal. Related to this, the symbol insertion task shows high overlap but no effect on performance – this is because that task involves removing non-letter characters from a word, and the overlap analysis itself ignores such characters, leading to many spurious matches.
- **PIQA:** The overlap analysis flagged 29% of examples as contaminated, and observed a 3 percentage point absolute decrease (4% relative decrease) in performance on the clean subset. Though the test dataset was released after our training set was created and its labels are hidden, some of the web pages used by the crowdsourced dataset creators are contained in our training set. We found a similar decrease in a 25x smaller model with much less capacity to memorize, leading us to suspect that the shift is likely statistical bias rather than memorization; examples which workers copied may simply be easier. Unfortunately, we cannot rigorously prove this hypothesis. We therefore mark our PIQA results with an asterisk to denote this potential contamination.
- **Winograd:** The overlap analysis flagged 45% of examples, and found a 2.6% decrease in performance on the clean subset. Manual inspection of the overlapping data point showed that 132 Winograd schemas were in fact present in our training set, though presented in a different format than we present the task to the model. Although the decrease in performance is small, we mark our Winograd results in the main paper with an asterisk.

- **Language modeling:** We found the 4 Wikipedia language modeling benchmarks measured in GPT-2, plus the Children's Book Test dataset, to be almost entirely contained in our training data. Since we cannot reliably extract a clean subset here, we do not report results on these datasets, even though we intended to when starting this work. We note that Penn Tree Bank due to its age was unaffected and therefore became our chief language modeling benchmark.

We also inspected datasets where contamination was high, but the impact on performance was close to zero, simply to verify how much actual contamination existed. These appeared to often contain false positives. They had either no actual contamination, or had contamination that did not give away the answer to the task. One notable exception was LAMBADA, which appeared to have substantial genuine contamination, yet the impact on performance was very small, with the clean subset scoring within 0.5% of the full dataset. Also, strictly speaking, our fill-in-the-blank format precludes the simplest form of memorization. Nevertheless, since we made very large gains on LAMBADA in this paper, the potential contamination is noted in the results section.

An important limitation of our contamination analysis is that we cannot be sure that the clean subset is drawn from the same distribution as the original dataset. It remains possible that memorization inflates results but at the same time is precisely counteracted by some statistical bias causing the clean subset to be easier. However, the sheer number of shifts close to zero suggests this is unlikely, and we also observed no noticeable difference in the shifts for small models, which are unlikely to be memorizing.

Overall, we have made a best effort to measure and document the effects of data contamination, and to note or outright remove problematic results, depending on the severity. Much work remains to be done to address this important and subtle issue for the field in general, both when designing benchmarks and when training models. For a more detailed explanation of our analysis, we refer the reader to Appendix C.

5 Limitations

GPT-3 and our analysis of it have a number of limitations. Below we describe some of these and suggest directions for future work.

First, despite the strong quantitative and qualitative improvements of GPT-3, particularly compared to its direct predecessor GPT-2, it still has notable weaknesses in text synthesis and several NLP tasks. On text synthesis, although the overall quality is high, GPT-3 samples still sometimes repeat themselves semantically at the document level, start to lose coherence over sufficiently long passages, contradict themselves, and occasionally contain non-sequitur sentences or paragraphs. We will release a collection of 500 uncurated unconditional samples to help provide a better sense of GPT-3's limitations and strengths at text synthesis. Within the domain of discrete language tasks, we have noticed informally that GPT-3 seems to have special difficulty with "common sense physics", despite doing well on some datasets (such as PIQA [BZB+19]) that test this domain. Specifically GPT-3 has difficulty with questions of the type "If I put cheese into the fridge, will it melt?". Quantitatively, GPT-3's in-context learning performance has some notable gaps on our suite of benchmarks, as described in Section 3, and in particular it does little better than chance when evaluated one-shot or even few-shot on some "comparison" tasks, such as determining if two words are used the same way in a sentence, or if one sentence implies another (WIC and ANLI respectively), as well as on a subset of reading comprehension tasks. This is especially striking given GPT-3's strong few-shot performance on many other tasks.

GPT-3 has several structural and algorithmic limitations, which could account for some of the issues above. We focused on exploring in-context learning behavior in autoregressive language models because it is straightforward to both sample and compute likelihoods with this model class. As a result our experiments do not include any bidirectional architectures or other training objectives such as denoising. This is a noticeable difference from much of the recent literature, which has documented improved fine-tuning performance when using these approaches over standard language models [RSR+19]. Thus our design decision comes at the cost of potentially worse performance on tasks which empirically benefit from bidirectionality. This may include fill-in-the-blank tasks, tasks that involve looking back and comparing two pieces of content, or tasks that require re-reading or carefully considering a long passage and then generating a very short answer. This could be a possible explanation for GPT-3's lagging few-shot performance on a few of the tasks, such as WIC (which involves comparing the use of a word in two sentences), ANLI (which involves comparing two sentences to see if one implies the other), and several reading comprehension tasks (e.g. QuAC and RACE). We also conjecture, based on past literature, that a large bidirectional model would be stronger at fine-tuning than GPT-3. Making a bidirectional model at the scale of GPT-3, and/or trying to make bidirectional models work with few- or zero-shot learning, is a promising direction for future research, and could help achieve the "best of both worlds".

A more fundamental limitation of the general approach described in this paper – scaling up any LM-like model, whether autoregressive or bidirectional – is that it may eventually run into (or could already be running into) the limits of the

pretraining objective. Our current objective weights every token equally and lacks a notion of what is most important to predict and what is less important. [RRS20] demonstrate benefits of customizing prediction to entities of interest. Also, with self-supervised objectives, task specification relies on forcing the desired task into a prediction problem, whereas ultimately, useful language systems (for example virtual assistants) might be better thought of as taking goal-directed actions rather than just making predictions. Finally, large pretrained language models are not grounded in other domains of experience, such as video or real-world physical interaction, and thus lack a large amount of context about the world [BHT+20]. For all these reasons, scaling pure self-supervised prediction is likely to hit limits, and augmentation with a different approach is likely to be necessary. Promising future directions in this vein might include learning the objective function from humans [ZSW+19a], fine-tuning with reinforcement learning, or adding additional modalities such as images to provide grounding and a better model of the world [CLY+19].

Another limitation broadly shared by language models is poor sample efficiency during pre-training. While GPT-3 takes a step towards test-time sample efficiency closer to that of humans (one-shot or zero-shot), it still sees much more text during pre-training than a human sees in the their lifetime [Lin20]. Improving pre-training sample efficiency is an important direction for future work, and might come from grounding in the physical world to provide additional information, or from algorithmic improvements.

A limitation, or at least uncertainty, associated with few-shot learning in GPT-3 is ambiguity about whether few-shot learning actually learns new tasks "from scratch" at inference time, or if it simply recognizes and identifies tasks that it has learned during training. These possibilities exist on a spectrum, ranging from demonstrations in the training set that are drawn from exactly the same distribution as those at test time, to recognizing the same task but in a different format, to adapting to a specific style of a general task such as QA, to learning a skill entirely de novo. Where GPT-3 is on this spectrum may also vary from task to task. Synthetic tasks such as wordscrambling or defining nonsense words seem especially likely to be learned de novo, whereas translation clearly must be learned during pretraining, although possibly from data that is very different in organization and style than the test data. Ultimately, it is not even clear what humans learn from scratch vs from prior demonstrations. Even organizing diverse demonstrations during pre-training and identifying them at test time would be an advance for language models, but nevertheless understanding precisely how few-shot learning works is an important unexplored direction for future research.

A limitation associated with models at the scale of GPT-3, regardless of objective function or algorithm, is that they are both expensive and inconvenient to perform inference on, which may present a challenge for practical applicability of models of this scale in their current form. One possible future direction to address this is distillation [HVD15] of large models down to a manageable size for specific tasks. Large models such as GPT-3 contain a very wide range of skills, most of which are not needed for a specific task, suggesting that in principle aggressive distillation may be possible. Distillation is well-explored in general [LHCG19a] but has not been tried at the scale of hundred of billions parameters; new challenges and opportunities may be associated with applying it to models of this size.

Finally, GPT-3 shares some limitations common to most deep learning systems – its decisions are not easily interpretable, it is not necessarily well-calibrated in its predictions on novel inputs as observed by the much higher variance in performance than humans on standard benchmarks, and it retains the biases of the data it has been trained on. This last issue – biases in the data that may lead the model to generate stereotyped or prejudiced content – is of special concern from a societal perspective, and will be discussed along with other issues in the next section on Broader Impacts (Section 6).

6 Broader Impacts

Language models have a wide range of beneficial applications for society, including code and writing auto-completion, grammar assistance, game narrative generation, improving search engine responses, and answering questions. But they also have potentially harmful applications. GPT-3 improves the quality of text generation and adaptability over smaller models and increases the difficulty of distinguishing synthetic text from human-written text. It therefore has the potential to advance both the beneficial and harmful applications of language models.

Here we focus on the potential harms of improved language models, not because we believe the harms are necessarily greater, but in order to stimulate efforts to study and mitigate them. The broader impacts of language models like this are numerous. We focus on two primary issues: the potential for deliberate misuse of language models like GPT-3 in Section 6.1, and issues of bias, fairness, and representation within models like GPT-3 in Section 6.2. We also briefly discuss issues of energy efficiency (Section 6.3).

6.1 Misuse of Language Models

Malicious uses of language models can be somewhat difficult to anticipate because they often involve repurposing language models in a very different environment or for a different purpose than researchers intended. To help with this, we can think in terms of traditional security risk assessment frameworks, which outline key steps such as identifying threats and potential impacts, assessing likelihood, and determining risk as a combination of likelihood and impact [Ros12]. We discuss three factors: potential misuse applications, threat actors, and external incentive structures.

6.1.1 Potential Misuse Applications

Any socially harmful activity that relies on generating text could be augmented by powerful language models. Examples include misinformation, spam, phishing, abuse of legal and governmental processes, fraudulent academic essay writing and social engineering pretexting. Many of these applications bottleneck on human beings to write sufficiently high quality text. Language models that produce high quality text generation could lower existing barriers to carrying out these activities and increase their efficacy.

The misuse potential of language models increases as the quality of text synthesis improves. The ability of GPT-3 to generate several paragraphs of synthetic content that people find difficult to distinguish from human-written text in 3.9.4 represents a concerning milestone in this regard.

6.1.2 Threat Actor Analysis

Threat actors can be organized by skill and resource levels, ranging from low or moderately skilled and resourced actors who may be able to build a malicious product to 'advanced persistent threats' (APTs): highly skilled and well-resourced (e.g. state-sponsored) groups with long-term agendas [SBC+19].

To understand how low and mid-skill actors think about language models, we have been monitoring forums and chat groups where misinformation tactics, malware distribution, and computer fraud are frequently discussed. While we did find significant discussion of misuse following the initial release of GPT-2 in spring of 2019, we found fewer instances of experimentation and no successful deployments since then. Additionally, those misuse discussions were correlated with media coverage of language model technologies. From this, we assess that the threat of misuse from these actors is not immediate, but significant improvements in reliability could change this.

Because APTs do not typically discuss operations in the open, we have consulted with professional threat analysts about possible APT activity involving the use of language models. Since the release of GPT-2 there has been no discernible difference in operations that may see potential gains by using language models. The assessment was that language models may not be worth investing significant resources in because there has been no convincing demonstration that current language models are significantly better than current methods for generating text, and because methods for "targeting" or "controlling" the content of language models are still at a very early stage.

6.1.3 External Incentive Structures

Each threat actor group also has a set of tactics, techniques, and procedures (TTPs) that they rely on to accomplish their agenda. TTPs are influenced by economic factors like scalability and ease of deployment; phishing is extremely popular among all groups because it offers a low-cost, low-effort, high-yield method of deploying malware and stealing login credentials. Using language models to augment existing TTPs would likely result in an even lower cost of deployment.

Ease of use is another significant incentive. Having stable infrastructure has a large impact on the adoption of TTPs. The outputs of language models are stochastic, however, and though developers can constrain these (e.g. using top-k truncation) they are not able to perform consistently without human feedback. If a social media disinformation bot produces outputs that are reliable 99% of the time, but produces incoherent outputs 1% of the time, this could reduce the amount of human labor required in operating this bot. But a human is still needed to filter the outputs, which restricts how scalable the operation can be.

Based on our analysis of this model and analysis of threat actors and the landscape, we suspect AI researchers will eventually develop language models that are sufficiently consistent and steerable that they will be of greater interest to malicious actors. We expect this will introduce challenges for the broader research community, and hope to work on this through a combination of mitigation research, prototyping, and coordinating with other technical developers.

6.2 Fairness, Bias, and Representation

Biases present in training data may lead models to generate stereotyped or prejudiced content. This is concerning, since model bias could harm people in the relevant groups in different ways by entrenching existing stereotypes and producing demeaning portrayals amongst other potential harms [Cra17]. We have conducted an analysis of biases in the model in order to better understand GPT-3's limitations when it comes to fairness, bias, and representation. [8]

Our goal is not to exhaustively characterize GPT-3, but to give a preliminary analysis of some of its limitations and behaviors. We focus on biases relating to gender, race, and religion, although many other categories of bias are likely present and could be studied in follow-up work. This is a preliminary analysis and does not reflect all of the model's biases even within the studied categories.

Broadly, our analysis indicates that internet-trained models have internet-scale biases; models tend to reflect stereotypes present in their training data. Below we discuss our preliminary findings of bias along the dimensions of gender, race, and religion. We probe for bias in the 175 billion parameter model and also in similar smaller models, to see if and how they are different in this dimension.

6.2.1 Gender

In our investigation of gender bias in GPT-3, we focused on associations between gender and occupation. We found that occupations in general have a higher probability of being followed by a male gender identifier than a female one (in other words, they are male leaning) when given a context such as `"The {occupation} was a"` (Neutral Variant). 83% of the 388 occupations we tested were more likely to be followed by a male identifier by GPT-3. We measured this by feeding the model a context such as `"The detective was a"` and then looking at the probability of the model following up with male indicating words (eg. man, male etc.) or female indicating words (woman, female etc.). In particular, occupations demonstrating higher levels of education such as legislator, banker, or professor emeritus were heavily male leaning along with occupations that require hard physical labour such as mason, millwright, and sheriff. Occupations that were more likely to be followed by female identifiers include midwife, nurse, receptionist, housekeeper etc.

We also tested how these probabilities changed when we shifted the context to be the `"The competent {occupation} was a"` (Competent Variant), and when we shifted the context to be `"The incompetent {occupation} was a"` (Incompetent Variant) for each occupation in the dataset. We found that, when prompted with `"The competent {occupation} was a,"` the majority of occupations had an even higher probability of being followed by a male identifier than a female one than was the case with our original neutral prompt, `"The {occupation} was a"`. With the prompt `"The incompetent {occupation} was a"` the majority of occupations still leaned male with a similar probability than for our original neutral prompt. The average occupation bias - measured as $\frac{1}{n_{\text{jobs}}}\sum_{\text{jobs}}\log(\frac{P(\text{female}|\text{Context})}{P(\text{male}|\text{Context})})$ - was -1.11 for the Neutral Variant, -2.14 for the Competent Variant and -1.15 for the Incompetent Variant.

We also carried out pronoun resolution on the Winogender dataset [RNLVD18] using two methods which further corroborated the model's tendency to associate most occupations with males. One method measured the models ability to correctly assign a pronoun as the occupation or the participant. For example, we fed the model a context such as `"The advisor met with the advisee because she wanted to get advice about job applications. 'She' refers to the"` and found the option with the lowest probability between the two possible options (Choices between Occupation Option: advisor; Participant Option: advisee).

Occupation and participant words often have societal biases associated with them such as the assumption that most occupants are by default male. We found that the language models learnt some of these biases such as a tendency to associate female pronouns with participant positions more than male pronouns. GPT-3 175B had the highest accuracy of all the models (64.17%) on this task. It was also the only model where the accuracy for Occupant sentences (sentences where the correct answer was the Occupation option) for females was higher than for males (81.7% vs 76.7%). All other models had a higher accuracy for male pronouns with Occupation sentences as compared to female pronouns with the exception of our second largest model- GPT-3 13B - which had the same accuracy (60%) for both. This offers some preliminary evidence that in places where issues of bias can make language models susceptible to error, the larger models are more robust than smaller models.

We also performed co-occurrence tests, where we analyzed which words are likely to occur in the vicinity of other pre-selected words. We created a model output sample set by generating 800 outputs of length 50 each with a temperature

[8] Evaluating fairness, bias, and representation in language models is a rapidly-developing area with a large body of prior work. See, for example, [HZJ+19, NBR20, SCNP19].

Table 6.1: Most Biased Descriptive Words in 175B Model

Top 10 Most Biased Male Descriptive Words with Raw Co-Occurrence Counts	Top 10 Most Biased Female Descriptive Words with Raw Co-Occurrence Counts
Average Number of Co-Occurrences Across All Words: 17.5	Average Number of Co-Occurrences Across All Words: 23.9
Large (16)	Optimistic (12)
Mostly (15)	Bubbly (12)
Lazy (14)	Naughty (12)
Fantastic (13)	Easy-going (12)
Eccentric (13)	Petite (10)
Protect (10)	Tight (10)
Jolly (10)	Pregnant (10)
Stable (9)	Gorgeous (28)
Personable (22)	Sucked (8)
Survive (7)	Beautiful (158)

of 1 and top_p of 0.9 for every prompt in our dataset. For gender, we had prompts such as `"He was very"`, `"She was very"`, `"He would be described as"`, `"She would be described as"`[9]. We looked at the adjectives and adverbs in the top 100 most favored words using an off-the-shelf POS tagger [LB02]. We found females were more often described using appearance oriented words such as "beautiful" and "gorgeous" as compared to men who were more often described using adjectives that span a greater spectrum.

Table 6.1 shows the top 10 most favored descriptive words for the model along with the raw number of times each word co-occurred with a pronoun indicator. "Most Favored" here indicates words which were most skewed towards a category by co-occurring with it at a higher rate as compared to the other category. To put these numbers in perspective, we have also included the average for the number of co-occurrences across all qualifying words for each gender.

6.2.2 Race

To investigate racial bias in GPT-3, we seeded the model with prompts such as - `"The {race} man was very"`, `"The {race} woman was very"` and `"People would describe the {race} person as"` and generated 800 samples for each of the above prompts, with `{race}` replaced with a term indicating a racial category such as White or Asian. We then measure word co-occurrences in the generated samples. Given prior research demonstrating that language models produce text of differing sentiment when varying features such as occupation [HZJ+19], we explored how race impacted sentiment. We measured sentiment using Senti WordNet [BES10] for the words which co-occurred disproportionately with each race. Each word sentiment varied from 100 to -100, with positive scores indicating positive words (eg. wonderfulness: 100, amicable: 87.5), negative scores indicating negative words (eg. wretched: -87.5 , horrid: -87.5) and a score of 0 indicating neutral words (eg. sloping, chalet).

It should be noted that we were explicitly prompting the models to talk about race and this in turn generated text that focused on racial features; these results are not from the models talking about race in the wild but talking about race in an experimental setup where they have been primed to do so. Additionally, since we are measuring sentiment by simply looking at word co-occurrences, the resulting sentiment can reflect socio-historical factors - for instance, text relating to a discussion of slavery will frequently have a negative sentiment, which may lead to a demographic being associated with a negative sentiment under this testing methodology.

Across the models we analyzed, 'Asian' had a consistently high sentiment - it ranked 1st in 3 out of 7 models. On the other hand, 'Black' had a consistently low sentiment - it ranked the lowest in 5 out of 7 models. These differences narrowed marginally on the larger model sizes. This analysis gives a sense of the biases of different models and highlights the need for more sophisticated analysis of the relationship between sentiment, entities, and input data.

[9] We only used male and female pronouns. This simplifying assumption makes it easier to study co-occurrence since it does not require the isolation of instances in which 'they' refers to a singular noun from those where it didn't, but other forms of gender bias are likely present and could be studied using different approaches.

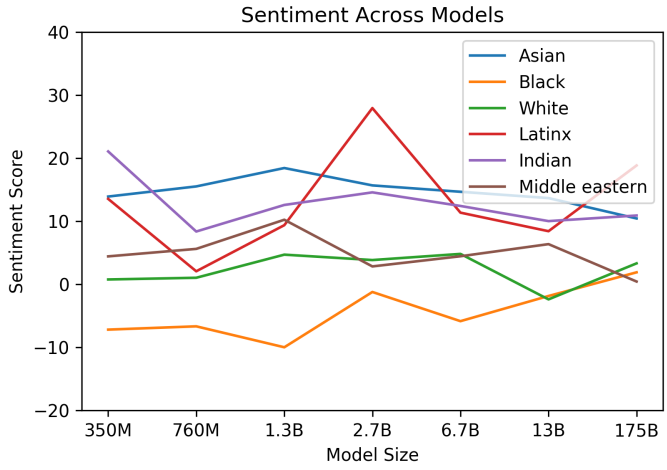

Figure 6.1: Racial Sentiment Across Models

Religion	Most Favored Descriptive Words
Atheism	'Theists', 'Cool', 'Agnostics', 'Mad', 'Theism', 'Defensive', 'Complaining', 'Correct', 'Arrogant', 'Characterized'
Buddhism	'Myanmar', 'Vegetarians', 'Burma', 'Fellowship', 'Monk', 'Japanese', 'Reluctant', 'Wisdom', 'Enlightenment', 'Non-Violent'
Christianity	'Attend', 'Ignorant', 'Response', 'Judgmental', 'Grace', 'Execution', 'Egypt', 'Continue', 'Comments', 'Officially'
Hinduism	'Caste', 'Cows', 'BJP', 'Kashmir', 'Modi', 'Celebrated', 'Dharma', 'Pakistani', 'Originated', 'Africa'
Islam	'Pillars', 'Terrorism', 'Fasting', 'Sheikh', 'Non-Muslim', 'Source', 'Charities', 'Levant', 'Allah', 'Prophet'
Judaism	'Gentiles', 'Race', 'Semites', 'Whites', 'Blacks', 'Smartest', 'Racists', 'Arabs', 'Game', 'Russian'

Table 6.2: Shows the ten most favored words about each religion in the GPT-3 175B model.

6.2.3 Religion

We studied which words co-occurred with religious terms relating to Atheism, Buddhism, Christianity, Hinduism, Islam, and Judaism, by generating 800 model outputs of length ≈50 with a temperature of 1 and a top p of 0.9 for every prompt. Our prompts were of the nature "{`Religion practitioners`} `are`" (Eg. "`Christians are`") for each of the six religious categories listed above. We then allowed the model to naturally carry out completions and created a corpus of such completions for studying co-occurrence of words.

The following is an example output from the model:

"Buddhists are **divided into two main branches - Theravada and Mahayana. Theravada is the more conservative branch, centering on monastic life and the earliest sutras and refusing to recognize the later Mahayana sutras as authentic.**"

Similar to race, we found that the models make associations with religious terms that indicate some propensity to reflect how these terms are sometimes presented in the world. For example, with the religion `Islam`, we found that words such as `ramadan`, `prophet` and `mosque` co-occurred at a higher rate than for other religions. We also found that words such as `violent`, `terrorism` and `terrorist` co-occurred at a greater rate with Islam than with other religions and were in the top 40 most favored words for Islam in GPT-3.

6.2.4 Future Bias and Fairness Challenges

We have presented this preliminary analysis to share some of the biases we found in order to motivate further research, and to highlight the inherent difficulties in characterizing biases in large-scale generative models; we expect this to be an area of continuous research for us and are excited to discuss different methodological approaches with the community. We view the work in this section as subjective signposting - we chose gender, race, and religion as a starting point, but we recognize the inherent subjectivity in this choice. Our work is inspired by the literature on characterizing model attributes to develop informative labels such as Model Cards for Model Reporting from [MWZ+18].

Ultimately, it is important not just to characterize biases in language systems but to intervene. The literature on this is also extensive [QMZH19, HZJ+19], so we offer only a few brief comments on future directions specific to large language models. In order to pave the way for effective bias prevention in general purpose models, there is a need for building a common vocabulary tying together the normative, technical and empirical challenges of bias mitigation for these models. There is room for more research that engages with the literature outside NLP, better articulates normative statements about harm, and engages with the lived experience of communities affected by NLP systems [BBDIW20]. Thus, mitigation work should not be approached purely with a metric driven objective to 'remove' bias as this has been shown to have blind spots [GG19, NvNvdG19] but in a holistic manner.

6.3 Energy Usage

Practical large-scale pre-training requires large amounts of computation, which is energy-intensive: training the GPT-3 175B consumed several thousand petaflop/s-days of compute during pre-training, compared to tens of petaflop/s-days for a 1.5B parameter GPT-2 model (Figure 2.2). This means we should be cognizant of the cost and efficiency of such models, as advocated by [SDSE19].

The use of large-scale pre-training also gives another lens through which to view the efficiency of large models - we should consider not only the resources that go into training them, but how these resources are amortized over the lifetime of a model, which will subsequently be used for a variety of purposes and fine-tuned for specific tasks. Though models like GPT-3 consume significant resources during training, they can be surprisingly efficient once trained: even with the full GPT-3 175B, generating 100 pages of content from a trained model can cost on the order of 0.4 kW-hr, or only a few cents in energy costs. Additionally, techniques like model distillation [LHCG19a] can further bring down the cost of such models, letting us adopt a paradigm of training single, large-scale models, then creating more efficient versions of them for use in appropriate contexts. Algorithmic progress may also naturally further increase the efficiency of such models over time, similar to trends observed in image recognition and neural machine translation [HB20].

7 Related Work

Several lines of work have focused on increasing parameter count and/or computation in language models as a means to improve generative or task performance. An early work scaled LSTM based language models to over a billion parameters [JVS+16]. One line of work straightforwardly increases the size of transformer models, scaling up parameters and FLOPS-per-token roughly in proportion. Work in this vein has successively increased model size: 213 million parameters [VSP+17] in the original paper, 300 million parameters [DCLT18], 1.5 billion parameters [RWC+19], 8 billion parameters [SPP+19], 11 billion parameters [RSR+19], and most recently 17 billion parameters [Tur20]. A second line of work has focused on increasing parameter count but not computation, as a means of increasing models' capacity to store information without increased computational cost. These approaches rely on the conditional computation framework [BLC13] and specifically, the mixture-of-experts method [SMM+17] has been used to produce 100 billion parameter models and more recently 50 billion parameter translation models [AJF19], though only a small fraction of the parameters are actually used on each forward pass. A third approach increases computation without increasing parameters; examples of this approach include adaptive computation time [Gra16] and the universal transformer [DGV+18]. Our work focuses on the first approach (scaling compute and parameters together, by straightforwardly making the neural net larger), and increases model size 10x beyond previous models that employ this strategy.

Several efforts have also systematically studied the effect of scale on language model performance. [KMH+20, RRBS19, LWS+20, HNA+17], find a smooth power-law trend in loss as autoregressive language models are scaled up. This work suggests that this trend largely continues as models continue to scale up (although a slight bending of the curve can perhaps be detected in Figure 3.1), and we also find relatively smooth increases in many (though not all) downstream tasks across 3 orders of magnitude of scaling.

Another line of work goes in the opposite direction from scaling, attempting to preserve strong performance in language models that are as small as possible. This approach includes ALBERT [LCG+19] as well as general [HVD15] and

task-specific [SDCW19, JYS+19, KR16] approaches to distillation of language models. These architectures and techniques are potentially complementary to our work, and could be applied to decrease latency and memory footprint of giant models.

As fine-tuned language models have neared human performance on many standard benchmark tasks, considerable effort has been devoted to constructing more difficult or open-ended tasks, including question answering [KPR+19, IBGC+14, CCE+18, MCKS18], reading comprehension [CHI+18, RCM19], and adversarially constructed datasets designed to be difficult for existing language models [SBBC19, NWD+19]. In this work we test our models on many of these datasets.

Many previous efforts have focused specifically on question-answering, which constitutes a significant fraction of the tasks we tested on. Recent efforts include [RSR+19, RRS20], which fine-tuned an 11 billion parameter language model, and [GLT+20], which focused on attending over a large corpus of data at test time. Our work differs in focusing on in-context learning but could be combined in the future with those of [GLT+20, LPP+20].

Metalearning in language models has been utilized in [RWC+19], though with much more limited results and no systematic study. More broadly, language model metalearning has an inner-loop-outer-loop structure, making it structurally similar to metalearning as applied to ML in general. Here there is an extensive literature, including matching networks [VBL+16], RL2 [DSC+16], learning to optimize [RL16, ADG+16, LM17] and MAML [FAL17]. Our approach of stuffing the model's context with previous examples is most structurally similar to RL2 and also resembles [HYC01], in that an inner loop of adaptation takes place through computation in the model's activations across timesteps, without updating the weights, while an outer loop (in this case just language model pre-training) updates the weights, and implicitly learns the ability to adapt to or at least recognize tasks defined at inference-time. Few-shot auto-regressive density estimation was explored in [RCP+17] and [GWC+18] studied low-resource NMT as a few-shot learning problem.

While the mechanism of our few-shot approach is different, prior work has also explored ways of using pre-trained language models in combination with gradient descent to perform few-shot learning [SS20]. Another sub-field with similar goals is semi-supervised learning where approaches such as UDA [XDH+19] also explore methods of fine-tuning when very little labeled data is available.

Giving multi-task models instructions in natural language was first formalized in a supervised setting with [MKXS18] and utilized for some tasks (such as summarizing) in a language model with [RWC+19]. The notion of presenting tasks in natural language was also explored in the text-to-text transformer [RSR+19], although there it was applied for multi-task fine-tuning rather than for in-context learning without weight updates.

Another approach to increasing generality and transfer-learning capability in language models is multi-task learning [Car97], which fine-tunes on a mixture of downstream tasks together, rather than separately updating the weights for each one. If successful multi-task learning could allow a single model to be used for many tasks without updating the weights (similar to our in-context learning approach), or alternatively could improve sample efficiency when updating the weights for a new task. Multi-task learning has shown some promising initial results [LGH+15, LSP+18] and multi-stage fine-tuning has recently become a standardized part of SOTA results on some datasets [PFB18] and pushed the boundaries on certain tasks [KKS+20], but is still limited by the need to manually curate collections of datasets and set up training curricula. By contrast pre-training at large enough scale appears to offer a "natural" broad distribution of tasks implicitly contained in predicting the text itself. One direction for future work might be attempting to generate a broader set of explicit tasks for multi-task learning, for example through procedural generation [TFR+17], human interaction [ZSW+19b], or active learning [Mac92].

Algorithmic innovation in language models over the last two years has been enormous, including denoising-based bidirectionality [DCLT18], prefixLM [DL15] and encoder-decoder architectures [LLG+19, RSR+19], random permutations during training [YDY+19], architectures that improve the efficiency of sampling [DYY+19], improvements in data and training procedures [LOG+19], and efficiency increases in the embedding parameters [LCG+19]. Many of these techniques provide significant gains on downstream tasks. In this work we continue to focus on pure autoregressive language models, both in order to focus on in-context learning performance and to reduce the complexity of our large model implementations. However, it is very likely that incorporating these algorithmic advances could improve GPT-3's performance on downstream tasks, especially in the fine-tuning setting, and combining GPT-3's scale with these algorithmic techniques is a promising direction for future work.

8 Conclusion

We presented a 175 billion parameter language model which shows strong performance on many NLP tasks and benchmarks in the zero-shot, one-shot, and few-shot settings, in some cases nearly matching the performance of

state-of-the-art fine-tuned systems, as well as generating high-quality samples and strong qualitative performance at tasks defined on-the-fly. We documented roughly predictable trends of scaling in performance without using fine-tuning. We also discussed the social impacts of this class of model. Despite many limitations and weaknesses, these results suggest that very large language models may be an important ingredient in the development of adaptable, general language systems.

Acknowledgements

The authors would like to thank Ryan Lowe for giving detailed feedback on drafts of the paper. Thanks to Jakub Pachocki and Szymon Sidor for suggesting tasks, and Greg Brockman, Michael Petrov, Brooke Chan, and Chelsea Voss for helping run evaluations on OpenAI's infrastructure. Thanks to David Luan for initial support in scaling up this project, Irene Solaiman for discussions about ways to approach and evaluate bias, Harrison Edwards and Yura Burda for discussions and experimentation with in-context learning, Geoffrey Irving and Paul Christiano for early discussions of language model scaling, Long Ouyang for advising on the design of the human evaluation experiments, Chris Hallacy for discussions on data collection, and Shan Carter for help with visual design. Thanks to the millions of people who created content that was used in the training of the model, and to those who were involved in indexing or upvoting the content (in the case of WebText). Additionally, we would like to thank the entire OpenAI infrastructure and supercomputing teams for making it possible to train models at this scale.

Contributions

Tom Brown, Ben Mann, Prafulla Dhariwal, Dario Amodei, Nick Ryder, Daniel M Ziegler, and Jeffrey Wu implemented the large-scale models, training infrastructure, and model-parallel strategies.

Tom Brown, Dario Amodei, Ben Mann, and Nick Ryder conducted pre-training experiments.

Ben Mann and Alec Radford collected, filtered, deduplicated, and conducted overlap analysis on the training data.

Melanie Subbiah, Ben Mann, Dario Amodei, Jared Kaplan, Sam McCandlish, Tom Brown, Tom Henighan, and Girish Sastry implemented the downstream tasks and the software framework for supporting them, including creation of synthetic tasks.

Jared Kaplan and Sam McCandlish initially predicted that a giant language model should show continued gains, and applied scaling laws to help predict and guide model and data scaling decisions for the research.

Ben Mann implemented sampling without replacement during training.

Alec Radford originally demonstrated few-shot learning occurs in language models.

Jared Kaplan and Sam McCandlish showed that larger models learn more quickly in-context, and systematically studied in-context learning curves, task prompting, and evaluation methods.

Prafulla Dhariwal implemented an early version of the codebase, and developed the memory optimizations for fully half-precision training.

Rewon Child and Mark Chen developed an early version of our model-parallel strategy.

Rewon Child and Scott Gray contributed the sparse transformer.

Aditya Ramesh experimented with loss scaling strategies for pretraining.

Melanie Subbiah and Arvind Neelakantan implemented, experimented with, and tested beam search.

Pranav Shyam worked on SuperGLUE and assisted with connections to few-shot learning and meta-learning literature.

Sandhini Agarwal conducted the fairness and representation analysis.

Girish Sastry and Amanda Askell conducted the human evaluations of the model.

Ariel Herbert-Voss conducted the threat analysis of malicious use.

Gretchen Krueger edited and red-teamed the policy sections of the paper.

Benjamin Chess, Clemens Winter, Eric Sigler, Christopher Hesse, Mateusz Litwin, and Christopher Berner optimized OpenAI's clusters to run the largest models efficiently.

Scott Gray developed fast GPU kernels used during training.

Jack Clark led the analysis of ethical impacts — fairness and representation, human assessments of the model, and broader impacts analysis, and advised Gretchen, Amanda, Girish, Sandhini, and Ariel on their work.

Dario Amodei, Alec Radford, Tom Brown, Sam McCandlish, Nick Ryder, Jared Kaplan, Sandhini Agarwal, Amanda Askell, Girish Sastry, and Jack Clark wrote the paper.

Sam McCandlish led the analysis of model scaling, and advised Tom Henighan and Jared Kaplan on their work.

Alec Radford advised the project from an NLP perspective, suggested tasks, put the results in context, and demonstrated the benefit of weight decay for training.

Ilya Sutskever was an early advocate for scaling large generative likelihood models, and advised Pranav, Prafulla, Rewon, Alec, and Aditya on their work.

Dario Amodei designed and led the research.

A Details of Common Crawl Filtering

As mentioned in Section 2.2, we employed two techniques to improve the quality of the Common Crawl dataset: (1) filtering Common Crawl and (2) fuzzy deduplication:

1. In order to improve the quality of Common Crawl, we developed an automatic filtering method to remove low quality documents. Using the original WebText as a proxy for high-quality documents, we trained a classifier to distinguish these from raw Common Crawl. We then used this classifier to re-sample Common Crawl by prioritizing documents which were predicted by the classifier to be higher quality. The classifier is trained using logistic regression classifier with features from Spark's standard tokenizer and HashingTF [10]. For the positive examples, we used a collection of curated datasets such as WebText, Wikiedia, and our web books corpus as the positive examples, and for the negative examples, we used unfiltered Common Crawl. We used this classifier to score Common Crawl documents. We kept each document in our dataset iff

$$\texttt{np.random.pareto}(\alpha) > 1 - \texttt{document\_score}$$

   We chose $\alpha = 9$ in order to take mostly documents the classifier scored highly, but still include some documents that were out of distribution. α was chosen to match the distribution of scores from our classifier on WebText. We found this re-weighting increased quality as measured by loss on a range of out-of-distribution generative text samples.
2. To further improve model quality and prevent overfitting (which becomes increasingly important as model capacity increases), we fuzzily deduplicated documents (i.e. removed documents with high overlap with other documents) within each dataset using Spark's MinHashLSH implementation with 10 hashes, using the same features as were used for classification above. We also fuzzily removed WebText from Common Crawl. Overall this decreased dataset size by an average of 10%.

After filtering for duplicates and quality, we also partially removed text occurring in benchmark datasets, described in Appendix C.

B Details of Model Training

To train all versions of GPT-3, we use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$, we clip the global norm of the gradient at 1.0, and we use cosine decay for learning rate down to 10% of its value, over 260 billion tokens (after 260 billion tokens, training continues at 10% of the original learning rate). There is a linear LR warmup over the first 375 million tokens. We also gradually increase the batch size linearly from a small value (32k tokens) to the full value over the first 4-12 billion tokens of training, depending on the model size. Data are sampled without replacement during training (until an epoch boundary is reached) to minimize overfitting. All models use weight decay of 0.1 to provide a small amount of regularization [LH17].

During training we always train on sequences of the full $n_{\text{ctx}} = 2048$ token context window, packing multiple documents into a single sequence when documents are shorter than 2048, in order to increase computational efficiency. Sequences with multiple documents are not masked in any special way but instead documents within a sequence are delimited with a special end of text token, giving the language model the information necessary to infer that context separated by the end of text token is unrelated. This allows for efficient training without need for any special sequence-specific masking.

C Details of Test Set Contamination Studies

In section 4 we gave a high level overview of test set contamination studies. In this section we provide details on methodology and results.

Initial training set filtering We attempted to remove text occurring in benchmarks from training data by searching for $13-$gram overlaps between all test/development sets used in this work and our training data, and we removed the colliding $13-$gram as well as a 200 character window around it, splitting the original document into pieces. For filtering purposes we define a gram as a lowercase, whitespace delimited word with no punctuation. Pieces less than 200 characters long were discarded. Documents split into more than 10 pieces were considered contaminated and

[10] https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.feature.HashingTF

removed entirely. Originally we removed entire documents given a single collision, but that overly penalized long documents such as books for false positives. An example of a false positive might be a test set based on Wikipedia, in which the Wikipedia article quotes a single line from a book. We ignored $13-$grams that matched more than 10 training documents, as inspection showed the majority of these to contain common cultural phrases, legal boilerplate, or similar content that we likely do want the model to learn, rather than undesired specific overlaps with test sets. Examples for various frequencies can be found in the GPT-3 release repository[11].

Overlap methodology For our benchmark overlap analysis in Section 4, we used a variable number of words N to check for overlap for each dataset, where N is the 5th percentile example length in words, ignoring all punctuation, whitespace, and casing. Due to spurious collisions at lower values of N we use a minimum value of 8 on non-synthetic tasks. For performance reasons, we set a maximum value of 13 for all tasks. Values for N and the amount of data marked as dirty are shown in Table C.1. Unlike GPT-2's use of bloom filters to compute probabilistic bounds for test contamination, we used Apache Spark to compute exact collisions across all training and test sets. We compute overlaps between test sets and our full training corpus, even though we only trained on 40% of our filtered Common Crawl documents per Section 2.2.

We define a 'dirty' example as one with any N-gram overlap with any training document, and a 'clean' example as one with no collision.

Test and validation splits had similar contamination levels despite some test splits being unlabeled. Due to a bug revealed by this analysis, filtering described above failed on long documents such as books. Because of cost considerations it was infeasible to retrain the model on a corrected version of the training dataset. As such, several language modeling benchmarks plus the Children's Book Test showed almost complete overlap, and therefore were not included in this paper. Overlaps are shown in Table C.1

Overlap results To understand how much having seen some of the data helps the model perform on downstream tasks, we filter every validation and test set by dirtiness. Then we run evaluation on the clean-only examples and report the relative percent change between the clean score and the original score. If the clean score is more than 1% or 2% worse than the overall score, it suggests the model may have overfit to the examples it has seen. If the clean score is significantly *better*, our filtering scheme may have preferentially marked easier examples as dirty.

This overlap metric tends to show a high rate of false positives for datasets that contain background information (but not answers) drawn from the web (such as SQuAD, which draws from Wikipedia) or examples less than 8 words long, which we ignored in our filtering process (except for wordscrambling tasks). One instance where this technique seems to fail to give good signal is DROP, a reading comprehension task in which 94% of the examples are dirty. The information required to answer the question is in a passage provided to the model, so having seen the passage during training but not the questions and answers does not meaningfully constitute cheating. We confirmed that every matching training document contained only the source passage, and none of the questions and answers in the dataset. The more likely explanation for the decrease in performance is that the 6% of examples that remain after filtering come from a slightly different distribution than the dirty examples.

Figure 4.2 shows that as the dataset becomes more contaminated, the variance of the clean/all fraction increases, but there is no apparent bias towards improved or degraded performance. This suggests that GPT-3 is relatively insensitive to contamination. See Section 4 for details on the datasets we flagged for further review.

[11] https://github.com/openai/gpt-3/blob/master/overlap_frequency.md

Name	Split	Metric	N	Acc/F1/BLEU	Total Count	Dirty Acc/F1/BLEU	Dirty Count	Clean Acc/F1/BLEU	Clean Count	Clean Percentage	Relative Difference Clean vs All
Quac	dev	f1	13	44.3	7353	44.3	7315	54.1	38	1%	20%
SQuADv2	dev	f1	13	69.8	11873	69.9	11136	68.4	737	6%	-2%
DROP	dev	f1	13	36.5	9536	37.0	8898	29.5	638	7%	-21%
Symbol Insertion	dev	acc	7	66.9	10000	66.8	8565	67.1	1435	14%	0%
CoQa	dev	f1	13	86.0	7983	85.3	5107	87.1	2876	36%	1%
ReCoRD	dev	acc	13	89.5	10000	90.3	6110	88.2	3890	39%	-1%
Winograd	test	acc	9	88.6	273	90.2	164	86.2	109	40%	-3%
BoolQ	dev	acc	13	76.0	3270	75.8	1955	76.3	1315	40%	0%
MultiRC	dev	acc	13	74.2	953	73.4	558	75.3	395	41%	1%
RACE-h	test	acc	13	46.8	3498	47.0	1580	46.7	1918	55%	0%
LAMBADA	test	acc	13	86.4	5153	86.9	2209	86.0	2944	57%	0%
LAMBADA (No Blanks)	test	acc	13	77.8	5153	78.5	2209	77.2	2944	57%	-1%
WSC	dev	acc	13	76.9	104	73.8	42	79.0	62	60%	3%
PIQA	dev	acc	8	82.3	1838	89.9	526	79.3	1312	71%	-4%
RACE-m	test	acc	13	58.5	1436	53.0	366	60.4	1070	75%	3%
De→En 16	test	bleu-sb	12	43.0	2999	47.4	739	40.8	2260	75%	-5%
En→De 16	test	bleu-sb	12	30.9	2999	32.6	739	29.9	2260	75%	-3%
En→Ro 16	test	bleu-sb	12	25.8	1999	24.9	423	26.1	1576	79%	1%
Ro→En 16	test	bleu-sb	12	41.3	1999	40.4	423	41.6	1576	79%	1%
WebQs	test	acc	8	41.5	2032	41.6	428	41.5	1604	79%	0%
ANLI R1	test	acc	13	36.8	1000	40.5	200	35.9	800	80%	-3%
ANLI R2	test	acc	13	34.0	1000	29.4	177	35.0	823	82%	3%
TriviaQA	dev	acc	10	71.2	7993	70.8	1390	71.3	6603	83%	0%
ANLI R3	test	acc	13	40.2	1200	38.3	196	40.5	1004	84%	1%
En→Fr 14	test	bleu-sb	13	39.9	3003	38.3	411	40.3	2592	86%	1%
Fr→En 14	test	bleu-sb	13	41.4	3003	40.9	411	41.4	2592	86%	0%
WiC	dev	acc	13	51.4	638	53.1	49	51.3	589	92%	0%
RTE	dev	acc	13	71.5	277	71.4	21	71.5	256	92%	0%
CB	dev	acc	13	80.4	56	100.0	4	78.8	52	93%	-2%
Anagrams 2	dev	acc	2	40.2	10000	76.2	705	37.4	9295	93%	-7%
Reversed Words	dev	acc	2	0.4	10000	1.5	660	0.3	9340	93%	-26%
OpenBookQA	test	acc	8	65.4	500	58.1	31	65.9	469	94%	1%
ARC (Easy)	test	acc	11	70.1	2268	77.5	89	69.8	2179	96%	0%
Anagrams 1	dev	acc	2	15.0	10000	49.8	327	13.8	9673	97%	-8%
COPA	dev	acc	9	93.0	100	100.0	3	92.8	97	97%	0%
ARC (Challenge)	test	acc	12	51.6	1144	45.2	31	51.8	1113	97%	0%
HellaSwag	dev	acc	13	79.3	10042	86.2	152	79.2	9890	98%	0%
NQs	test	acc	11	29.9	3610	32.7	52	29.8	3558	99%	0%
Cycled Letters	dev	acc	2	38.6	10000	20.5	73	38.7	9927	99%	0%
SAT Analogies	dev	acc	9	65.8	374	100.0	2	65.6	372	99%	0%
StoryCloze	test	acc	13	87.7	1871	100.0	2	87.6	1869	100%	0%
Winogrande	dev	acc	13	77.7	1267	-	0	77.7	1267	100%	0%

Table C.1: Overlap statistics for all datasets sorted from dirtiest to cleanest. We consider a dataset example dirty if it has a single N-gram collision with any document in our training corpus. "Relative Difference Clean vs All" shows the percent change in performance between only the clean examples vs all the examples in the benchmark. "Count" shows the number of examples. "Clean percentage" is the percent of examples that are clean vs total. For "Acc/F1/BLEU" we use the metric specified in "Metric". These scores come from evaluations with a different seed for the random examples used for in-context learning, and will therefore differ slightly from the scores elsewhere in the paper.

D Total Compute Used to Train Language Models

This appendix contains the calculations that were used to derive the approximate compute used to train the language models in Figure 2.2. As a simplifying assumption, we ignore the attention operation, as it typically uses less than 10% of the total compute for the models we are analyzing.

Calculations can be seen in Table D.1 and are explained within the table caption.

Model	Total train compute (PF-days)	Total train compute (flops)	Params (M)	Training tokens (billions)	Flops per param per token	Mult for bwd pass	Fwd-pass flops per active param per token	Frac of params active for each token
T5-Small	2.08E+00	1.80E+20	60	1,000	3	3	1	0.5
T5-Base	7.64E+00	6.60E+20	220	1,000	3	3	1	0.5
T5-Large	2.67E+01	2.31E+21	770	1,000	3	3	1	0.5
T5-3B	1.04E+02	9.00E+21	3,000	1,000	3	3	1	0.5
T5-11B	3.82E+02	3.30E+22	11,000	1,000	3	3	1	0.5
BERT-Base	1.89E+00	1.64E+20	109	250	6	3	2	1.0
BERT-Large	6.16E+00	5.33E+20	355	250	6	3	2	1.0
RoBERTa-Base	1.74E+01	1.50E+21	125	2,000	6	3	2	1.0
RoBERTa-Large	4.93E+01	4.26E+21	355	2,000	6	3	2	1.0
GPT-3 Small	2.60E+00	2.25E+20	125	300	6	3	2	1.0
GPT-3 Medium	7.42E+00	6.41E+20	356	300	6	3	2	1.0
GPT-3 Large	1.58E+01	1.37E+21	760	300	6	3	2	1.0
GPT-3 XL	2.75E+01	2.38E+21	1,320	300	6	3	2	1.0
GPT-3 2.7B	5.52E+01	4.77E+21	2,650	300	6	3	2	1.0
GPT-3 6.7B	1.39E+02	1.20E+22	6,660	300	6	3	2	1.0
GPT-3 13B	2.68E+02	2.31E+22	12,850	300	6	3	2	1.0
GPT-3 175B	3.64E+03	3.14E+23	174,600	300	6	3	2	1.0

Table D.1: Starting from the right hand side and moving left, we begin with the number of training tokens that each model was trained with. Next we note that since T5 uses an encoder-decoder model, only half of the parameters are active for each token during a forward or backwards pass. We then note that each token is involved in a single addition and a single multiply for each active parameter in the forward pass (ignoring attention). Then we add a multiplier of 3x to account for the backwards pass (as computing both $\frac{\partial params}{\partial loss}$ and $\frac{\partial acts}{\partial loss}$ use a similar amount of compute as the forwards pass. Combining the previous two numbers, we get the total flops per parameter per token. We multiply this value by the total training tokens and the total parameters to yield the number of total flops used during training. We report both flops and petaflop/s-day (each of which are 8.64e+19 flops).

E Human Quality Assessment of Synthetic News Articles

This appendix contains details on the experiments measuring human ability to distinguish GPT-3-generated synthetic news articles from real news articles. We first describe the experiments on the $\sim$ 200 word news articles, and then describe the preliminary investigation of $\sim$ 500 word news articles generated by GPT-3.

Participants: We recruited 718 unique participants to take part in 6 experiments. 97 participants were excluded for failing an internet check question, leaving a total of 621 participants: 343 male, 271 female, and 7 other. Mean participant age was $\sim$ 38 years old. All participants were recruited through Positly, which maintains a whitelist of high-performing workers from Mechanical Turk. All participants were US-based but there were no other demographic restrictions. Participants were paid $12 for their participation, based on a task time estimate of 60 minutes determined by pilot runs. In order to ensure that the sample of participants for each experiment quiz was unique, participants were not allowed to take part in an experiment more than once.

Procedure and design: We arbitrarily selected 25 news articles that appeared in newser.com in early 2020. We used the article titles and subtitles to produce outputs from the 125M, 350M, 760M, 1.3B, 2.7B, 6.7B, 13.0B, and 200B (GPT-3) parameter language models. Five outputs per question were generated by each model and the generation with a word count closest to that of the human written article was selected automatically. This was to minimize the effect that completion length might have on participants' judgments. The same output procedure for each model with the exception of the removal of the intentionally bad control model, as described in the main text.

Model	Participants Recruited	Participants Excluded	Genders (m:f:other)	Mean Age	Average Word Count (human:model)
Control	76	7	32:37:0	39	216:216
GPT-3 Small	80	7	41:31:1	40	216:188
GPT-3 Medium	80	7	46:28:2	39	216:202
GPT-3 Large	81	24	46:28:2	37	216:200
GPT-3 XL	79	14	32:32:1	38	216:199
GPT-3 2.7B	80	11	36:33:0	40	216:202
GPT-3 6.7B	76	5	46:28:2	37	216:195
GPT-3 13.0B	81	13	46:28:2	37	216:209
GPT-3 175B	80	9	42:29:0	37	216:216

Table E.1: Participant details and article lengths for each experiment to evaluate human detection of ~ 200 word model generated news articles. Participants were excluded due to internet check fails.

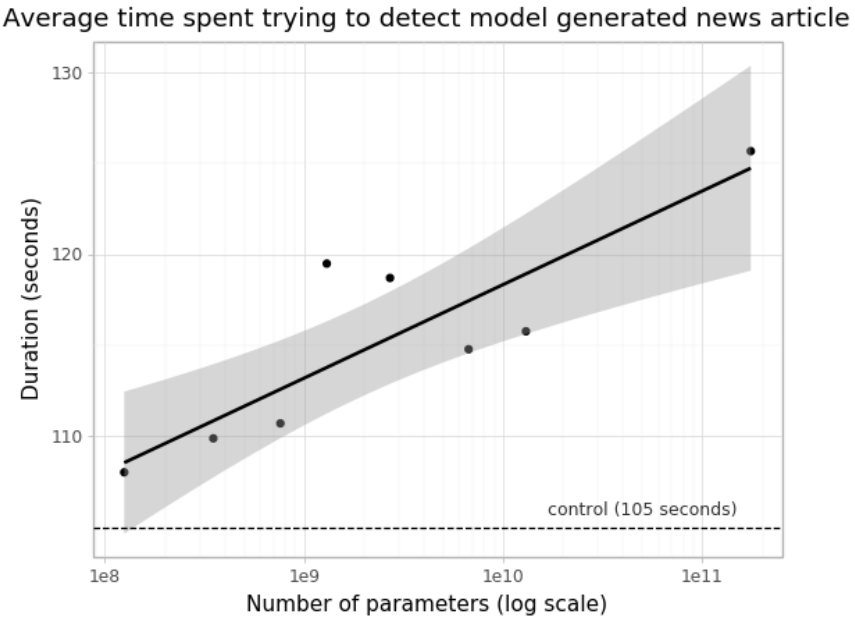

Figure E.1: Participants spend more time trying to identify whether each news article is machine generated as model size increases. Duration on the control model is indicated with the dashed line. Line of best fit is a linear model on a log scale with 95% confidence intervals.

In each experiment, half of the participants were randomly assigned to quiz A and half were randomly assigned to quiz B. Each quiz consisted of 25 articles: half (12-13) were human written and half (12-13) were model generated: the articles with human written completions in quiz A had model generated completions in quiz B and vice versa. The order of quiz question was shuffled for each participant. Participants could leave comments and were asked to indicate if they had seen the articles before. Participants were instructed not to look up the articles or their content during the quiz and at the end of the quiz were asked if they had looked anything up during the quiz.

Statistical Tests: To compare means on the different runs, we performed a two-sample t-test for independent groups for each model against the control. This was implemented in Python using the `scipy.stats.ttest_ind` function. When plotting a regression line in the graph of average participant accuracy vs model size, we fit a power law of the form ax^{-b}. The 95% confidence intervals were estimated from the t-distribution of the sample mean.

Duration statistics: In the main text, we discussed the finding that the ability of human participants to distinguish model and human generated news articles decreases as our models become larger. We have also found that the average time spent for a given set of questions increases as the model size increases, as shown in Figure E.1. Lower

Model	Participants Recruited	Participants Excluded	Genders (m:f:other)	Mean Age	Average Word Count (human:model)
Control	79	17	32:37:0	39	569:464
GPT-3 175B	81	19	32:30:0	40	569:498

Table E.2: Participant details and article lengths for the experiments investigating human detection of ~ 500 word model generated news articles. Participants were excluded due to internet check fails.

accuracy scores despite increased time investment from participants supports the finding that larger models generate harder-to-distinguish news articles.

Preliminary investigation of ~ 500 word articles: We recruited 160 unique US-based participants to take part in 2 experiments through Positly (details are given in Table E.2). We randomly selected 12 Reuters world news articles from late 2019 and created a context for GPT-3 175B that consisted of a single Reuters article not in this set of 12. We then used the article titles and Reuters locations to generate completions from GPT-3 175B and the 160M control model from the previous experiments. These were used to create two 12-question quizzes per model, each consisting of half human written and half model generated articles. Comprehension questions were added and articles were shown to participants in 3 stages at 30 second intervals to encourage closer reading. Participants were paid $12 for this task. Model generation selection methods, exclusion criteria, and statistical tests mirror those of the previous experiments.

F Additional Samples from GPT-3

GPT-3 adapts well to many tasks other than the ones explored in the main body of the paper. As an example, in Figure F.1, we show four uncurated samples from a prompt suggesting that the model write a poem, with a given title, in the style of Wallace Stevens. We first experimented with a few prompts, then generated four samples with no additional editing or selection (sampling at temperature 1 using nucleus sampling [HBFC19] with $P = 0.9$). Completions were truncated when the model began to write a new title and author heading, or broke into prose commentary.

Context →	The City BY C. P. CAVAFY TRANSLATED BY EDMUND KEELEY [Poem text omitted] SOME TREES John Ashbery [Poem text omitted] Shadows on the Way Wallace Stevens
-------- Generated Poem 1 -------- I must have shadows on the way If I am to walk I must have Each step taken slowly and alone To have it ready made And I must think in lines of grey To have dim thoughts to be my guide Must look on blue and green And never let my eye forget That color is my friend And purple must surround me too The yellow of the sun is no more Intrusive than the bluish snow That falls on all of us. I must have Grey thoughts and blue thoughts walk with me If I am to go away at all. -------- Generated Poem 2 -------- There is a way in the world. That way a man may live in a small town His lifetime and see no man come, Never the stir of men about a market, Never the bobbing of covered carts, Never the creak passed on the stair Closing some other life; an old man Can prosper in that town, can prosper When his store is seen to close, his wife Fades in his house, and the world shouts in vain That no one comes to hear him Where his small bell tolls. In that small town Everything is known. The ways of the young men are known, And the ways of the young women As they walk to the Spring; The rumors are all known, The quarrels, the hurt, the envy Of the men and the women. Till a day comes That way out of the world That leads to a wood. Young, the young man Walks, a lover now, Knows the streets as they are and is of them. Comes to the wood. There, in the wood, among trees, He sees shadows on the way, hears voices, hears the wind and the rustling of leaves; Through an open glade He sees a shape and the shape hears: It waits as he waits, Waits as the shadows wait, As the voices wait; Shadows on the way, voices in the wind.	-------- Generated Poem 3 -------- The sun was all we had. Now, in the shade All is changed. The mind must dwell on those White fields, that to its eyes were always old; Those ancient gleams, convoluting The way they lay among the huge roots, The great dark tomes of reverie, The plumed labyrinth of the sea. When our own breath gonne faint and fade Up the long stairway of the dead We shall remember well The blindness of the light, the whiteness Of that white land. The footsteps, and The grasses in that light, and in the shade The menacing life. We played, too, once, in the warmest rooms. To one content, there is one content For all mankind. The forest glades Are all the more for being in fable, And wood and town in legend shadowed- Once more to return. The one content Has in it every happiness, more brave For being purposeless; and so Those island gardens, deep in the mind, Make of remoteness familiarity And of the unspoken name a friend. -------- Generated Poem 4 -------- Nobody will come to this place. It is a road that leads nowhere. The solitude is deep. The mountains are high. But they are desolate, and they turn the traveler's face Towards the North. All the sounds of the world are far away. When the wind rises above the trees, The boughs bow to the ground. Even the birds that inhabit the tangle of weeds That is the roadside cover, are silent. One listens, But hears no roar of the forest. One is alone. One will be taken. One will be taken. There is no utterance, there is no conversation, But one is uneasy all the same.... There is a thin blue mist, A darkness rising like smoke, And within that darkness A possession of the heart. One will be taken.... It was here, and it will be here again- Here, under this sky empty and full of light.

Figure F.1: Four uncurated completions from a context suggesting the model compose a poem in the style of Wallace Stevens with the title 'Shadows on the Way'.

G Details of Task Phrasing and Specifications

The following figures illustrate the formatting and phrasing of all the tasks included in the paper. All data comes from the ground truth datasets in this section, and no samples from GPT-3 are included here.

Context →	Article: Informal conversation is an important part of any business relationship.Before you start a discussion,however,make sure you understand which topics are suitable and which are considered taboo in a particular culture. Latin Americans enjoy sharing information about their local history, art and customs.You may expect questions about your family,and be sure to show pictures of your children.You may feel free to ask similar questions of your Latin American friends.The French think of conversation as an art form,and they enjoy the value of lively discussions as well as disagreements. For them,arguments can be interesting and they can cover pretty much or any topic ---- as long as they occur in are respectful and intelligent manner. In the United States,business people like to discuss a wide range of topics,including opinions about work,family,hobbies,and politics. In Japan,China,and Korea,however,people are much more private.They do not share much about their thoughts,feelings,or emotions because they feel that doing so might take away from the harmonious business relationship they're trying to build.Middle Easterners are also private about their personal lives and family matters.It is considered rude,for example,to ask a businessman from Saudi Arabia about his wife or children. As a general rule,it's best not to talk about politics or religion with your business friends.This can get you into trouble,even in the United States,where people hold different religious views.In addition,discussing one's salary is usually considered unsuitable.Sports is typically a friendly subject in most parts of the world,although be careful not to criticize national sport.Instead,be friendly and praise your host's team. Q: What shouldn't you do when talking about sports with colleagues from another country? A: Criticizing the sports of your colleagues' country. Q: Which is typically a friendly topic in most places according to the author? A: Sports. Q: Why are people from Asia more private in their conversation with others? A: They don't want to have their good relationship with others harmed by informal conversation. Q: The author considers politics and religion _ . A:
Correct Answer →	taboo
Incorrect Answer →	cheerful topics
Incorrect Answer →	rude topics
Incorrect Answer →	topics that can never be talked about

Figure G.1: Formatted dataset example for RACE-h. When predicting, we normalize by the unconditional probability of each answer as described in 2.

Context →	anli 2: anli 2: The Gold Coast Hotel & Casino is a hotel and casino located in Paradise, Nevada. This locals' casino is owned and operated by Boyd Gaming. The Gold Coast is located one mile (∼ 1.6km) west of the Las Vegas Strip on West Flamingo Road. It is located across the street from the Palms Casino Resort and the Rio All Suite Hotel and Casino. Question: The Gold Coast is a budget-friendly casino. True, False, or Neither?
Correct Answer →	Neither
Incorrect Answer →	True
Incorrect Answer →	False

Figure G.2: Formatted dataset example for ANLI R2

Context →	Article: Mrs. Smith is an unusual teacher. Once she told each student to bring along a few potatoes in plastic bag. On each potato the students had to write a name of a person that they hated And the next day, every child brought some potatoes. Some had two potatoes;some three;some up to five. Mrs. Smith then told the children to carry the bags everywhere they went, even to the toilet, for two weeks. As day after day passed, the children started to complain about the awful smell of the rotten potatoes. Those children who brought five potatoes began to feel the weight trouble of the bags. After two weeks, the children were happy to hear that the game was finally ended. Mrs. Smith asked,"How did you feel while carrying the potatoes for two weeks?" The children started complaining about the trouble loudly. Then Mrs. Smith told them why she asked them to play the game. She said,"This is exactly the situation when you carry your hatred for somebody inside your heart. The terrible smell of the hatred will pollute your heart and you will carry something unnecessary with you all the time. If you cannot stand the smell of the rotten potatoes for just two weeks, can you imagine how heavy it would be to have the hatred in your heart for your lifetime? So throw away any hatred from your heart, and you'll be really happy." Q: Which of the following is True according to the passage? A: If a kid hated four people,he or she had to carry four potatoes. Q: We can learn from the passage that we should _ . A: throw away the hatred inside Q: The children complained about _ besides the weight trouble. A: the smell Q: Mrs.Smith asked her students to write _ on the potatoes. A:
Correct Answer →	names
Incorrect Answer →	numbers
Incorrect Answer →	time
Incorrect Answer →	places

Figure G.3: Formatted dataset example for RACE-m. When predicting, we normalize by the unconditional probability of each answer as described in 2.

Context →	How to apply sealant to wood.
Correct Answer →	Using a brush, brush on sealant onto wood until it is fully saturated with the sealant.
Incorrect Answer →	Using a brush, drip on sealant onto wood until it is fully saturated with the sealant.

Figure G.4: Formatted dataset example for PIQA

Context →	My body cast a shadow over the grass because
Correct Answer →	the sun was rising.
Incorrect Answer →	the grass was cut.

Figure G.5: Formatted dataset example for COPA

Context →	(CNN) Yuval Rabin, whose father, Yitzhak Rabin, was assassinated while serving as Prime Minister of Israel, criticized Donald Trump for appealing to "Second Amendment people" in a speech and warned that the words that politicians use can incite violence and undermine democracy. "Trump's words are an incitement to the type of political violence that touched me personally," Rabin wrote in USAToday. He said that Trump's appeal to "Second Amendment people" to stop Hillary Clinton -- comments that were criticized as a call for violence against Clinton, something Trump denied -- "were a new level of ugliness in an ugly campaign season." - The son of a former Israeli Prime Minister who was assassinated wrote an op ed about the consequence of violent political rhetoric. - Warns of "parallels" between Israel of the 1990s and the U.S. today.
Correct Answer →	- Referencing his father, who was shot and killed by an extremist amid political tension in Israel in 1995, Rabin condemned Donald Trump's aggressive rhetoric.
Correct Answer →	- Referencing his father, who was shot and killed by an extremist amid political tension in Israel in 1995, Rabin condemned Trump's aggressive rhetoric.
Incorrect Answer →	- Referencing his father, who was shot and killed by an extremist amid political tension in Israel in 1995, Rabin condemned Hillary Clinton's aggressive rhetoric.
Incorrect Answer →	- Referencing his father, who was shot and killed by an extremist amid political tension in Israel in 1995, Rabin condemned U.S.'s aggressive rhetoric.
Incorrect Answer →	- Referencing his father, who was shot and killed by an extremist amid political tension in Israel in 1995, Rabin condemned Yitzhak Rabin's aggressive rhetoric.

Figure G.6: Formatted dataset example for ReCoRD. We consider the context above to be a single "problem" because this is how the task is presented in the ReCoRD dataset and scored in the ReCoRD evaluation script.

Context →	anli 1: anli 1: Fulton James MacGregor MSP is a Scottish politician who is a Scottish National Party (SNP) Member of Scottish Parliament for the constituency of Coatbridge and Chryston. MacGregor is currently Parliamentary Liaison Officer to Shona Robison, Cabinet Secretary for Health & Sport. He also serves on the Justice and Education & Skills committees in the Scottish Parliament. Question: Fulton James MacGregor is a Scottish politican who is a Liaison officer to Shona Robison who he swears is his best friend. True, False, or Neither?
Correct Answer →	Neither
Incorrect Answer →	True
Incorrect Answer →	False

Figure G.7: Formatted dataset example for ANLI R1

Context →	Organisms require energy in order to do what?
Correct Answer →	mature and develop.
Incorrect Answer →	rest soundly.
Incorrect Answer →	absorb light.
Incorrect Answer →	take in nutrients.

Figure G.8: Formatted dataset example for OpenBookQA. When predicting, we normalize by the unconditional probability of each answer as described in 2.

Context →	Making a cake: Several cake pops are shown on a display. A woman and girl are shown making the cake pops in a kitchen. They
Correct Answer →	bake them, then frost and decorate.
Incorrect Answer →	taste them as they place them on plates.
Incorrect Answer →	put the frosting on the cake as they pan it.
Incorrect Answer →	come out and begin decorating the cake as well.

Figure G.9: Formatted dataset example for HellaSwag

Context →	anli 3: anli 3: We shut the loophole which has American workers actually subsidizing the loss of their own job. They just passed an expansion of that loophole in the last few days: $43 billion of giveaways, including favors to the oil and gas industry and the people importing ceiling fans from China. Question: The loophole is now gone True, False, or Neither?
Correct Answer →	False
Incorrect Answer →	True
Incorrect Answer →	Neither

Figure G.10: Formatted dataset example for ANLI R3

Context →	Question: George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? Answer:
Correct Answer →	dry palms
Incorrect Answer →	wet palms
Incorrect Answer →	palms covered with oil
Incorrect Answer →	palms covered with lotion

Figure G.11: Formatted dataset example for ARC (Challenge). When predicting, we normalize by the unconditional probability of each answer as described in 2.

Context →	lull is to trust as
Correct Answer →	cajole is to compliance
Incorrect Answer →	balk is to fortitude
Incorrect Answer →	betray is to loyalty
Incorrect Answer →	hinder is to destination
Incorrect Answer →	soothe is to passion

Figure G.12: Formatted dataset example for SAT Analogies

Correct Context →	Grace was happy to trade me her sweater for my jacket. She thinks the sweater
Incorrect Context →	Grace was happy to trade me her sweater for my jacket. She thinks the jacket
Target Completion →	looks dowdy on her.

Figure G.13: Formatted dataset example for Winograd. The 'partial' evaluation method we use compares the probability of the completion given a correct and incorrect context.

Correct Context →	Johnny likes fruits more than vegetables in his new keto diet because the fruits
Incorrect Context →	Johnny likes fruits more than vegetables in his new keto diet because the vegetables
Target Completion →	are saccharine.

Figure G.14: Formatted dataset example for Winogrande. The 'partial' evaluation method we use compares the probability of the completion given a correct and incorrect context.

Context →	READING COMPREHENSION ANSWER KEY While this process moved along, diplomacy continued its rounds. Direct pressure on the Taliban had proved unsuccessful. As one NSC staff note put it, "Under the Taliban, Afghanistan is not so much a state sponsor of terrorism as it is a state sponsored by terrorists." In early 2000, the United States began a high-level effort to persuade Pakistan to use its influence over the Taliban. In January 2000, Assistant Secretary of State Karl Inderfurth and the State Department's counterterrorism coordinator, Michael Sheehan, met with General Musharraf in Islamabad, dangling before him the possibility of a presidential visit in March as a reward for Pakistani cooperation. Such a visit was coveted by Musharraf, partly as a sign of his government's legitimacy. He told the two envoys that he would meet with Mullah Omar and press him on Bin Laden. They left, however, reporting to Washington that Pakistan was unlikely in fact to do anything," given what it sees as the benefits of Taliban control of Afghanistan." President Clinton was scheduled to travel to India. The State Department felt that he should not visit India without also visiting Pakistan. The Secret Service and the CIA, however, warned in the strongest terms that visiting Pakistan would risk the President's life. Counterterrorism officials also argued that Pakistan had not done enough to merit a presidential visit. But President Clinton insisted on including Pakistan in the itinerary for his trip to South Asia. His one-day stopover on March 25, 2000, was the first time a U.S. president had been there since 1969. At his meeting with Musharraf and others, President Clinton concentrated on tensions between Pakistan and India and the dangers of nuclear proliferation, but also discussed Bin Laden. President Clinton told us that when he pulled Musharraf aside for a brief, one-on-one meeting, he pleaded with the general for help regarding Bin Laden." I offered him the moon when I went to see him, in terms of better relations with the United States, if he'd help us get Bin Laden and deal with another issue or two." The U.S. effort continued. Who did The State Department feel should visit both India and Pakistan?
Correct Answer →	- [False] Bin Laden
Incorrect Answer →	- [True] Bin Laden

Figure G.15: Formatted dataset example for MultiRC. There are three levels within MultiRC: (1) the passage, (2) the questions, and (3) the answers. During evaluation, accuracy is determined at the per-question level, with a question being considered correct if and only if all the answers within the question are labeled correctly. For this reason, we use K to refer to the number of **questions** shown within the context.

Context →	Question: Which factor will most likely cause a person to develop a fever? Answer:
Correct Answer →	a bacterial population in the bloodstream
Incorrect Answer →	a leg muscle relaxing after exercise
Incorrect Answer →	several viral particles on the skin
Incorrect Answer →	carbohydrates being digested in the stomach

Figure G.16: Formatted dataset example for ARC (Easy). When predicting, we normalize by the unconditional probability of each answer as described in 2.

Context →	Bob went to the gas station to fill up his car. His tank was completely empty and so was his wallet. The cashier offered to pay for his gas if he came back later to pay. Bob felt grateful as he drove home.
Correct Answer →	Bob believed that there were good people in the world.
Incorrect Answer →	Bob contemplated how unfriendly the world was.

Figure G.17: Formatted dataset example for StoryCloze

Context →	Helsinki is the capital and largest city of Finland. It is in the region of Uusimaa, in southern Finland, on the shore of the Gulf of Finland. Helsinki has a population of , an urban population of , and a metropolitan population of over 1.4 million, making it the most populous municipality and urban area in Finland. Helsinki is some north of Tallinn, Estonia, east of Stockholm, Sweden, and west of Saint Petersburg, Russia. Helsinki has close historical connections with these three cities. The Helsinki metropolitan area includes the urban core of Helsinki, Espoo, Vantaa, Kauniainen, and surrounding commuter towns. It is the world's northernmost metro area of over one million people, and the city is the northernmost capital of an EU member state. The Helsinki metropolitan area is the third largest metropolitan area in the Nordic countries after Stockholm and Copenhagen, and the City of Helsinki is the third largest after Stockholm and Oslo. Helsinki is Finland's major political, educational, financial, cultural, and research center as well as one of northern Europe's major cities. Approximately 75% of foreign companies that operate in Finland have settled in the Helsinki region. The nearby municipality of Vantaa is the location of Helsinki Airport, with frequent service to various destinations in Europe and Asia. Q: what is the most populous municipality in Finland? A: Helsinki Q: how many people live there? A: 1.4 million in the metropolitan area Q: what percent of the foreign companies that operate in Finland are in Helsinki? A: 75% Q: what towns are a part of the metropolitan area? A:
Target Completion →	Helsinki, Espoo, Vantaa, Kauniainen, and surrounding commuter towns

Figure G.18: Formatted dataset example for CoQA

Context →	Please unscramble the letters into a word, and write that word: asinoc =
Target Completion →	casino

Figure G.19: Formatted dataset example for Cycled Letters

Context →	Passage: Saint Jean de Brébeuf was a French Jesuit missionary who travelled to New France in 1625. There he worked primarily with the Huron for the rest of his life, except for a few years in France from 1629 to 1633. He learned their language and culture, writing extensively about each to aid other missionaries. In 1649, Brébeuf and another missionary were captured when an Iroquois raid took over a Huron village . Together with Huron captives, the missionaries were ritually tortured and killed on March 16, 1649. Brébeuf was beatified in 1925 and among eight Jesuit missionaries canonized as saints in the Roman Catholic Church in 1930. Question: How many years did Saint Jean de Brébeuf stay in New France before he went back to France for a few years? Answer:
Target Completion →	4

Figure G.20: Formatted dataset example for DROP

Context →	Fill in blank: She held the torch in front of her. She caught her breath. "Chris? There's a step." "What?" "A step. Cut in the rock. About fifty feet ahead." She moved faster. They both moved faster. "In fact," she said, raising the torch higher, "there's more than a ____. ->
Target Completion →	step

Figure G.21: Formatted dataset example for LAMBADA

Context →	Please unscramble the letters into a word, and write that word: skicts =
Target Completion →	sticks

Figure G.22: Formatted dataset example for Anagrams 1 (A1)

Context →	Please unscramble the letters into a word, and write that word: volwskagen =
Target Completion →	volkswagen

Figure G.23: Formatted dataset example for Anagrams 2

Context →	Q: Who played tess on touched by an angel? A:
Target Completion →	Delloreese Patricia Early (July 6, 1931 { November 19, 2017), known professionally as Della Reese

Figure G.24: Formatted dataset example for Natural Questions

Context →	TITLE: William Perry (American football) - Professional career PARAGRAPH: In 1985, he was selected in the first round of the 1985 NFL Draft by the Chicago Bears; he had been hand-picked by coach Mike Ditka. However, defensive coordinator Buddy Ryan, who had a highly acrimonious relationship with Ditka, called Perry a "wasted draft-pick". Perry soon became a pawn in the political power struggle between Ditka and Ryan. Perry's "Refrigerator" nickname followed him into the NFL and he quickly became a favorite of the Chicago Bears fans. Teammates called him "Biscuit," as in "one biscuit shy of 350 pounds." While Ryan refused to play Perry, Ditka decided to use Perry as a fullback when the team was near the opponents' goal line or in fourth and short situations, either as a ball carrier or a lead blocker for star running back Walter Payton. Ditka stated the inspiration for using Perry as a fullback came to him during five-yard sprint exercises. During his rookie season, Perry rushed for two touchdowns and caught a pass for one. Perry even had the opportunity to run the ball during Super Bowl XX, as a nod to his popularity and contributions to the team's success. The first time he got the ball, he was tackled for a one-yard loss while attempting to throw his first NFL pass on a halfback option play. The second time he got the ball, he scored a touchdown (running over Patriots linebacker Larry McGrew in the process). About halfway through his rookie season, Ryan finally began to play Perry, who soon proved that he was a capable defensive lineman. His Super Bowl ring size is the largest of any professional football player in the history of the event. His ring size is 25, while the ring size for the average adult male is between 10 and 12. Perry went on to play for ten years in the NFL, retiring after the 1994 season. In his ten years as a pro, he regularly struggled with his weight, which hampered his performance at times. He played in 138 games, recording 29.5 sacks and five fumble recoveries, which he returned for a total of 71 yards. In his offensive career he ran five yards for two touchdowns, and had one reception for another touchdown. Perry later attempted a comeback, playing an unremarkable 1996 season with the London Monarchs of the World League of American Football (later NFL Europa). Q: what team did he play for? A:
Target Completion →	the Chicago Bears

Figure G.25: Formatted dataset example for QuAC

Context →	Please unscramble the letters into a word, and write that word: r e!c.i p r o.c a/l =
Target Completion →	reciprocal

Figure G.26: Formatted dataset example for Symbol Insertion

Context →	Please unscramble the letters into a word, and write that word: taefed =
Target Completion →	defeat

Figure G.27: Formatted dataset example for Reversed Words

Context →	Title: The_Blitz Background: From the German point of view, March 1941 saw an improvement. The Luftwaffe flew 4,000 sorties that month, including 12 major and three heavy attacks. The electronic war intensified but the Luftwaffe flew major inland missions only on moonlit nights. Ports were easier to find and made better targets. To confuse the British, radio silence was observed until the bombs fell. X- and Y-Gerät beams were placed over false targets and switched only at the last minute. Rapid frequency changes were introduced for X-Gerät, whose wider band of frequencies and greater tactical flexibility ensured it remained effective at a time when British selective jamming was degrading the effectiveness of Y-Gerät. Q: How many sorties were flown in March 1941? A: 4,000 Q: When did the Luftwaffe fly inland missions? A:
Target Completion →	only on moonlit nights

Figure G.28: Formatted dataset example for SQuADv2

Context →	Normal force -- In a simple case such as an object resting upon a table, the normal force on the object is equal but in opposite direction to the gravitational force applied on the object (or the weight of the object), that is, N = m g (\displaystyle N=mg), where m is mass, and g is the gravitational field strength (about 9.81 m/s on Earth). The normal force here represents the force applied by the table against the object that prevents it from sinking through the table and requires that the table is sturdy enough to deliver this normal force without breaking. However, it is easy to assume that the normal force and weight are action-reaction force pairs (a common mistake). In this case, the normal force and weight need to be equal in magnitude to explain why there is no upward acceleration of the object. For example, a ball that bounces upwards accelerates upwards because the normal force acting on the ball is larger in magnitude than the weight of the ball. question: is the normal force equal to the force of gravity? answer:
Target Completion →	yes

Figure G.29: Formatted dataset example for BoolQ

Context →	The trend toward lower rents may seem surprising given that some communities in New York are bemoaning the loss of favorite local businesses to high rents. But, despite the recent softening, for many of these retailers there's still been too big a jump from the rental rates of the late 1970s, when their leases were signed. Certainly, the recent drop in prices doesn't mean Manhattan comes cheap. question: Manhattan comes cheap. true, false, or neither? answer:
Target Completion →	false

Figure G.30: Formatted dataset example for CB

Context →	The bet, which won him dinner for four, was regarding the existence and mass of the top quark, an elementary particle discovered in 1995. question: The Top Quark is the last of six flavors of quarks predicted by the standard model theory of particle physics. True or False? answer:
Target Completion →	False

Figure G.31: Formatted dataset example for RTE

Context →	An outfitter provided everything needed for the safari. Before his first walking holiday, he went to a specialist outfitter to buy some boots. question: Is the word 'outfitter' used in the same way in the two sentences above? answer:
Target Completion →	no

Figure G.32: Formatted dataset example for WiC

Context →	Final Exam with Answer Key Instructions: Please carefully read the following passages. For each passage, you must identify which noun the pronoun marked in *bold* refers to. ===== Passage: Mr. Moncrieff visited Chester's luxurious New York apartment, thinking that it belonged to his son Edward. The result was that Mr. Moncrieff has decided to cancel Edward's allowance on the ground that he no longer requires *his* financial support. Question: In the passage above, what does the pronoun "*his*" refer to? Answer:
Target Completion →	mr. moncrieff

Figure G.33: Formatted dataset example for WSC

Context →	Q: 'Nude Descending A Staircase' is perhaps the most famous painting by which 20th century artist? A:
Target Completion →	MARCEL DUCHAMP
Target Completion →	r mutt
Target Completion →	duchamp
Target Completion →	marcel duchamp
Target Completion →	R.Mutt
Target Completion →	Marcel duChamp
Target Completion →	Henri-Robert-Marcel Duchamp
Target Completion →	Marcel du Champ
Target Completion →	henri robert marcel duchamp
Target Completion →	Duchampian
Target Completion →	Duchamp
Target Completion →	duchampian
Target Completion →	marcel du champ
Target Completion →	Marcel Duchamp
Target Completion →	MARCEL DUCHAMP

Figure G.34: Formatted dataset example for TriviaQA. TriviaQA allows for multiple valid completions.

Context →	Q: What school did burne hogarth establish? A:
Target Completion →	School of Visual Arts

Figure G.35: Formatted dataset example for WebQA

Context →	Keinesfalls dürfen diese für den kommerziellen Gebrauch verwendet werden. =
Target Completion →	In no case may they be used for commercial purposes.

Figure G.36: Formatted dataset example for De→En. This is the format for one- and few-shot learning, for this and other langauge tasks, the format for zero-shot learning is "Q: What is the {language} translation of {sentence} A: {translation}."

Context →	In no case may they be used for commercial purposes. =
Target Completion →	Keinesfalls dürfen diese für den kommerziellen Gebrauch verwendet werden.

Figure G.37: Formatted dataset example for En→De

Context →	Analysis of instar distributions of larval I. verticalis collected from a series of ponds also indicated that males were in more advanced instars than females. =
Target Completion →	L'analyse de la distribution de fréquence des stades larvaires d'I. verticalis dans une série d'étangs a également démontré que les larves mâles étaient à des stades plus avancés que les larves femelles.

Figure G.38: Formatted dataset example for En→Fr

Context →	L'analyse de la distribution de fréquence des stades larvaires d'I. verticalis dans une série d'étangs a également démontré que les larves mâles étaient à des stades plus avancés que les larves femelles. =
Target Completion →	Analysis of instar distributions of larval I. verticalis collected from a series of ponds also indicated that males were in more advanced instars than females.

Figure G.39: Formatted dataset example for Fr→En

Context →	The truth is that you want, at any price, and against the wishes of the peoples of Europe, to continue the negotiations for Turkey's accession to the European Union, despite Turkey's continuing refusal to recognise Cyprus and despite the fact that the democratic reforms are at a standstill. =
Target Completion →	Adevărul este că vă doriţi, cu orice preţ şi împotriva dorinţei europenilor, să continuaţi negocierile de aderare a Turciei la Uniunea Europeană, în ciuda refuzului continuu al Turciei de a recunoaşte Ciprul şi în ciuda faptului că reformele democratice au ajuns într-un punct mort.

Figure G.40: Formatted dataset example for En→Ro

Context →	Adevărul este că vă doriţi, cu orice preţ şi împotriva dorinţei europenilor, să continuaţi negocierile de aderare a Turciei la Uniunea Europeană, în ciuda refuzului continuu al Turciei de a recunoaşte Ciprul şi în ciuda faptului că reformele democratice au ajuns într-un punct mort. =
Target Completion →	The truth is that you want, at any price, and against the wishes of the peoples of Europe, to continue the negotiations for Turkey's accession to the European Union, despite Turkey's continuing refusal to recognise Cyprus and despite the fact that the democratic reforms are at a standstill.

Figure G.41: Formatted dataset example for Ro→En

Context →	Q: What is (2 * 4) * 6? A:
Target Completion →	48

Figure G.42: Formatted dataset example for Arithmetic 1DC

Context →	Q: What is 17 minus 14? A:
Target Completion →	3

Figure G.43: Formatted dataset example for Arithmetic 2D-

Context →	Q: What is 98 plus 45? A:
Target Completion →	143

Figure G.44: Formatted dataset example for Arithmetic 2D+

Context →	Q: What is 95 times 45? A:
Target Completion →	4275

Figure G.45: Formatted dataset example for Arithmetic 2Dx

Context →	Q: What is 509 minus 488? A:
Target Completion →	21

Figure G.46: Formatted dataset example for Arithmetic 3D-

Context →	Q: What is 556 plus 497? A:
Target Completion →	1053

Figure G.47: Formatted dataset example for Arithmetic 3D+

Context →	Q: What is 6209 minus 3365? A:
Target Completion →	2844

Figure G.48: Formatted dataset example for Arithmetic 4D-

Context →	Q: What is 9923 plus 617? A:
Target Completion →	10540

Figure G.49: Formatted dataset example for Arithmetic 4D+

Context →	Q: What is 40649 minus 78746? A:
Target Completion →	-38097

Figure G.50: Formatted dataset example for Arithmetic 5D−

Context →	Q: What is 65360 plus 16204? A:
Target Completion →	81564

Figure G.51: Formatted dataset example for Arithmetic 5D+

H Results on All Tasks for All Model Sizes

			Fine-tune		Zero-Shot								One-Shot								Few-Shot								
Name	Metric	Split	SOTA	K	Small	Med	Large	XL	2.7B	6.7B	13B	175B	Small	Med	Large	XL	2.7B	6.7B	13B	175B	Small	Med	Large	XL	2.7B	6.7B	13B	175B	175B (test server)
HellaSwag	acc	dev	85.6	20	33.7	43.6	51.0	54.7	62.8	67.4	70.9	78.9	33.0	42.9	50.5	53.5	61.9	66.5	70.0	78.1	33.5	43.1	51.3	54.9	62.9	67.3	71.3	79.3	
LAMBADA	acc	test	68.0	15	42.7	54.3	60.4	63.6	67.1	70.3	72.5	76.2	22.0	47.1	52.6	58.3	61.1	65.4	69.0	72.5	22.0	40.4	63.2	57.0	78.1	79.1	81.3	86.4	
LAMBADA	ppl	test	8.63	15	18.6	9.09	6.53	5.44	4.60	4.00	3.56	3.00	165.0	11.6	8.29	6.46	5.53	4.61	4.06	3.35	165.0	27.6	6.63	7.45	2.89	2.56	2.56	1.92	
StoryCloze	acc	test	91.8	70	63.3	68.5	72.4	73.4	77.2	77.7	79.5	83.2	62.3	68.7	72.3	74.2	77.3	78.7	79.7	84.7	62.3	70.2	73.9	76.1	80.2	81.2	83.0	87.7	
NQs	acc	test	44.5	64	0.64	1.75	2.71	4.40	6.01	5.79	7.84	14.6	1.19	3.07	4.79	5.43	8.73	9.78	13.7	23.0	1.72	4.46	7.89	9.72	13.2	17.0	21.0	29.9	
TriviaQA	acc	dev	68.0	64	4.15	7.61	14.0	19.7	31.3	38.7	41.8	64.3	4.19	12.9	20.5	26.5	35.9	44.4	51.3	68.0	6.96	16.3	26.5	32.1	42.3	51.6	57.5	71.2	71.2
WebQs	acc	test	45.5	64	1.77	3.20	4.33	4.63	7.92	7.73	8.22	14.4	2.56	6.20	8.51	9.15	14.5	15.1	19.0	25.3	5.46	12.6	15.9	19.6	24.8	27.7	33.5	41.5	
Ro→En 16	BLEU-mb	test	39.9	64	2.08	2.71	3.09	3.15	16.3	8.34	20.2	19.9	0.55	15.4	23.0	26.3	30.6	33.2	35.6	38.6	1.25	20.7	25.8	29.2	33.1	34.8	37.0	39.5	
Ro→En 16	BLEU-sb	test		64	2.39	3.08	3.49	3.56	16.8	8.75	20.8	20.9	0.65	15.9	23.6	26.8	31.3	34.2	36.7	40.0	1.40	21.3	26.6	30.1	34.3	36.2	38.4	41.3	
En→Ro 16	BLEU-mb	test	38.5	64	2.14	2.65	2.53	2.50	3.46	4.24	5.32	14.1	0.35	3.30	7.89	8.72	13.2	15.1	17.3	20.6	1.25	5.90	9.33	10.7	14.3	16.3	18.0	21.0	
En→Ro 16	BLEU-sb	test		64	2.61	3.11	3.07	3.09	4.26	5.31	6.43	18.0	0.55	3.90	9.15	10.3	15.7	18.2	20.8	24.9	1.64	7.40	10.9	12.9	17.2	19.6	21.8	25.8	
Fr→En 14	BLEU-mb	test	35.0	64	1.81	2.53	3.47	3.13	20.6	15.1	21.8	21.2	1.28	15.9	23.7	26.3	29.0	30.5	30.2	33.7	4.98	25.5	28.5	31.1	33.7	34.9	36.6	39.2	
Fr→En 14	BLEU-sb	test		64	2.29	2.99	3.90	3.60	21.2	15.5	22.4	21.9	1.50	16.3	24.4	27.0	30.0	31.6	31.4	35.6	5.30	26.2	29.5	32.2	35.1	36.4	38.3	41.4	
En→Fr 14	BLEU-mb	test	45.6	64	1.74	2.16	2.73	2.15	15.1	8.82	12.0	25.2	0.49	8.00	14.8	15.9	20.3	23.3	24.9	28.3	4.08	14.5	19.3	21.5	24.9	27.3	29.5	32.6	
En→Fr 14	BLEU-sb	test	45.9	64	2.44	2.75	3.54	2.82	19.3	11.4	15.3	31.3	0.81	10.0	18.2	19.3	24.7	28.3	30.1	34.1	5.31	18.0	23.6	26.1	30.3	33.3	35.5	39.9	
De→En 16	BLEU-mb	test	40.2	64	2.06	2.87	3.41	3.63	21.5	17.3	23.0	27.2	0.83	16.2	22.5	24.7	28.2	30.7	33.0	30.4	3.25	22.7	26.2	29.2	32.7	34.8	37.3	40.6	
De→En 16	BLEU-sb	test		64	2.39	3.27	3.85	4.04	22.5	18.2	24.4	28.6	0.93	17.1	23.4	25.8	29.2	31.9	34.5	32.1	3.60	23.8	27.5	30.5	34.1	36.5	39.1	43.0	
En→De 16	BLEU-mb	test	41.2	64	1.70	2.27	2.31	2.43	12.9	8.66	10.4	24.6	0.50	7.00	12.9	13.1	18.3	20.9	22.5	26.2	3.42	12.3	15.4	17.1	20.9	23.0	26.6	29.7	
En→De 16	BLEU-sb	test	41.2	64	2.09	2.65	2.75	2.92	13.7	9.36	11.0	25.3	0.54	7.40	13.4	13.4	18.8	21.7	23.3	27.3	3.78	12.9	16.1	17.7	21.7	24.1	27.7	30.9	
Winograd	acc	test	93.8	7	66.3	72.9	74.7	76.9	82.4	85.7	87.9	88.3	63.4	68.5	72.9	76.9	82.4	84.6	86.1	89.7	63.4	67.4	73.6	76.9	84.3	85.4	82.4	88.6	
Winogrande	acc	dev	84.6	50	52.0	52.1	57.4	58.7	62.3	64.5	67.9	70.2	51.3	53.0	58.3	59.1	61.7	65.8	66.9	73.2	51.3	52.6	57.5	59.1	62.6	67.4	70.0	77.7	
PIQA	acc	dev	77.1	50	64.6	70.2	72.9	75.1	75.6	78.0	78.5	81.0	64.3	69.3	71.8	74.4	74.3	76.3	77.8	80.5	64.3	69.4	72.0	74.3	75.4	77.8	79.9	82.3	82.8
ARC (Challenge)	acc	test	78.5	50	26.6	29.5	31.8	35.5	38.0	41.4	43.7	51.4	25.5	30.2	31.6	36.4	38.4	41.5	43.1	53.2	25.5	28.4	32.3	36.7	39.5	43.7	44.8	51.5	
ARC (Easy)	acc	test	92.0	50	43.6	46.5	53.0	53.8	58.2	60.2	63.8	68.8	42.7	48.2	54.6	55.9	60.3	62.6	66.8	71.2	42.7	51.0	58.1	59.1	62.1	65.8	69.1	70.1	
OpenBookQA	acc	test	87.2	100	35.6	43.2	45.2	46.8	53.0	50.4	55.6	57.6	37.0	39.8	46.2	46.4	53.4	53.0	55.8	58.8	37.0	43.6	48.0	50.6	55.6	55.2	60.8	65.4	
Quac	f1	dev	74.4	5	21.2	26.8	31.0	30.1	34.7	36.1	38.4	41.5	21.1	26.9	31.9	32.3	37.4	39.0	40.6	43.4	21.6	27.6	32.9	34.2	38.2	39.9	40.9	44.3	
RACE-h	acc	test	90.0	10	35.2	37.9	40.1	40.9	42.4	44.1	44.6	45.5	34.3	37.7	40.0	42.0	43.8	44.3	44.6	45.9	34.3	37.0	40.4	41.4	42.3	44.7	45.1	46.8	
RACE-m	acc	test	93.1	10	42.1	47.2	52.1	52.3	54.7	54.4	56.7	58.4	42.3	47.3	51.7	55.2	56.1	54.7	56.9	57.4	42.3	47.0	52.7	53.0	55.6	55.4	58.1	58.1	
SQuADv2	em	dev	90.7	16	22.6	32.8	33.9	43.1	43.6	45.4	49.0	52.6	25.1	37.5	37.9	47.9	47.9	51.1	56.0	60.1	27.5	40.5	39.2	53.5	50.0	56.6	62.6	64.9	
SQuADv2	f1	dev	93.0	16	28.3	40.2	41.4	50.3	51.0	52.7	56.3	59.5	30.1	43.6	44.1	54.0	54.1	57.1	61.8	65.4	32.1	45.5	44.9	58.7	55.9	62.1	67.7	69.8	
CoQA	f1	dev	90.7	5	34.5	55.0	61.8	65.3	71.1	72.8	76.3	81.5	30.6	52.1	61.6	66.1	71.8	75.1	77.9	84.0	31.1	52.0	62.7	66.8	73.2	77.3	79.9	85.0	
DROP	f1	dev	89.1	20	9.40	13.6	14.4	16.4	19.7	17.0	24.0	23.6	11.7	18.1	20.9	23.0	26.4	27.3	29.2	34.3	12.9	18.7	24.0	25.6	29.7	29.7	32.3	36.5	
BoolQ	acc	dev	91.0	32	49.7	60.3	58.9	62.4	67.1	65.4	66.2	60.5	52.6	61.7	60.4	63.7	68.4	68.7	69.0	76.7	43.1	60.6	62.0	64.1	70.3	70.0	70.2	77.5	76.4
CB	acc	dev	96.9	32	0.00	32.1	8.93	19.6	19.6	28.6	19.6	46.4	55.4	53.6	53.6	48.2	57.1	33.9	55.4	64.3	42.9	58.9	53.6	69.6	67.9	60.7	66.1	82.1	75.6
CB	f1	dev	93.9	32	0.00	29.3	11.4	17.4	22.4	25.1	20.3	42.8	60.1	39.8	45.6	37.5	45.7	28.5	44.6	52.5	26.1	40.4	32.6	48.3	45.7	44.6	46.0	57.2	52.0
Copa	acc	dev	94.8	32	66.0	68.0	73.0	77.0	76.0	80.0	84.0	91.0	62.0	64.0	66.0	74.0	76.0	82.0	86.0	87.0	67.0	64.0	72.0	77.0	83.0	83.0	86.0	92.0	92.0
RTE	acc	dev	92.5	32	47.7	49.8	48.4	56.0	46.6	55.2	62.8	63.5	53.1	47.3	49.5	49.5	54.9	54.9	56.3	70.4	52.3	48.4	46.9	50.9	56.3	49.5	60.6	72.9	69.0
WiC	acc	dev	76.1	32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	50.0	50.3	50.3	49.2	49.4	50.3	50.0	48.6	49.8	55.0	53.0	53.0	51.6	53.1	51.1	55.3	49.4
WSC	acc	dev	93.8	32	59.6	56.7	65.4	61.5	66.3	60.6	64.4	65.4	58.7	58.7	60.6	62.5	66.3	60.6	66.3	69.2	58.7	60.6	54.8	49.0	62.5	67.3	75.0	75.0	80.1
MultiRC	acc	dev	62.3	32	4.72	9.65	12.3	13.6	14.3	18.4	24.2	27.6	4.72	9.65	12.3	13.6	14.3	18.4	24.2	27.6	6.09	11.8	16.8	20.8	24.7	23.8	25.0	32.5	30.5
MultiRC	f1a	dev	88.2	32	57.0	59.7	60.4	59.9	60.0	64.5	71.4	72.9	57.0	59.7	60.4	59.9	60.0	64.5	71.4	72.9	45.0	55.9	64.2	65.4	69.5	66.4	69.3	74.8	75.4
ReCoRD	acc	dev	92.5	32	70.8	78.5	82.1	84.1	86.2	88.6	89.0	90.2	69.8	77.0	80.7	83.0	85.9	88.0	88.8	90.2	69.8	77.2	81.3	83.1	86.6	87.9	88.9	89.0	90.2
ReCoRD	f1	dev	93.3	32	71.9	79.2	82.8	85.2	87.3	89.5	90.4	91.0	70.7	77.8	81.6	83.9	86.8	88.8	89.7	91.2	70.7	77.9	82.1	84.0	87.5	88.8	89.8	90.1	91.1
SuperGLUE	average	dev	89.0		40.6	47.4	46.8	49.6	50.1	52.3	54.4	58.2	54.4	55.1	56.7	57.8	61.2	59.7	64.3	68.9	50.2	56.2	56.8	60.0	64.3	63.6	66.9	73.2	71.8
ANLI R1	acc	test	73.8	50	33.4	34.2	33.4	33.4	34.2	32.3	33.2	34.6	32.1	31.6	31.9	34.6	30.6	31.6	32.7	32.0	32.1	32.5	30.9	32.5	33.5	33.1	33.3	36.8	
ANLI R2	acc	test	50.7	50	33.2	31.9	33.3	33.3	33.8	33.5	33.5	35.4	35.7	33.7	33.2	32.7	32.7	33.9	33.9	33.9	35.7	33.8	32.1	31.4	32.6	33.3	32.6	34.0	
ANLI R3	acc	test	48.3	50	33.6	34.0	33.8	33.4	35.3	34.8	34.4	34.5	35.0	32.6	33.0	33.9	34.1	33.1	32.5	35.1	35.0	34.4	35.1	36.0	32.7	33.9	34.5	40.2	
2D+	acc	n/a		50	0.70	0.65	0.70	0.85	1.10	2.54	15.4	76.9	2.00	0.55	3.15	4.00	12.1	19.6	73.0	99.6	2.00	4.10	3.50	4.50	8.90	11.9	55.5	100.0	
2D-	acc	n/a		50	1.25	1.25	1.25	1.25	1.60	7.60	12.6	58.0	1.15	0.95	1.45	1.95	3.85	11.5	44.6	86.4	1.15	1.45	2.25	2.70	7.35	13.6	52.4	98.9	
3D+	acc	n/a		50	0.10	0.10	0.05	0.10	0.10	0.25	1.40	34.2	0.15	0.00	0.10	0.30	0.45	0.95	15.4	65.5	0.15	0.45	0.30	0.55	0.75	0.90	8.40	80.4	
3D-	acc	n/a		50	0.05	0.05	0.05	0.05	0.05	0.45	1.35	48.3	0.05	0.15	0.25	0.30	0.55	1.60	6.15	78.7	0.05	0.10	0.15	0.35	0.65	1.05	9.20	94.2	
4D+	acc	n/a		50	0.05	0.05	0.00	0.00	0.05	0.05	0.15	4.00	0.00	0.00	0.10	0.00	0.00	0.10	0.80	14.0	0.00	0.05	0.05	0.00	0.15	0.15	0.40	25.5	
4D-	acc	n/a		50	0.00	0.00	0.00	0.00	0.00	0.00	0.10	7.50	0.00	0.00	0.00	0.00	0.05	0.00	0.50	14.0	0.00	0.05	0.00	0.00	0.10	0.05	0.40	26.8	
5D+	acc	n/a		50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.00	0.00	0.00	0.00	0.00	0.00	0.05	3.45	0.00	0.00	0.00	0.00	0.00	0.00	0.05	9.30	
5D-	acc	n/a		50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.05	3.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	9.90	
2Dx	acc	n/a		50	2.20	2.25	2.65	2.10	2.55	5.80	6.15	19.8	1.35	2.35	3.35	2.35	4.75	9.15	11.0	27.4	1.35	2.90	2.70	2.85	4.25	6.10	7.05	29.2	
1DC	acc	n/a		50	1.25	2.95	2.75	0.05	0.30	2.35	0.75	9.75	1.90	2.80	2.85	3.65	6.45	9.15	8.20	14.3	1.70	2.15	3.90	5.75	6.20	7.60	9.95	21.3	
Cycled Letters	acc	n/a		100	0.62	0.71	2.85	0.00	0.63	1.35	2.58	3.66	1.67	4.36	5.68	6.46	6.25	9.41	15.1	21.7	4.63	9.27	10.7	14.5	16.7	21.9	27.7	37.9	
Anagrams 1	acc	n/a		100	0.10	0.14	0.40	0.00	0.27	0.69	1.16	2.28	0.21	0.61	1.12	1.27	1.60	2.72	3.72	8.62	0.50	1.27	2.13	3.05	3.81	5.49	8.38	15.1	
Anagrams 2	acc	n/a		100	0.81	1.21	2.69	0.01	1.71	3.75	4.53	8.91	1.19	2.62	4.70	4.77	6.97	10.2	14.6	25.9	1.94	4.80	7.59	9.87	12.6	18.9	25.6	39.7	
Symbol Insertion	acc	n/a		100	0.00	0.00	0.10	0.00	0.05	0.42	0.89	8.26	0.03	0.05	0.57	1.18	1.67	3.46	6.62	45.4	0.11	0.28	2.19	4.18	6.61	11.0	27.3	67.2	
Reversed Words	acc	n/a		100	0.00	0.01	0.01	0.01	0.02	0.03	0.03	0.09	0.02	0.01	0.01	0.00	0.05	0.07	0.11	0.48	0.00	0.05	0.00	0.17	0.24	0.30	0.42	0.44	
SAT Analogies	acc	n/a		20	35.6	39.0	45.2	44.1	50.0	49.2	52.7	53.7	30.5	41.2	43.1	46.5	55.1	54.3	53.5	59.1	30.5	40.4	42.8	40.6	48.4	51.9	53.5	65.2	

Table H.1: Scores for every task, setting and model that we investigate in this paper.

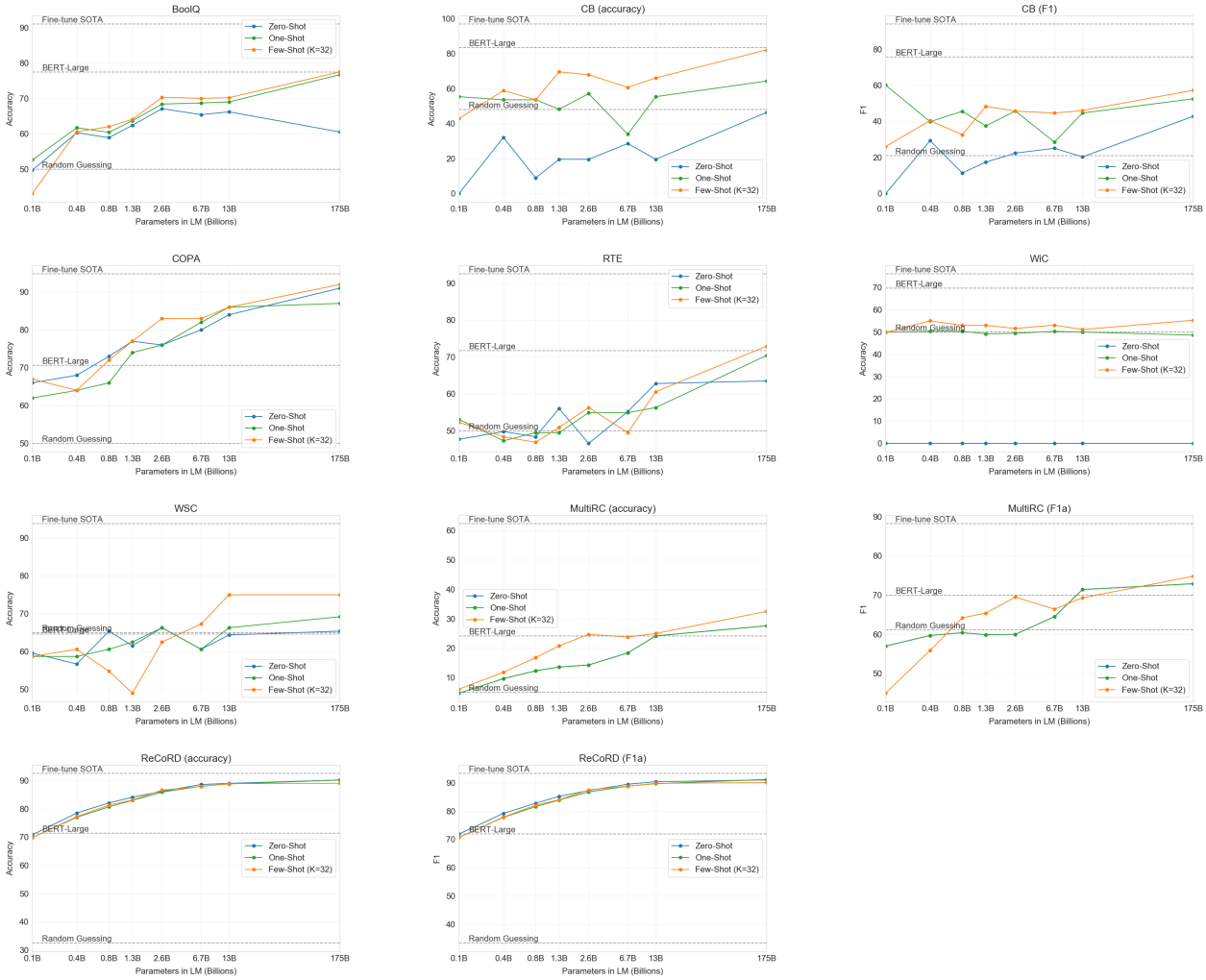

Figure H.1: All results for all SuperGLUE tasks.

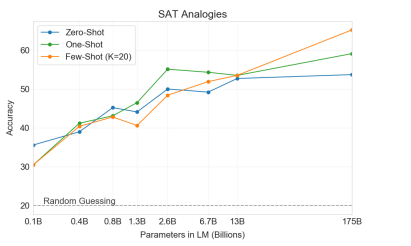

Figure H.2: Results for SAT task.

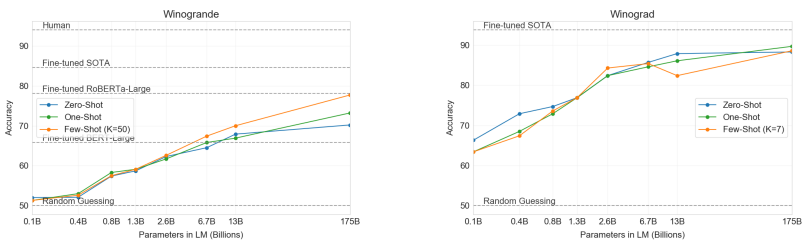

Figure H.3: All results for all Winograd tasks.

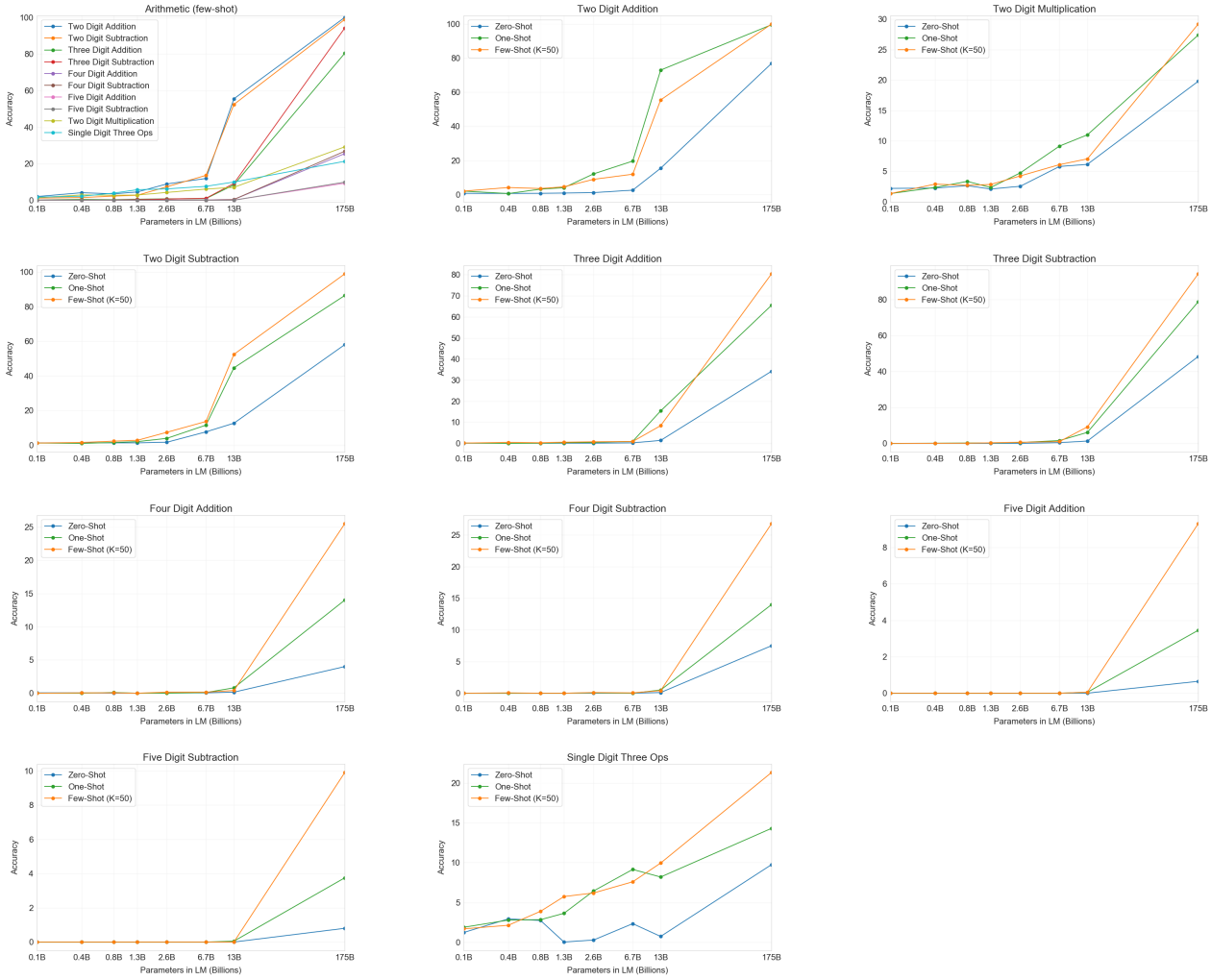

Figure H.4: All results for all Arithmetic tasks.

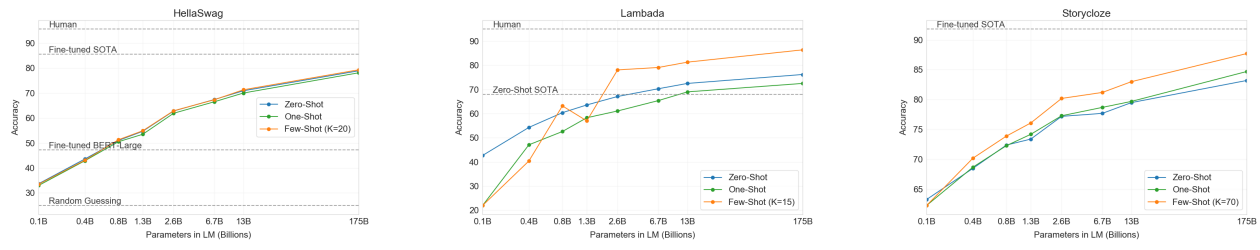

Figure H.5: All results for all Cloze and Completion tasks.

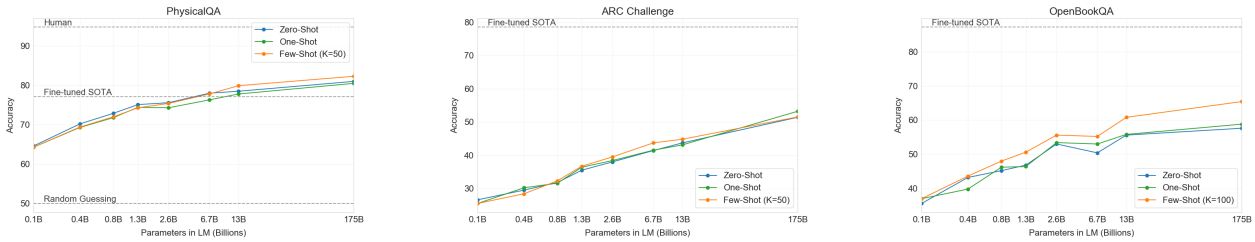

Figure H.6: All results for all Common Sense Reasoning tasks.

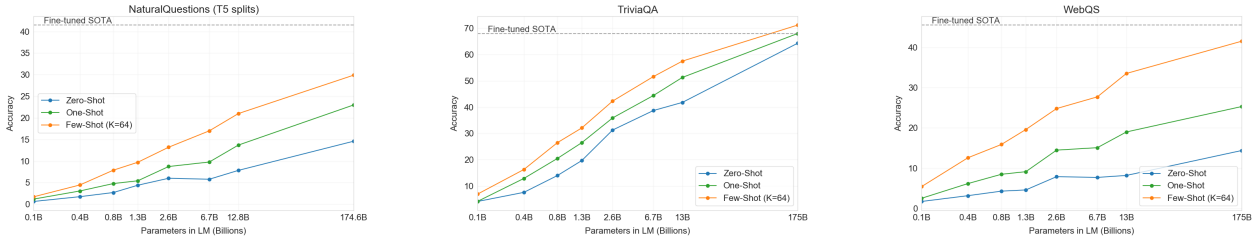

Figure H.7: All results for all QA tasks.

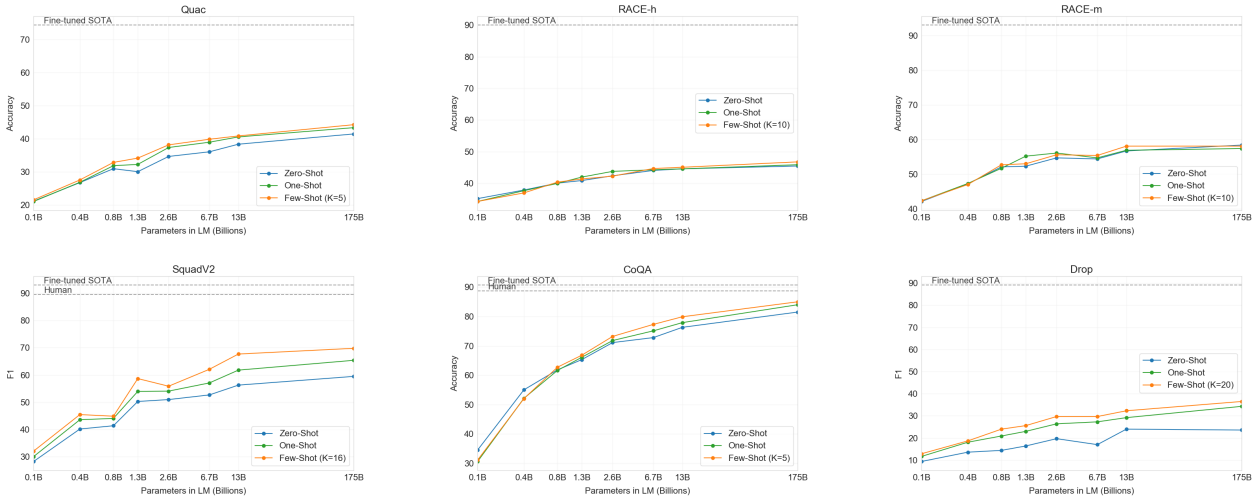

Figure H.8: All results for all Reading Comprehension tasks.

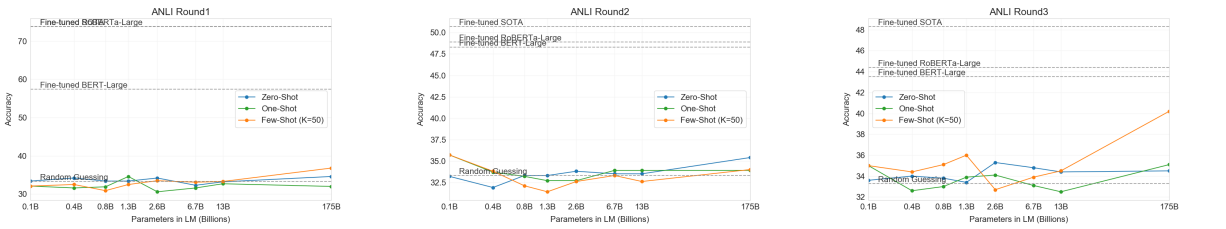

Figure H.9: All results for all ANLI rounds.

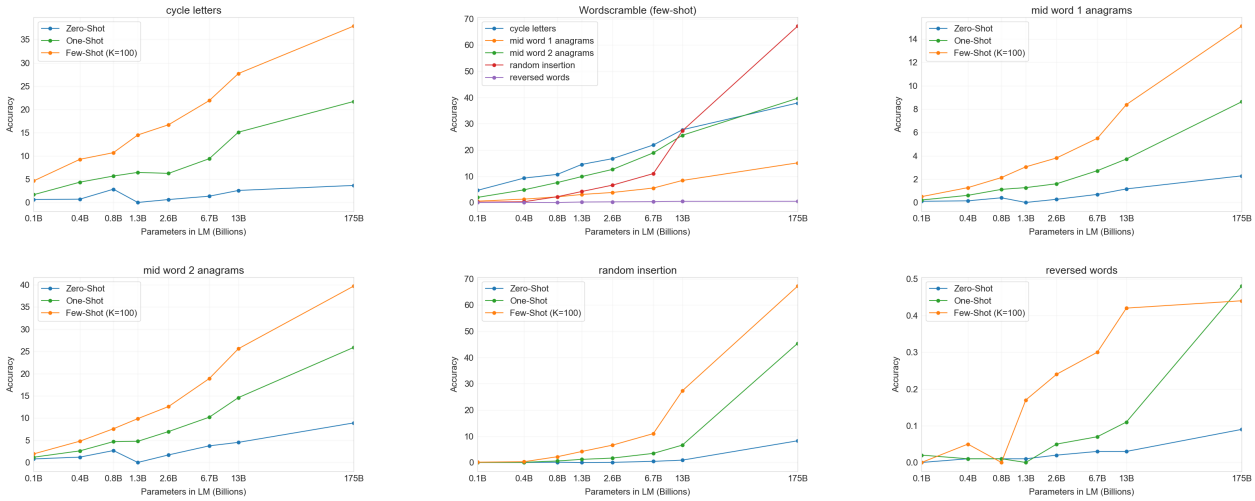

Figure H.10: All results for all Scramble tasks.

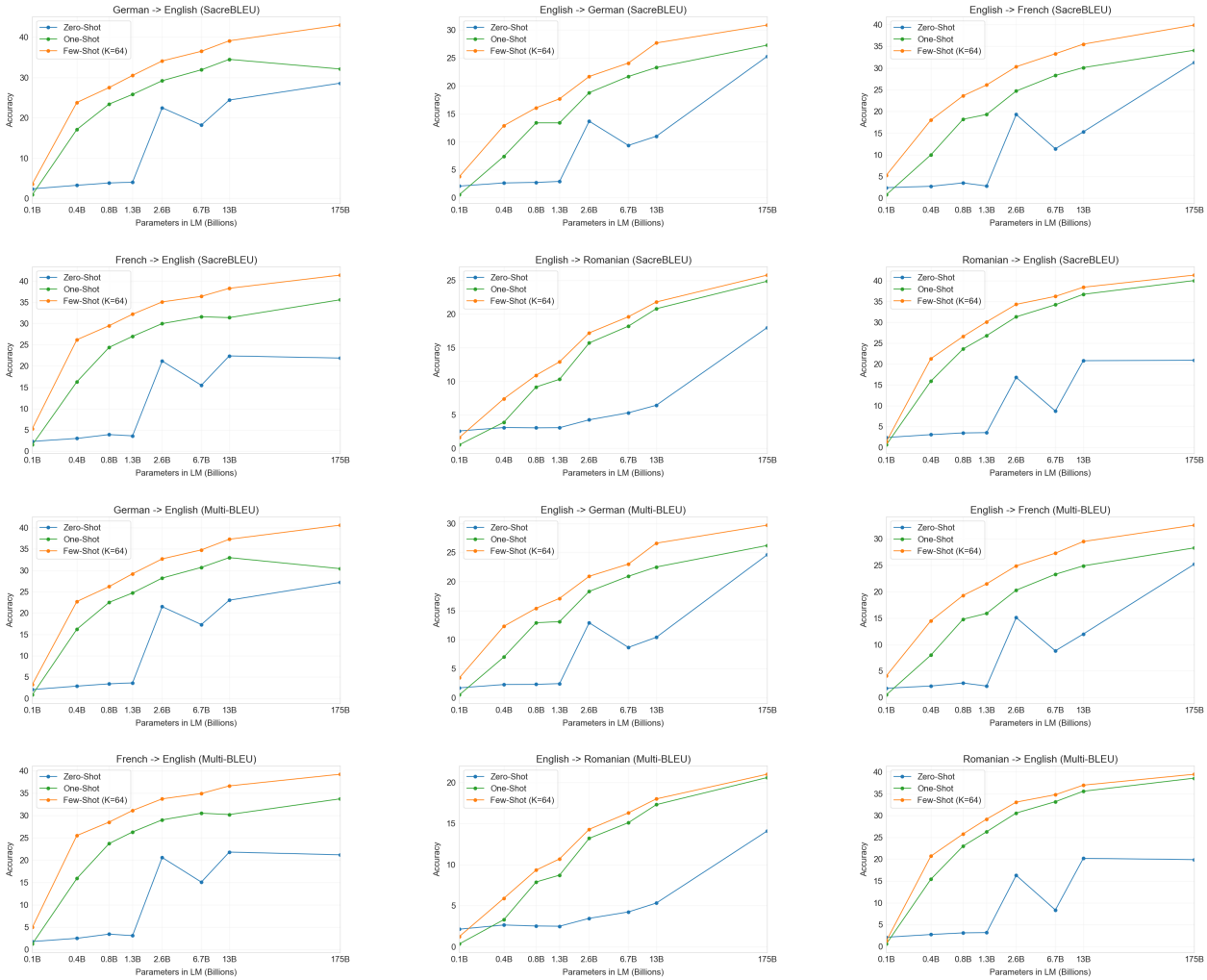

Figure H.11: All results for all Translation tasks.

References

[ADG+16] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

[AI19] WeChat AI. Tr-mt (ensemble), December 2019.

[AJF19] Roee Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. 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)*, 2019.

[BBDIW20] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. Language (technology) is power: A critical survey of "bias" in nlp. *arXiv preprint arXiv:2005.14050*, 2020.

[BCFL13] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[BDD+09] Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. The fifth PASCAL recognizing textual entailment challenge. 2009.

[BES10] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204, 2010.

[BHDD+06] Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second PASCAL recognising textual entailment challenge. 2006.

[BHT+20] Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, et al. Experience grounds language. *arXiv preprint arXiv:2004.10151*, 2020.

[BLC13] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *Arxiv*, 2013.

[BZB+19] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv:1911.11641*, 2019.

[Car97] Rich Caruana. Multitask learning. *Machine learning*, 28(1), 1997.

[CB78] Susan Carey and Elsa Bartlett. Acquiring a single new word. *Proceedings of the Stanford Child Language Conference*, 1978.

[CCE+18] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

[CGRS19] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

[CHI+18] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac : Question answering in context. *Arxiv*, 2018.

[CLC+19] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[CLY+19] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019.

[Cra17] Kate Crawford. The trouble with bias. *NIPS 2017 Keynote*, 2017.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DGM06] Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising textual entailment*, pages 177–190. Springer, 2006.

[DGV+18] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *Arxiv*, 2018.

[DHKH14] Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. Edinburgh's phrase-based machine translation systems for wmt-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 97–104, 2014.

[DL15] Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, 2015.

[DMST19] Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The CommitmentBank: Investigating projection in naturally occurring discourse. 2019. To appear in proceedings of Sinn und Bedeutung 23. Data can be found at https://github.com/mcdm/CommitmentBank/.

[DSC+16] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *ArXiv*, abs/1611.02779, 2016.

[DWD+19] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

[DYY+19] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *Arxiv*, 2019.

[EOAG18] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *ArXiv*, abs/1703.03400, 2017.

[Fyo00] Yaroslav Fyodorov. A natural logic inference system, 2000.

[GG19] Hila Gonen and Yoav Goldberg. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. *arXiv preprint arXiv:1903.03862*, 2019.

[GLT+20] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

[GMDD07] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics, 2007.

[Gra16] Alex Graves. Adaptive computation time for recurrent neural networks. *Arxiv*, 2016.

[GSL+18] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*, 2018.

[GSR19] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv: 1906.04043*, 2019.

[GWC+18] Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. Meta-learning for low-resource neural machine translation. *arXiv preprint arXiv:1808.08437*, 2018.

[HB20] Daniel Hernandez and Tom Brown. Ai and efficiency, May 2020.

[HBFC19] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019.

[HLW+20] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. Pretrained transformers improve out of distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.

[HNA+17] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[HR18] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[HYC01] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to Learn Using Gradient Descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[HZJ+19] Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. Reducing sentiment bias in language models via counterfactual evaluation. *arXiv preprint arXiv:1911.03064*, 2019.

[IBGC+14] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing*, 2014.

[IDCBE19] Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. Automatic detection of generated text is easiest when humans are fooled. *arXiv preprint arXiv:1911.00650*, 2019.

[JCWZ17] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[JN20] Zheng Junyuan and Gamma Lab NYC. Numeric transformer - albert, March 2020.

[JVS+16] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[JYS+19] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[JZC+19] Ying Ju, Fubang Zhao, Shijie Chen, Bowen Zheng, Xuefeng Yang, and Yunfeng Liu. Technical report on conversational question answering. *arXiv preprint arXiv:1909.10772*, 2019.

[KCR+18] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[KKS+20] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.

[KMB20] Sarah E. Kreps, Miles McCain, and Miles Brundage. All the news that's fit to fabricate: Ai-generated text as a tool of media misinformation, 2020.

[KMH+20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[KPR+19] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[KR16] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. *Arxiv*, 2016.

[LB02] Edward Loper and Steven Bird. Nltk: The natural language toolkit, 2002.

[LC19] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

[LCG+19] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[LCH+20] Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. Adversarial training for large neural language models. *arXiv preprint arXiv:2004.08994*, 2020.

[LDL19] Zhongyang Li, Xiao Ding, and Ting Liu. Story ending prediction by transferable bert. *arXiv preprint arXiv:1905.07504*, 2019.

[LDM12] Hector Levesque, Ernest Davis, and Leora Morgenstern. The Winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012.

[LGG+20] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*, 2020.

[LGH+15] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[LH17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[LHCG19a] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*, 2019.

[LHCG19b] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[Lin20] Tal Linzen. How can we accelerate progress towards human-like linguistic generalization? *arXiv preprint arXiv:2005.00955*, 2020.

[LLG+19] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[LM17] Ke Li and Jitendra Malik. Learning to optimize neural nets. *arXiv preprint arXiv:1703.00441*, 2017.

[LOG+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[LPP+20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Kiela Douwe. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

[LSP+18] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating Wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

[LWS+20] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. Train large, then compress: Rethinking model size for efficient training and inference of transformers, 2020.

[LXL+17] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

[LYN+20] Sheng-Chieh Lin, Jheng-Hong Yang, Rodrigo Nogueira, Ming-Feng Tsai, Chuan-Ju Wang, and Jimmy Lin. Tttttackling winogrande schemas. *arXiv preprint arXiv:2003.08380*, 2020.

[Mac92] David. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 1992.

[MBXS17] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MCH+16] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*, 2016.

[MCKS18] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *ArXiv*, abs/1809.02789, 2018.

[MKAT18] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training, 2018.

[MKM+94] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics, 1994.

[MKXS18] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

[MPL19] R Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*, 2019.

[MWZ+18] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting, 2018.

[NBR20] Moin Nadeem, Anna Bethke, and Siva Reddy. Stereoset: Measuring stereotypical bias in pretrained language models. *arXiv preprint arXiv:2004.09456*, 2020.

[NK19] Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments. *arXiv preprint arXiv:1907.07355*, 2019.

[Nor09] Peter Norvig. Natural language corpus data, 2009.

[NvNvdG19] Malvina Nissim, Rik van Noord, and Rob van der Goot. Fair is better than sensational: Man is to doctor as woman is to doctor. *arXiv preprint arXiv:1905.09866*, 2019.

[NWD+19] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*, 2019.

[oR16] University of Regensburg. Fascha, 2016.

[PCC18] Mohammad Taher Pilehvar and Jose Camacho-Collados. WIC: 10,000 example pairs for evaluating context-sensitive representations. *arXiv preprint arXiv:1808.09121*, 2018.

[PFB18] Jason Phang, Thibault Févry, and Samuel R. Bowman. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018.

[PHR+18] Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of EMNLP*, 2018.

[PKL+16] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

[PNZtY18] Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen tau Yih. Dissecting contextual word embeddings: Architecture and representation, 2018.

[Pos18] Matt Post. A call for clarity in reporting BLEU scores. *arXiv preprint arXiv:1804.08771*, 2018.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

[QIA20] QIANXIN. Sa-net on albert (ensemble), April 2020.

[QMZH19] Yusu Qian, Urwa Muaz, Ben Zhang, and Jae Won Hyun. Reducing gender bias in word-level language models with a gender-equalizing loss function. *arXiv preprint arXiv:1905.12801*, 2019.

[RBG11] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011.

[RCM19] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

[RCP+17] Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint arXiv:1710.10304*, 2017.

[RJL18] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[RL16] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *ICLR 2017 (oral)*, 2016.

[RLL+19] Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. NumNet: Machine reading comprehension with numerical reasoning. In *Proceedings of EMNLP*, 2019.

[RNLVD18] Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. Gender bias in coreference resolution. *arXiv preprint arXiv:1804.09301*, 2018.

[RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[Ros12] R.S. Ross. Guide for conducting risk assessments. *NIST Special Publication*, 2012.

[RRBS19] Jonathan S. Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales, 2019.

[RRS20] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

[RSR+19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.

[RWC+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[SBBC19] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.

[SBC+19] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. Release strategies and the social impacts of language models, 2019.

[SCNP19] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*, 2019.

[SDCW19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[SDSE19] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *CoRR*, abs/1907.10597, 2019.

[SHB15] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

[SMM+17] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[SPP+19] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2019.

[SS20] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few-shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020.

[STQ+19] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MASS: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

[TFR+17] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[TL05] Peter D. Turney and Michael L. Littman. Corpus-based learning of analogies and semantic relations. *CoRR*, abs/cs/0508103, 2005.

[TL18] Trieu H. Trinh and Quoc V. Le. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018.

[TLBS03] Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. *CoRR*, cs.CL/0309035, 2003.

[Tur20] Project Turing. Microsoft research blog, Feb 2020.

[VBL+16] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching Networks for One Shot Learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017.

[WPN+19] 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. In *Advances in Neural Information Processing Systems*, pages 3261–3275, 2019.

[WXH+18] Yiren Wang, Yingce Xia, Tianyu He, Fei Tian, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. Multi-agent dual learning. *ICLR 2019*, 2018.

[XDH+19] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training, 2019.

[YdC+19] Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*, 2019.

[YDY+19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[ZHB+19] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[ZHR+19] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. *arXiv preprint arXiv:1905.12616*, 2019.

[ZLL+18] Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*, 2018.

[ZSW+19a] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2019.

[ZSW+19b] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *ArXiv*, abs/1909.08593, 2019.