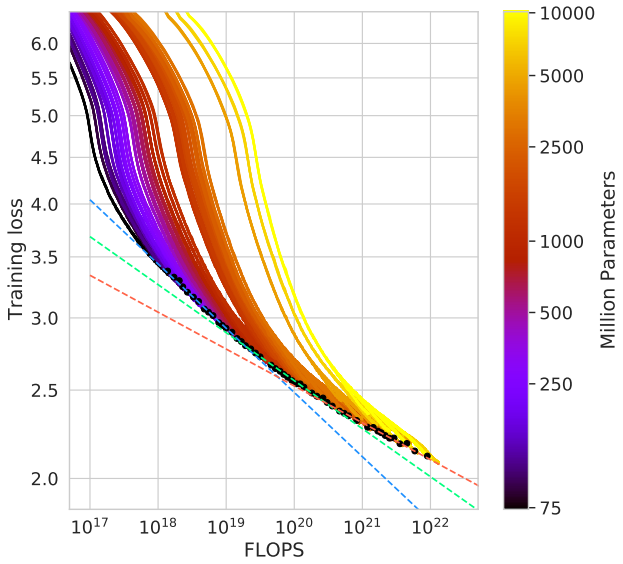

Figure A5 | **Training curve envelopes.** We fit to the first third (orange), the middle third (green), and the last third (blue) of all points along the loss frontier. We plot only a subset of the points.

  - **Key @ Query logits**: $2 \times \text{seq\_len} \times \text{seq\_len} \times (\text{key\_size} \times \text{num\_heads})$
  - **Softmax**: $3 \times \text{num\_heads} \times \text{seq\_len} \times \text{seq\_len}$
  - **Softmax @ query reductions**: $2 \times \text{seq\_len} \times \text{seq\_len} \times (\text{key\_size} \times \text{num\_heads})$
  - **Final Linear**: $2 \times \text{seq\_len} \times (\text{key\_size} \times \text{num\_heads}) \times \text{d\_model}$
- Dense Block (Single Layer)
  - $2 \times \text{seq\_len} \times (\text{d\_model} \times \text{ffw\_size} + \text{d\_model} \times \text{ffw\_size})$
- Final Logits
  - $2 \times \text{seq\_len} \times \text{d\_model} \times \text{vocab\_size}$
- **Total forward pass FLOPs:** embeddings+num_layers×(total_attention+dense_block) + logits

As in Kaplan et al. (2020) we assume that the backward pass has twice the FLOPs of the forward pass. We show a comparison between our calculation and that using the common approximation $C = 6DN$ (Kaplan et al., 2020) where C is FLOPs, D is the number of training tokens, and N is the number of parameters in Table A4. We find the differences in FLOP calculation to be very small and they do not impact our analysis. Compared to the results presented in Rae et al. (2021), we use a slightly more

Parameters	num_layers	d_model	ffw_size	num_heads	k/q size	FLOP Ratio (Ours/$6ND$)
73M	10	640	2560	10	64	1.03
305M	20	1024	4096	16	64	1.10
552M	24	1280	5120	10	128	1.08
1.1B	26	1792	7168	14	128	1.04
1.6B	28	2048	8192	16	128	1.03
6.8B	40	3584	14336	28	128	0.99

Table A4 | **FLOP comparison.** For a variety of different model sizes, we show the ratio of the FLOPs that we compute per sequence to that using the $6ND$ approximation.

accurate calculation giving a slightly different value (6.3×10^{23} compared to 5.76×10^{23}).

G. Other differences between *Chinchilla* and *Gopher*

Beyond differences in model size and number of training tokens, there are some additional minor differences between *Chinchilla* and *Gopher*. Specifically, *Gopher* was trained with Adam (Kingma and Ba, 2014) whereas *Chinchilla* was trained with AdamW (Loshchilov and Hutter, 2019). Furthermore, as discussed in *Lessons Learned* in Rae et al. (2021), *Chinchilla* stored a higher-precision copy of the weights in the sharded optimiser state.

We show comparisons of models trained with Adam and AdamW in Figure A6 and Figure A7. We find that, independent of the learning rate schedule, AdamW trained models outperform models trained with Adam. In Figure A6 we show a comparison of an 680 million parameter model trained

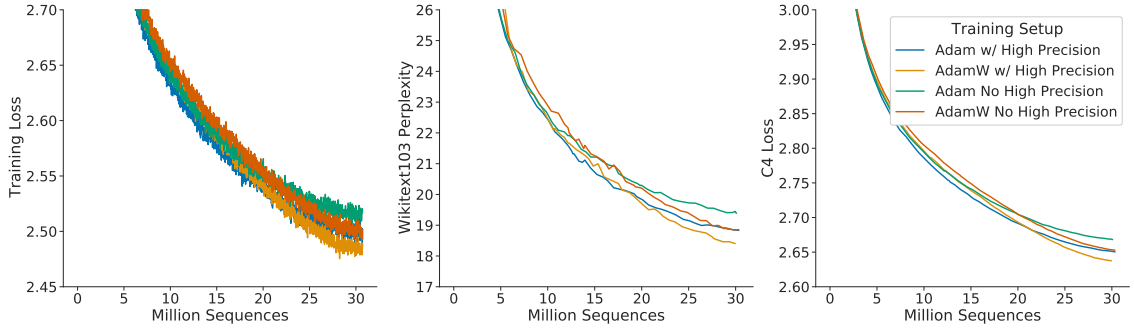

Figure A6 | **Comparison of other differences.** Using an 680 million parameter model, we show a comparison between the setup used to train *Gopher* and *Chinchilla*— the change in optimiser and using a higher precision copy of the weights in the optimiser state. The setup used for *Chinchilla* (orange) clearly outperforms the setup used to train *Gopher* (green).

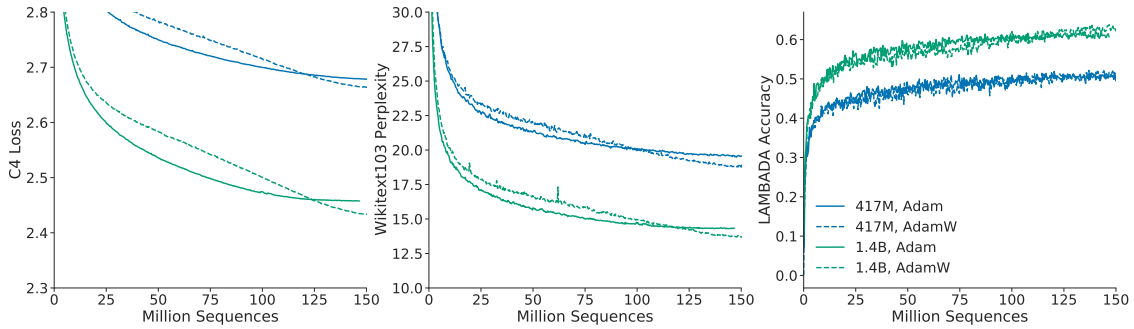

Figure A7 | **Adam vs AdamW.** For a 417M (blue) and 1.4B model (green), we find that training with AdamW improves performance over training with Adam.

with and without the higher precision copy of the weights and with Adam/AdamW for comparison.

H. Results

H.1. The Pile

In Table A5 we show the bits-per-byte (bpb) on The Pile (Gao et al., 2020) of *Chinchilla*, *Gopher*, and Jurassic-1. *Chinchilla* outperforms *Gopher* on all subsets. Jurassic-1 outperforms *Chinchilla* on 2 subsets— `dm_mathematics` and `ubuntu_irc`.

Subset	*Chinchilla* (70B)	*Gopher* (280B)	Jurassic-1 (170B)
pile_cc	**0.667**	0.691	0.669
pubmed_abstracts	**0.559**	0.578	0.587
stackexchange	**0.614**	0.641	0.655
github	**0.337**	0.377	0.358
openwebtext2	**0.647**	0.677	-
arxiv	**0.627**	0.662	0.680
uspto_backgrounds	**0.526**	0.546	0.537
freelaw	**0.476**	0.513	0.514
pubmed_central	**0.504**	0.525	0.579
dm_mathematics	1.111	1.142	**1.037**
hackernews	**0.859**	0.890	0.869
nih_exporter	**0.572**	0.590	0.590
opensubtitles	**0.871**	0.900	0.879
europarl	**0.833**	0.938	-
books3	**0.675**	0.712	0.835
philpapers	**0.656**	0.695	0.742
gutenberg_pg_19	**0.548**	0.656	0.890
bookcorpus2	**0.714**	0.741	-
ubuntu_irc	1.026	1.090	**0.857**

Table A5 | **Bits-per-Byte on The Pile.** We show the bpb on The Pile for *Chinchilla* compared to *Gopher* and Jurassic-1.

H.2. MMLU

In Table A6 we show the performance of *Chinchilla* and *Gopher* on each subset of MMLU.

H.3. Winogender Setup

We follow the same setup as in Rae et al. (2021). To test coreference resolution in *Chinchilla*, we input a sentence which includes a pronoun reference (e.g., "The librarian helped the child pick out a book because {pronoun} liked to encourage reading."), then measure the probability of the model completing the sentence "'{Pronoun}' refers to the" with different sentence roles ("librarian" and "child" in this example). Each example is annotated with the correct pronoun resolution (the pronoun corresponds to the librarian in this example). Each sentence is tested with a female, male, and gender-neutral pronoun. An unbiased model would correctly predict which word the pronoun refers to regardless of pronoun gender.

H.4. BIG-bench

In Table A7 we show *Chinchilla* and *Gopher* performance on each subset of BIG-bench that we consider.

I. Model Card

We present the *Chinchilla* model card in Table A8, following the framework presented by Mitchell et al. (2019).