Assignment 5 - Question 1

1. IsoFLOP Profile Scaling Exponents and Implications

The paper's Approach 2 (IsoFLOP profiles) found the following scaling relationships:

$$N_{opt} \propto C^a \text{ where } a = 0.49$$
$$D_{opt} \propto C^b \text{ where } b = 0.51$$

These values stand in stark contrast to Kaplan et al.'s findings of:

$$a_{Kaplan} = 0.73$$
$$b_{Kaplan} = 0.27$$

The practical implication of this difference is fundamental to how we should scale language models. While Kaplan et al.'s analysis suggested heavily favoring model size increases over training tokens when scaling compute, the new analysis shows that both quantities should scale almost equally. As stated in the paper, "All three approaches suggest that as compute budget increases, model size and the amount of training data should be increased in approximately equal proportions" (p.7). This represents a significant shift from current practices where models have grown larger while keeping training tokens relatively constant.

2. Optimal Configuration for Gopher's Compute Budget

For the compute budget of

$$5.76 \times 10^{23} \text{ FLOPs}$$

(Gopher's budget), the paper's analysis indicated:

Optimal Configuration:

- Model size: 40-70B parameters (Chinchilla implemented at 70B)
- Training tokens: 1.4T tokens

Gopher's Actual Configuration:

- Model size: 280B parameters
- Training tokens: 300B tokens

The analysis reveals that Gopher was significantly over-parameterized while being undertrained relative to the optimal configuration. As stated in the paper: "Based on our analysis in Section 3, the optimal model size for the Gopher compute budget is somewhere between 40 and 70 billion parameters" (p.9).

3. Computational Efficiency Analysis

Chinchilla's improvements came with no additional computational cost during training, while actually reducing computational requirements for deployment. Key evidence:

1. Training compute parity:

$$\text{Compute}_{Chinchilla} = \text{Compute}_{Gopher}$$

2. Downstream efficiency advantages:

- $4\times$ smaller parameter count
- Reduced memory footprint
- Lower inference cost

The paper explicitly states: "Both Chinchilla and Gopher have been trained for the same number of FLOPs but differ in the size of the model and the number of training tokens" (p.9).

4. MMLU Benchmark Performance

Performance comparison on the Massive Multitask Language Understanding (MMLU) benchmark:

$$\begin{aligned}\text{Chinchilla accuracy} &= 67.6\%\\ \text{Gopher accuracy} &= 60.0\%\\ \text{Human expert performance} &= 89.8\%\end{aligned}$$

Notably, Chinchilla achieved:

- 7.6 percentage point improvement over Gopher
- Greater than 90
- First model to achieve >90

5. Implications for Trillion-Parameter Models

According to Table 3, a compute-optimal 1 trillion parameter model would require:

$$\begin{aligned}\text{Compute budget} &= 1.27 \times 10^{26}\text{ FLOPs}\\ &= 221.3 \times \text{Gopher compute}\\ \text{Training tokens} &= 21.2\text{ trillion}\end{aligned}$$

The paper concludes that trillion-parameter models are currently sub-optimal, stating: ”Unless one has a compute budget of

$$10^{26}$$

FLOPs (over 250× the compute used to train Gopher), a 1 trillion parameter model is unlikely to be the optimal model to train” (p.8). This suggests that current efforts to train trillion-parameter models are computationally inefficient and that resources would be better spent training smaller models on more data.