Unified Scaling Laws for Routed Language Models

Unified Scaling Laws for Routed Language Models

Aidan Clark [*1] **Diego de las Casas** [*1] **Aurelia Guy** [*1] **Arthur Mensch** [*1] **Michela Paganini** [1] **Jordan Hoffmann** [1] **Bogdan Damoc** [1] **Blake Hechtman** [2] **Trevor Cai** [1] **Sebastian Borgeaud** [1] **George van den Driessche** [1] **Eliza Rutherford** [1] **Tom Hennigan** [1] **Matthew Johnson** [2] **Katie Millican** [1] **Albin Cassirer** [1] **Chris Jones** [1] **Chris Jones** [1] **Elena Buchatskaya** [1] **David Budden** [1] **Laurent Sifre** [1] **Simon Osindero** [1] **Oriol Vinyals** [1] **Jack Rae** [1] **Erich Elsen** [1] **Koray Kavukcuoglu** [1] **Karen Simonyan** [1]

Abstract

The performance of a language model has been shown to be effectively modeled as a power-law in its parameter count. Here we study the scaling behaviors of *Routing Networks*: architectures that conditionally use only a subset of their parameters while processing an input. For these models, parameter count and computational requirement form two independent axes along which an increase leads to better performance. In this work we derive and justify scaling laws defined on these two variables which generalize those known for standard language models and describe the performance of a wide range of routing architectures trained via three different techniques. Afterwards we provide two applications of these laws: first deriving an *Effective Parameter Count* along which all models scale at the same rate, and then using the scaling coefficients to give a quantitative comparison of the three routing techniques considered. Our analysis derives from an extensive evaluation of Routing Networks across five orders of magnitude of size, including models with hundreds of experts and hundreds of billions of parameters.

1. Introduction

It is a commonly held belief that increasing the size of a neural network leads to better performance, especially

in model size leads to an additive reduction in the model's loss (Kaplan et al., 2020; Hernandez et al., 2021; Henighan et al., 2020; Rosenfeld et al., 2019). These relationships are not well understood, but a key implication is that a sequence of small[1] models can be used both to infer the performance of models many times more powerful, but also to provide global information about the scalability of an architecture.

Enter Routing Networks: models with the unusual property that each input interacts with only a subset of the network's parameters — chosen independently for each datapoint (Bengio et al., 2016; 2013; Denoyer & Gallinari, 2014). For a Routing Network, the number of parameters is nearly independent from the computational cost of processing a datapoint. This bifurcates the definition of size and prevents a scaling law in parameters alone from fully describing the model class. Specific Routing Networks have been trained successfully at large scales (Fedus et al., 2021; Du et al., 2021; Artetxe et al., 2021), but the general scaling behavior is not well understood. In this work we analyze the behavior of routed language models so that we might infer the scaling laws that describe their performance.

Key contributions. We analyze three different techniques for training Routing Networks, detailed in §3: Sinkhorn-BASE, a sparse mixture-of-experts (SMOE) approach modifying BASE (Lewis et al., 2021); non-parametric HASH Layers (Roller et al., 2021); and routing via Reinforcement Learning (RL-R). With models up to 200 billion parameters, we observe the following:

tl;dr

(or just get bored and start scrolling twitter)

If you don't have time for the whole talk, our paper says:

> Mixture-of-Expert-like routed language models obey simple scaling laws which predict their performance as a function of the number of experts **E** and the base size of the model **N**.

This Talk:

Background
- Routing
- Scaling Laws

Main Result: A Scaling Law for Routed LMs
- Functional Form + Data Fitting
- Generalizing Across Routing Techniques
- Implications of the Scaling Law

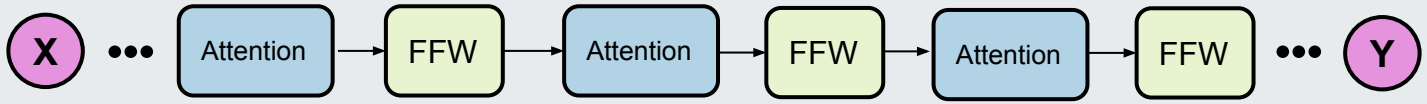
Three Applications
- Effective Parameter Count
- Limit Behavior of Routing

Conclusion

Background (Routing and Scaling Laws)

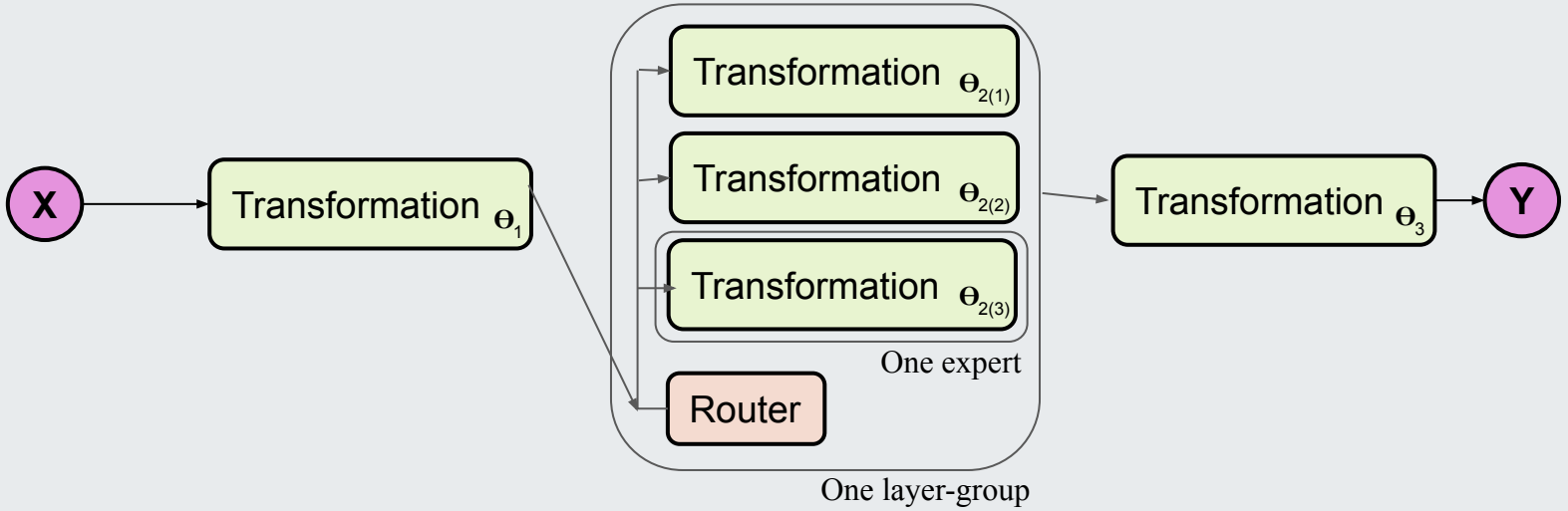
Neural Networks and Scaling

Most neural networks, including Transformers, are easily described as a sequence of transformations with distinct parameters applied iteratively to an input.



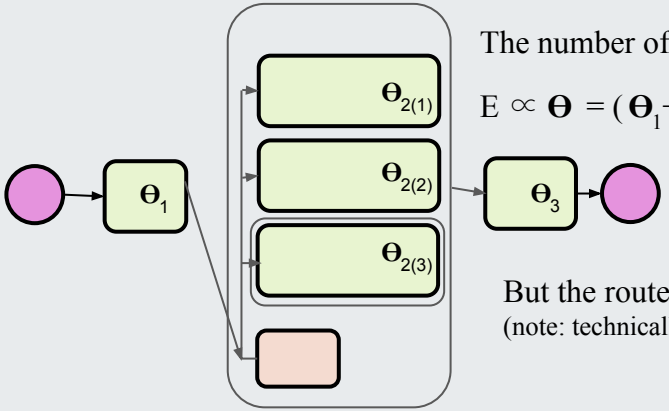
Routing

Routing is an alternative scaling method: where extra layers are added in parallel to existing ones, and a new layer called the **Router** decides which layer to use.



Routing Tradeoff

With routing, parameter count is proportional to **E** but execution has constant cost.



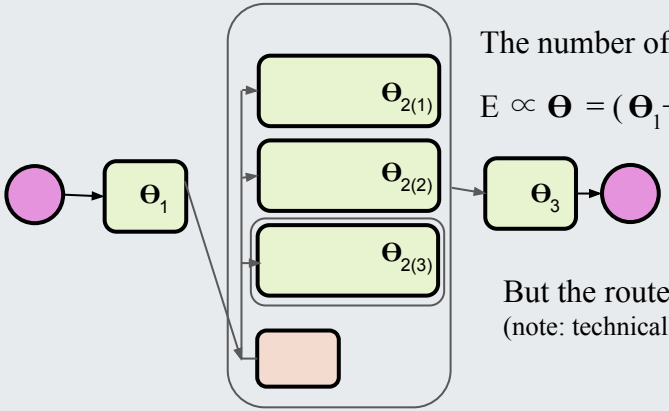
The number of parameters is proportional to E = 3…

$$E \propto \theta = (\theta_1 + \theta_{2(1)} + \theta_{2(2)} + \theta_{2(3)} + \theta_3)$$

But the router only picks a single expert! So the cost is fixed!
(note: technically the router picks K experts where K could be > 1)

Routing Tradeoff

With routing, parameter count is proportional to **E** but execution has constant cost.



The number of parameters is proportional to E = 3…

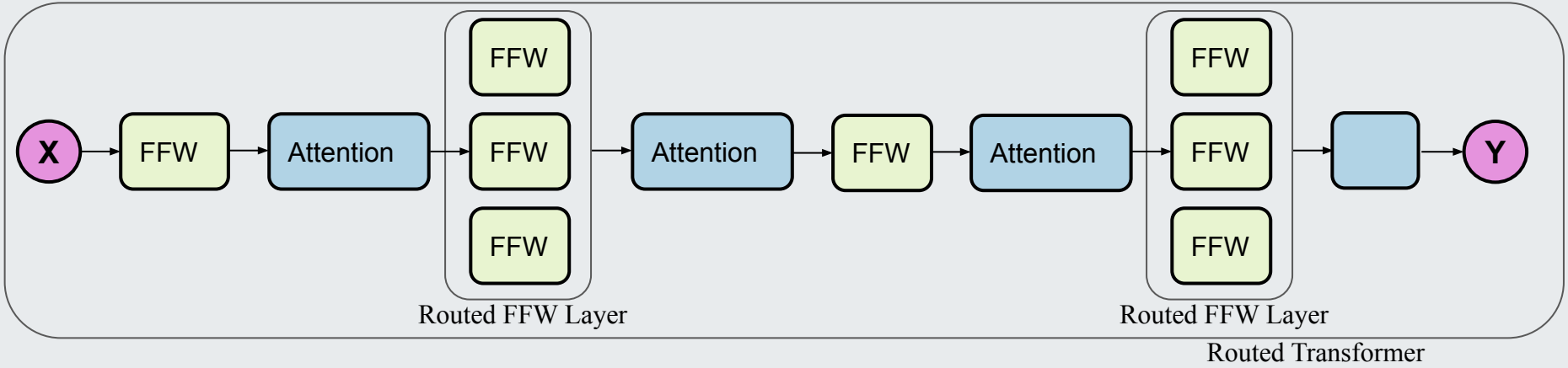
$$E \propto \theta = (\theta_1 + \theta_{2(1)} + \theta_{2(2)} + \theta_{2(3)} + \theta_3)$$

But the router only picks a single expert! So the cost is fixed!
(note: technically the router picks K experts where K could be > 1)

But this adds a challenge: you need a good router to pick the right expert, but its output (which expert to pick) is discrete so it can't just be trained end to end!

Routed Transformers

Routing can be applied to transformers by converting a subset of the feed-forward layers (FFWs) into routed equivalents, each with a distinct router.



(you can also route attention layers, but we don't analyze this in our paper)

Routing History

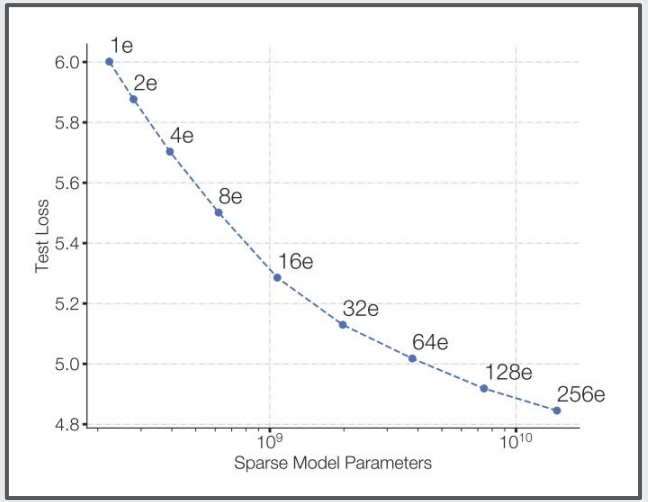
Multiple groups have trained routed transformers in this way [1, 2, 3], and argue that they outperform equivalently-costly dense transformers, most based on the Sparse Mixture of Experts (SMoE) architecture introduced by Shazeer et al [4].

Several of these papers argue that the improvement over dense transformers increases as you increase $\mathbf{E}$, the number of experts.

[1] Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer, https://arxiv.org/abs/1701.06538
[2] GShard: Scaling Giant Models with Conditional Computation [...], https://arxiv.org/abs/2006.16668
[3] Switch Transformers: Scaling to Trillion Parameter Models [...], https://arxiv.org/abs/2101.03961
[4] Efficient Large Scale Language Modeling with Mixtures of Experts https://arxiv.org/abs/2112.10684

Routing has Potential!

These are exciting results! We know that the cost of executing a routing network is independent of the number of experts **E**, so if increasing **E** improves performance, this potentially provides a cost-free avenue of scaling!



The performance of routed networks increases with E!
From [3] (Switch Transformer)

Routing: what do we want to know?

Routing: what do we want to know?

Q: How much better will my network be if I increase $\mathbf{E}$?

Routing: what do we want to know?

Q: How much better will my network be if I increase $\mathbf{E}$?

Q: Given a routed network, what is the equivalently powerful dense model?

Routing: what do we want to know?

Q: How much better will my network be if I increase $\mathbf{E}$?

Q: Given a routed network, what is the equivalently powerful dense model?

Q: How will the improvement from routing change if I increase the model's size?

Unknown Performance

We don’t know how to answer these questions! Routing improves the performance of language models, but exactly how and by how much is still unknown!

Unknown Performance

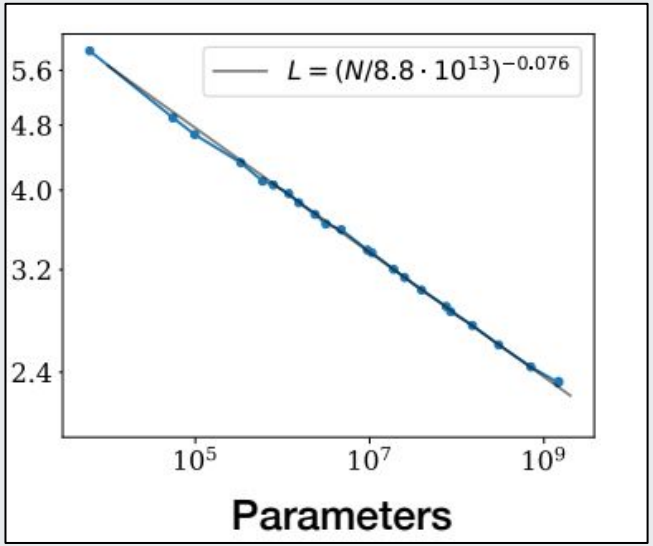
We don’t know how to answer these questions! Routing improves the performance of language models, but exactly how and by how much is still unknown!

What we want are the sort of guarantees that have been done for dense transformers with **scaling laws.**

Scaling Cost

Kaplan et al [5] showed the loss of dense transformers obey a power law:

$$\log L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$



[5] Scaling Laws for Neural Language Models, https://arxiv.org/abs/2001.08361

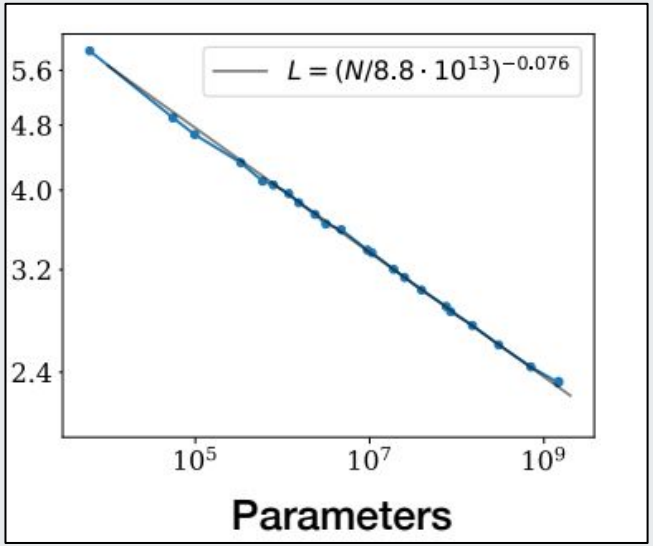
Scaling Cost

Kaplan et al [5] showed the loss of dense transformers obey a power law:

$$\log L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$

This is a near-guarantee that scaling works!
Knowing a functional form for the loss provides opportunities for all sorts of interesting analysis!

(like estimating the optimal model size for a given compute)



[5] Scaling Laws for Neural Language Models, https://arxiv.org/abs/2001.08361

Motivation + Overview

We want to extend the guarantees given by scaling laws to routing, so that we can predict the improvement routing will provide to any model and analyze their behavior at a larger scale to draw conclusions about the properties of routing networks.

Main Results

The Main Result: A Scaling Law for Routed Language Models

The performance of a routed transformer is given by a simple function of **N** and **E**:

$$\log L(N,E) \triangleq a \log N + b \log \widehat{E} + c \log N \log \widehat{E} + d$$

$$\text{where} \quad \frac{1}{\widehat{E}} \triangleq \frac{1}{E - 1 + \left(\frac{1}{E_{\text{start}}} - \frac{1}{E_{\text{max}}}\right)^{-1}} + \frac{1}{E_{\text{max}}}.$$

a, b, c, d, E_{start} and E_{max} are coefficients to be fit.

The Main Result: A Scaling Law for Routed Language Models

This scaling law can be decomposed in a clear way:

$$\log L(N, E) = a \log N + b \log \widehat{E} + c \log N \log \widehat{E} + d$$

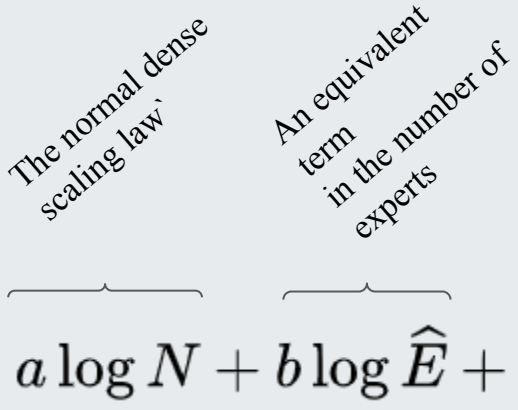
The Main Result: A Scaling Law for Routed Language Models

This scaling law can be decomposed in a clear way:

$$\log L(N, E) = \overbrace{a \log N + b \log \widehat{E} + c \log N \log \widehat{E} + d}^{\text{The normal dense scaling law}}$$

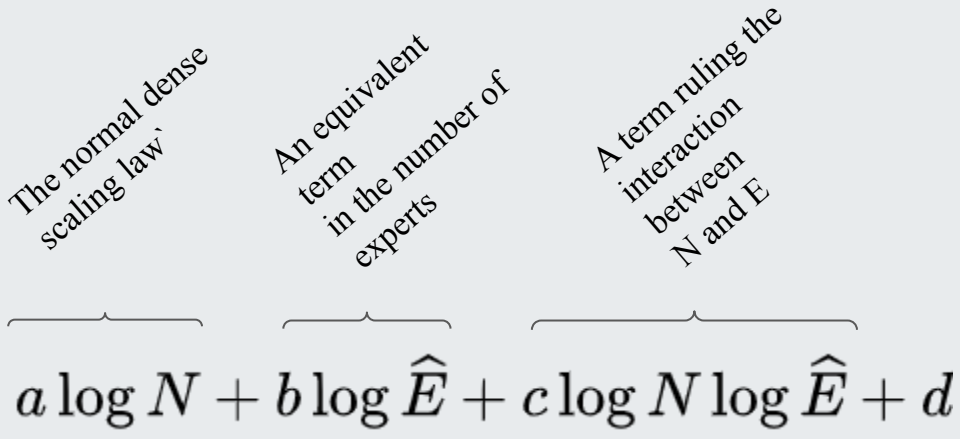
The Main Result: A Scaling Law for Routed Language Models

This scaling law can be decomposed in a clear way:



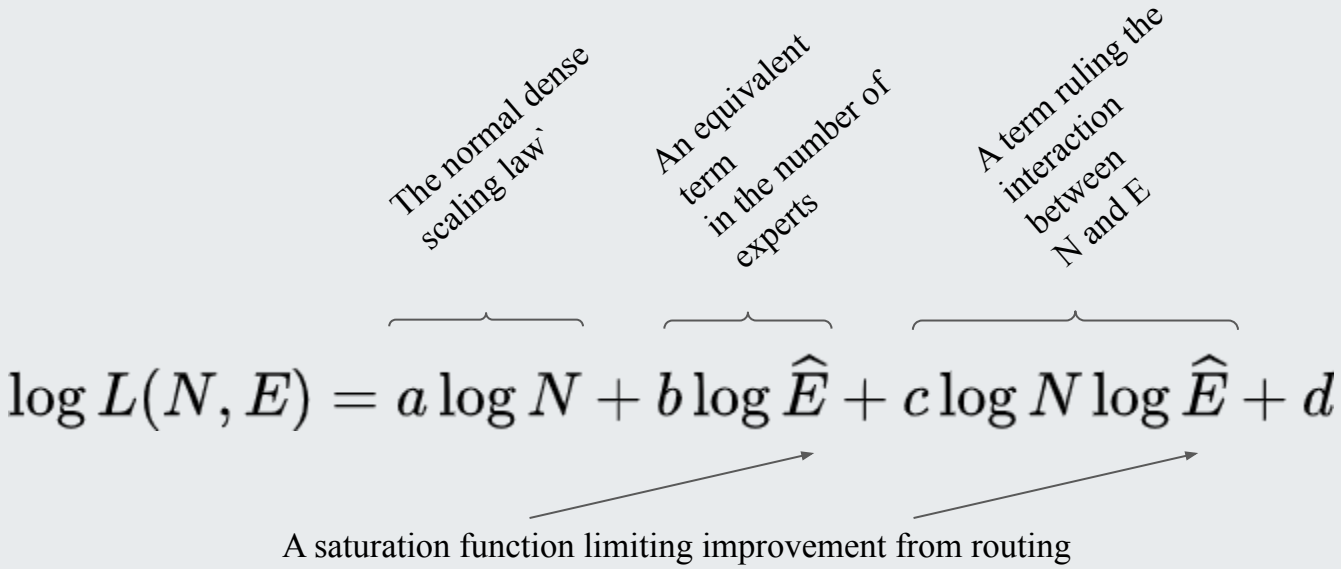
The Main Result: A Scaling Law for Routed Language Models

This scaling law can be decomposed in a clear way:



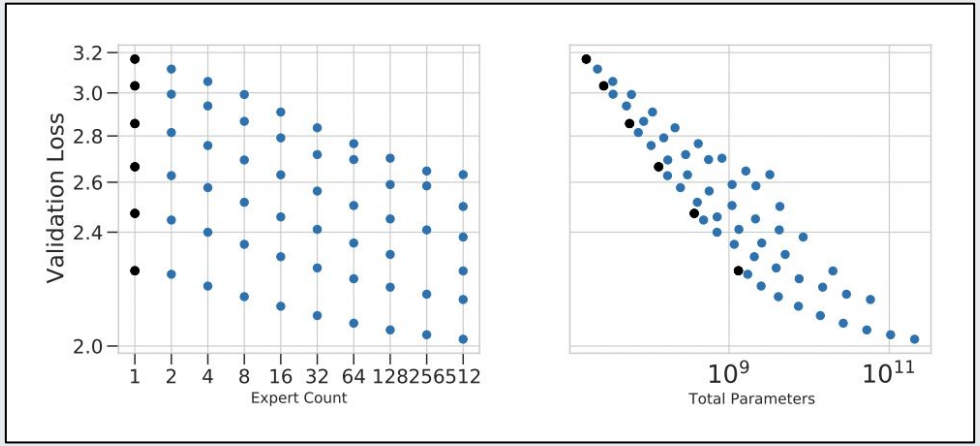
The Main Result: A Scaling Law for Routed Language Models

This scaling law can be decomposed in a clear way:



The Main Experiment Sweep

To provide evidence for this scaling law, we trained a large set of routed transformers, varying **N** from 15M to 1.3B and **E** from 1 to 512.



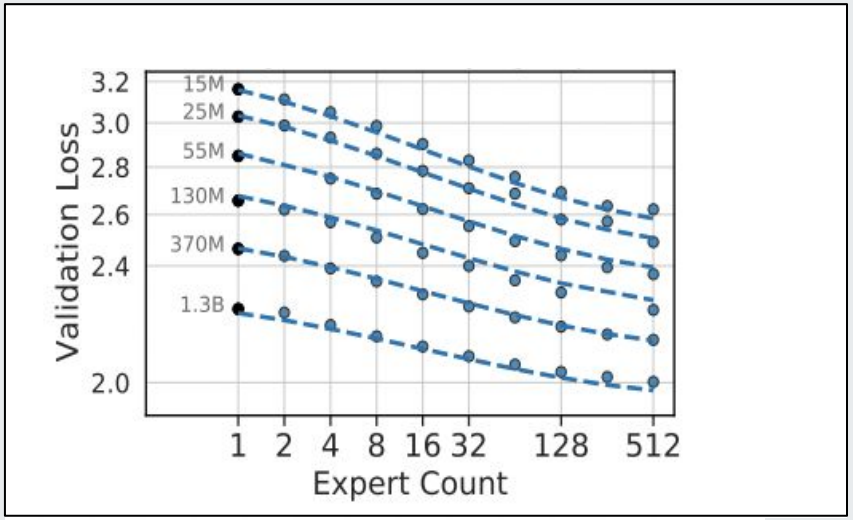
Each dot here is the validation loss of a separately trained routed transformer. Different rows represent different values of **N**.

The Main Experiment Sweep

Given that data, we can solve for coefficients:

Table 3: Solutions to Eq. (1).

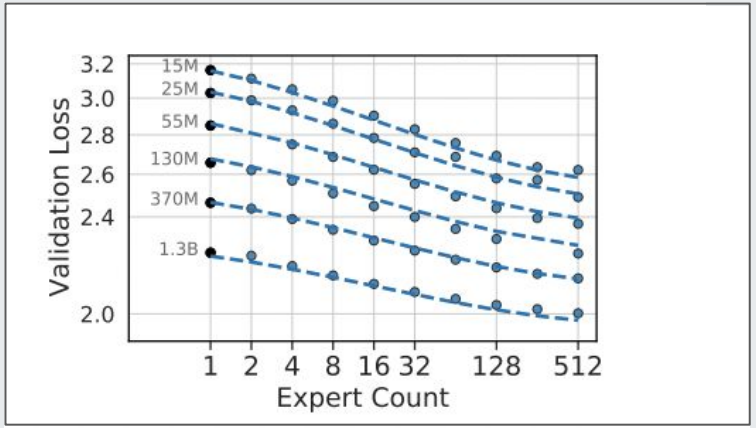
a	b	c	d	E_{start}	E_{max}
-0.082	-0.108	0.009	1.104	1.847	314.478



And the scaling laws can be used to give predictions for the loss any routed network will achieve as a function of **N** and **E**(dotted lines).

The Main Experiment Sweep (II)

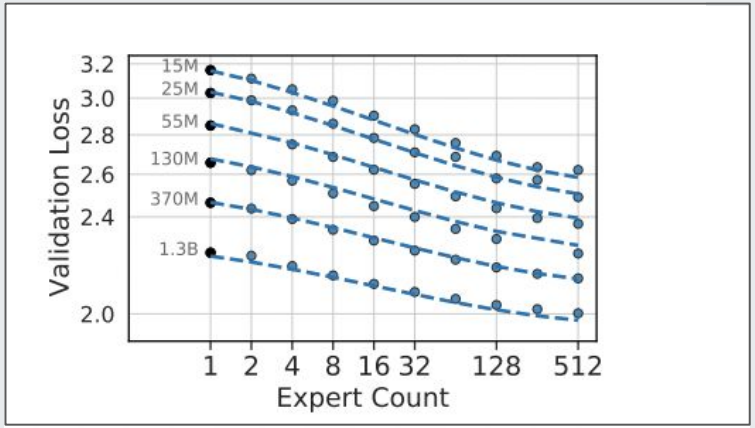
Let’s dwell on this plot for a second, it’s the key result!



The Main Experiment Sweep (II)

Let’s dwell on this plot for a second, it’s the key result!

The dots are real data, i.e. the validation loss of a routed transformer with a certain {**N**, **E**}

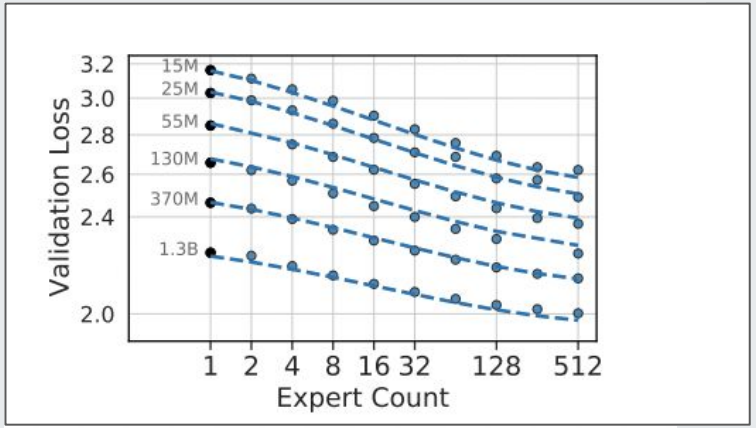


The Main Experiment Sweep (II)

Let's dwell on this plot for a second, it's the key result!

The dots are real data, i.e. the validation loss of a routed transformer with a certain {**N**, **E**}

The dotted lines are predicted fits, i.e., what our scaling laws predict the loss will be for a certain value of **N** when sweeping over **E**.

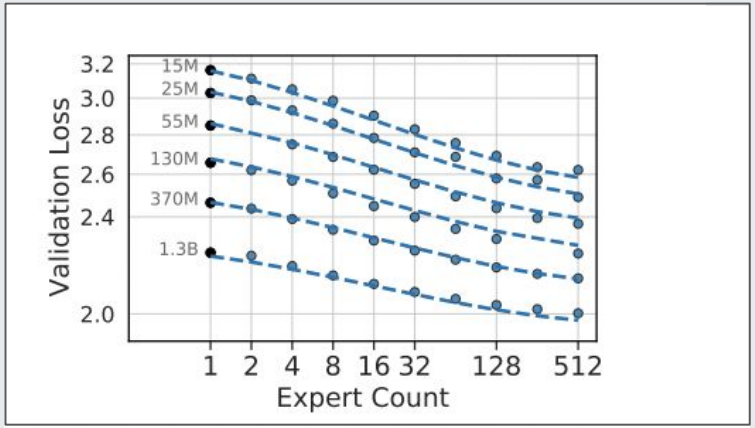


The Main Experiment Sweep (II)

Let's dwell on this plot for a second, it's the key result!

The dots are real data, i.e. the validation loss of a routed transformer with a certain {**N**, **E**}

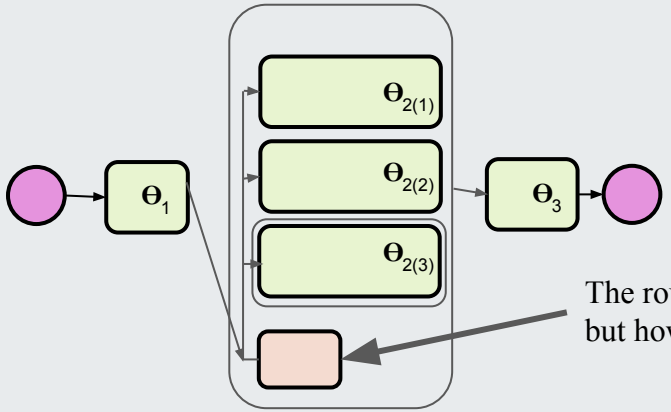
The dotted lines are predicted fits, i.e., what our scaling laws predict the loss will be for a certain value of **N** when sweeping over **E**.



They overlap extremely well! We have proposed a scaling law which can accurately model the behavior of routed transformers!

Different Routing Techniques

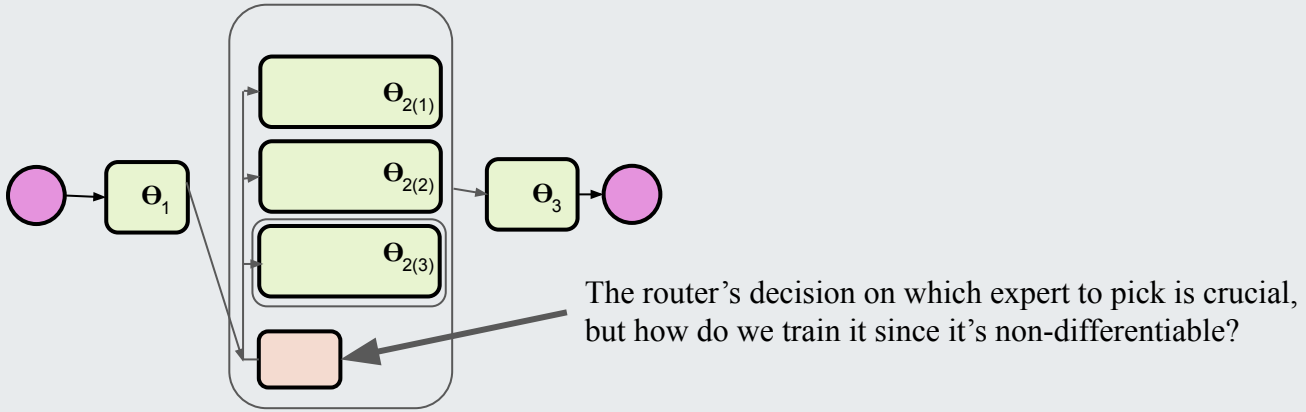
We want these scaling laws to generalize as much as possible. A key point of generalization comes down to exactly how we train the router.



The router's decision on which expert to pick is crucial, but how do we train it since it's non-differentiable?

Different Routing Techniques

We want these scaling laws to generalize as much as possible. A key point of generalization comes down to exactly how we train the router.



In fact, we ran that previous experiment sweep **three** different times!

S-BASE: Sinkhorn-balanced Switch Transformers

The first time we used **S-BASE**, a technique similar to BASE [6] or Switch Transformers [7].

We solve this problem using the Sinkhorn algorithm [Knopp and Sinkhorn, 1967], that takes the logit matrix $L \in \mathbb{R}^{T \times E}$ and returns a soft-assignment matrix $\Pi \in \mathbb{R}^{T \times E}$. The Sinkhorn algorithm solves Eq. (19) by alternated ascent in the dual (see Peyré and Cuturi [2019] for details). Starting from $f_0 = 0 \in \mathbb{R}^T$ and $g_0 = 0 \in \mathbb{R}^E$, we set

$$\forall i \in [T], \qquad (f_{t+1})_i = -\log \frac{1}{E} \sum_{j=1}^{E} \exp(L_{ij} - (g_t)_j), \tag{20}$$

$$\forall j \in [E], \qquad (g_{t+1})_j = -\log \frac{1}{T} \sum_{i=1}^{T} \exp(L_{ij} - (f_{t+1})_i).$$

These methods reuse the selection logits as scalar multipliers, providing a gradient to the policy (albeit not the exact right one).

[6] BASE Layers: Simplifying Training of Large, Sparse Models, https://arxiv.org/abs/2103.16716
[7] Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, https://arxiv.org/abs/2101.03961

RL-R: Training the Router via A2C

The second time, we used **RL-R**, a technique we propose that trains the router with RL.

An alternative improvement is to learn an additional baseline function for each router. This method has an additional entropy regularization loss and computes advantages $A_i = R_i - b_i$ for the learned baseline b_i:

$$L = \frac{1}{N}\sum_{i=1}^{N} \log p_i \cdot A_i - \frac{1}{N}\sum_{i=1}^{N} \log p_i \cdot p_i + \frac{1}{N}\sum_{i=1}^{N} v_i \tag{26}$$

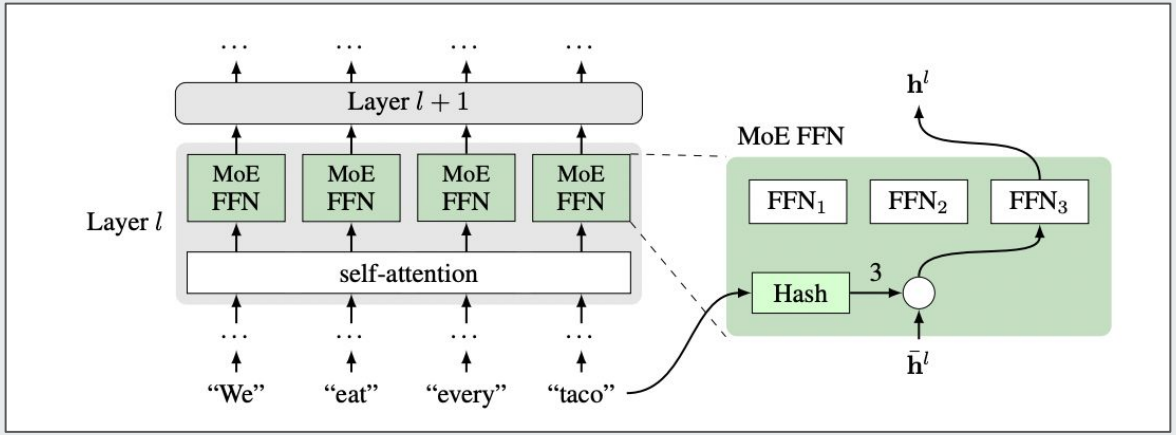
Where we use the Huber Loss to calculate the value loss v_i.

$$v_i = \begin{cases} \frac{1}{2}(R_i - b_i)^2 & \text{if } |R_i - b_i| \leqslant \delta, \\ \delta(|R_i - b_i| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases} \tag{27}$$

Using reinforcement learning for routing is an old and studied idea, but hasn't been revisited in the context of large routed transformers.

Hash Layers: Non-Parametric Expert Assignment

The third time, we used **Hash Layers [8]**, a recently proposed non-parametric alternative.



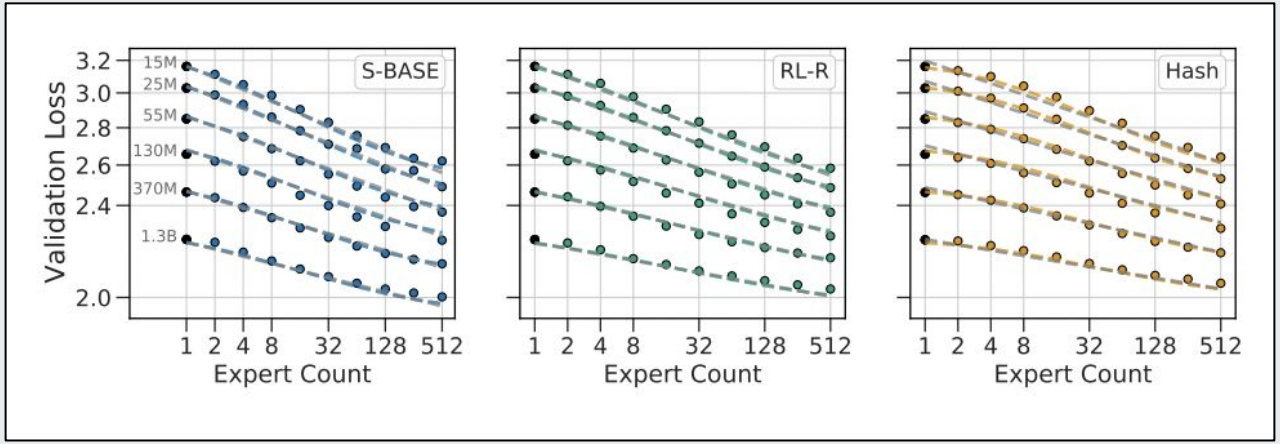
From [8]

[8] Hash Layers For Large Sparse Models, https://arxiv.org/abs/2106.04426

Generalization One: Different Routing Techniques (III)

In all three cases, the resulting models clearly and cleanly fit our proposed scaling law (with different coefficients).

Our scaling analysis generalized equally well to three different routing techniques!



Answering the First Question!

Recall one of the questions we wanted to ask of our routed networks:

Q: How much better will my network be if I increase $\mathbf{E}$?

Answering the First Question!

Recall one of the questions we wanted to ask of our routed networks:

Q: How much better will my network be if I increase **E**?

We can now trivially answer this! Plug in **N** and **E** and solve!

$$\log L(N, E) \triangleq a \log N + b \log \widehat{E} + c \log N \log \widehat{E} + d$$

$$\text{where} \quad \frac{1}{\widehat{E}} \triangleq \frac{1}{E - 1 + \left(\frac{1}{E_{\text{start}}} - \frac{1}{E_{\text{max}}}\right)^{-1}} + \frac{1}{E_{\text{max}}}.$$

Properties of the Scaling Law

The functional form we propose is simple and has a number of important properties.

$$\log L(N, E) \triangleq a \log N + b \log \widehat{E} + c \log N \log \widehat{E} + d$$

$$\text{where} \quad \frac{1}{\widehat{E}} \triangleq \frac{1}{E - 1 + \left(\frac{1}{E_{\text{start}}} - \frac{1}{E_{\text{max}}}\right)^{-1}} + \frac{1}{E_{\text{max}}}.$$

Properties of the Scaling Law (I)

Property One: (Log-)Linear Partial Derivatives

$$a(E) \triangleq -\frac{\partial \log L}{\partial \log N} = a + c \log(E)$$
$$b(N) \triangleq -\frac{\partial \log L}{\partial \log E} = b + c \log(N),$$

When fixing **N** or **E**, varying the other variable leads to a simple power-law, meaning this scaling law is compatible with the dense transformer's scaling law.

Properties of the Scaling Law (II)

Property Two: Saturation of E

$$\frac{1}{\widehat{E}} \triangleq \frac{1}{E - 1 + \left(\frac{1}{E_{\text{start}}} - \frac{1}{E_{\text{max}}}\right)^{-1}} + \frac{1}{E_{\text{max}}}.$$

Our scaling law includes a saturation function $\mathbf{E} \rightarrow \hat{\mathbf{E}}$, limiting the improvement so that an arbitrarily high number of experts gives bounded performance.

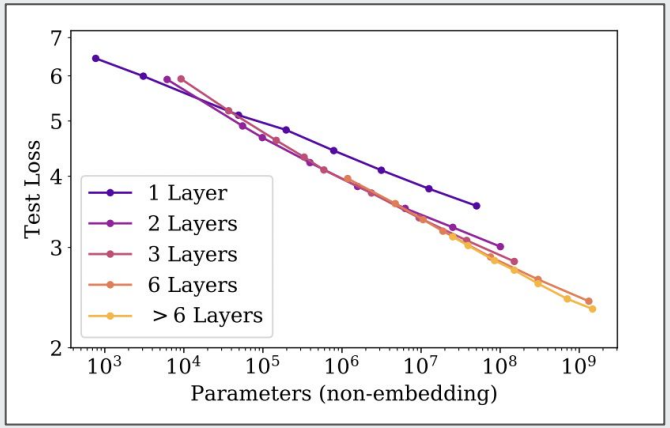
Under this saturation function, the maximum performance from routing is equivalent to the performance achieved with E_{max} experts **without** saturation.

Furthermore, for $E >> E_{start}$ and $E << E_{max}$, performance varies near-linearly.

Properties of the Scaling Law (III)

Property Three: Generalization via Change-of-Variables

This scaling law can be trivially reparametrized to fit a wider set of architectures, similar to how dense scaling laws apply equally when scaling in depth or scaling in width.



From [5] (Scaling Laws)

Properties of the Scaling Law (III)

Property Three: Generalization via Change-of-Variables

There are several key routing parameters for which changes affect the performance of the network but do not change $\mathbf{N}$ or $\mathbf{E}$, especially:

K: the number of experts each datapoint is sent to

(we default to K = 1 like Switch Transformers)

R: the percentage of layers which are routed

(we default to 50% like GShard and many others)

Properties of the Scaling Law (III)

Property Three: Generalization via Change-of-Variables

There are several key routing parameters for which changes affect the performance of the network but do not change $\mathbf{N}$ or $\mathbf{E}$, especially:

K: the number of experts each datapoint is sent to

(we default to K = 1 like Switch Transformers)

R: the percentage of layers which are routed

(we default to 50% like GShard and many others)

The scaling laws as proposed do not generalize across these architectures!

Properties of the Scaling Law (III)

Property Three: Generalization via Change-of-Variables

Luckily, we can fix that! We perform a change of variables, expressing our scaling laws exclusively in terms of the FLOPs required to execute a datapoint (F), and the fraction of total parameters any one datapoint interacts with (B).

$$\log L(F, B) \triangleq a \log F + b \log \widehat{B} + c \log F \log \widehat{B} + d,$$

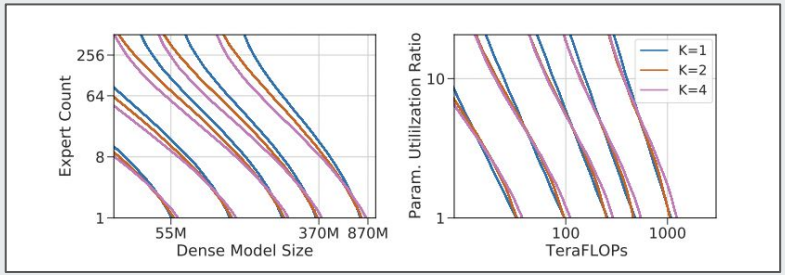
Properties of the Scaling Law (III)

Property Three: Generalization via Change-of-Variables

Luckily, we can fix that! We perform a change of variables, expressing our scaling laws exclusively in terms of the FLOPs required to execute a datapoint (F), and the fraction of total parameters any one datapoint interacts with (B).

$$\log L(F, B) \triangleq a \log F + b \log \widehat{B} + c \log F \log \widehat{B} + d,$$

The resulting scaling laws fit the data just as well, **and generalize to even more architectures!**



Half-way Recap!

1) Dense transformers obey simple scaling laws: bigger models are better.

2) Routing is an alternative type of scaling which improves performance at no cost!

3) We proposed a scaling law which predicts the performance of both dense and routed transformers as a function of **N** and **E**, and we trained a large sweep of routing networks to validate this law!

4) This scaling law fits the data well, and has a number of desired properties.

Two Applications of our Scaling Laws

Answering Questions

Recall the two other questions we wanted to ask of our routing networks:

Answering Questions

Recall the two other questions we wanted to ask of our routing networks:

Q: Given a routed network, what is the equivalently powerful dense model?

Answering Questions

Recall the two other questions we wanted to ask of our routing networks:

Q: Given a routed network, what is the equivalently powerful dense model?
Q: How will the improvement from routing change if I increase the model's size?

Answering Questions

Recall the two other questions we wanted to ask of our routing networks:

Q: Given a routed network, what is the equivalently powerful dense model?
Q: How will the improvement from routing change if I increase the model's size?

We can answer both questions now!

Application One: Effective Parameter Count

Q: Given a routed network, what is the equivalently powerful dense model?
I.e., L(**N**, **E**) is a certain value, what is the **N**' such that L(**N**, **E**) = L(**N**', 1)

Application One: Effective Parameter Count

Q: Given a routed network, what is the equivalently powerful dense model?
I.e., L($\mathbf{N}$, $\mathbf{E}$) is a certain value, what is the $\mathbf{N}$' such that L($\mathbf{N}$, $\mathbf{E}$) = L($\mathbf{N}$', 1)

A: We can solve for this!

Application One: Effective Parameter Count

Q: Given a routed network, what is the equivalently powerful dense model?
I.e., L(**N**, **E**) is a certain value, what is the **N**' such that L(**N**, **E**) = L(**N**', 1)

A: We can solve for this!

5.1 Effective Parameter Equivalence

We leverage Eq. (1) to compute the size $\bar{N}$ of a dense model giving the same performance as a Routing Network. Specifically, we solve for $L(\bar{N}, 1) = L(N, E)$, yielding

$$\bar{N} \triangleq (N)^{\alpha(\hat{E})/\alpha(E_{\text{start}})} \left(\hat{E}/E_{\text{start}}\right)^{b/\alpha(E_{\text{start}})} \tag{11}$$

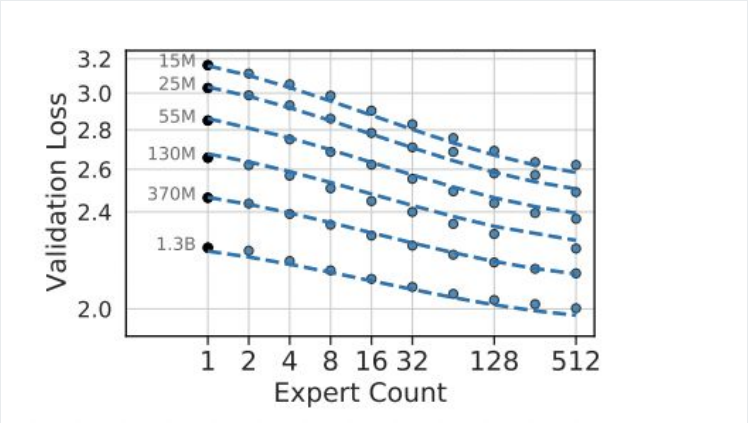
Here $\alpha(E) = a + c \log E$. Given a model with N and E, we call $\bar{N}$ that model's *Effective Parameter Count* (or EPC). Eq. (1) predicts that the performance of all models increases as a power law in this variable

$$\log L(N, E) = a \log \bar{N}(N, E) + d. \tag{12}$$

Given a network with **N** and **E**, we derive this equivalent size and call it the model's "Effective Parameter Count", or EPC.

Application One: Effective Parameter Count

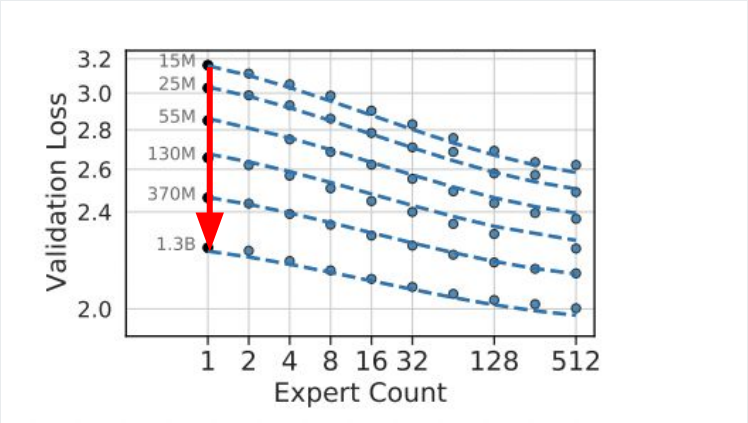
EPC represents a unified way to discuss model scale and power.



Application One: Effective Parameter Count

EPC represents a unified way to discuss model scale and power.

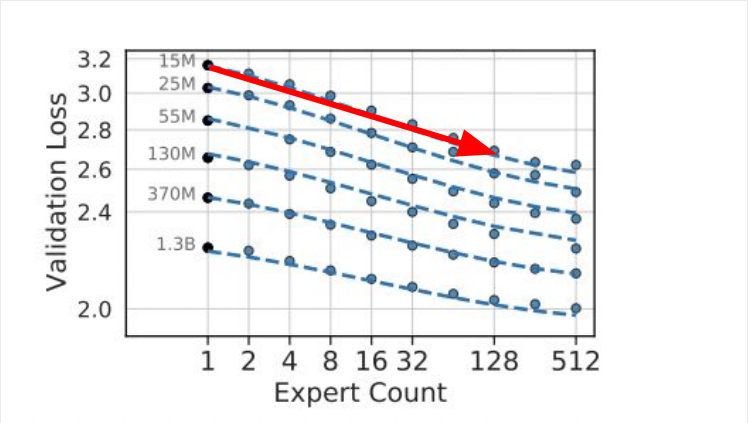
Before, **N** and **E** represented two different axes of improvement.
Increasing **N** improves performance very rapidly



Application One: Effective Parameter Count

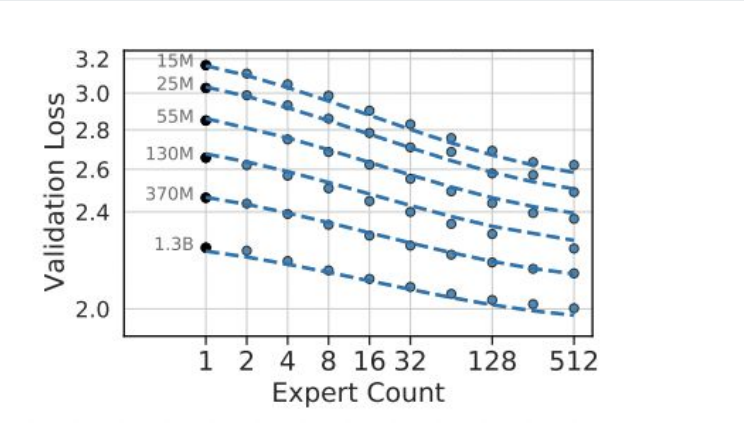
EPC represents a unified way to discuss model scale and power.

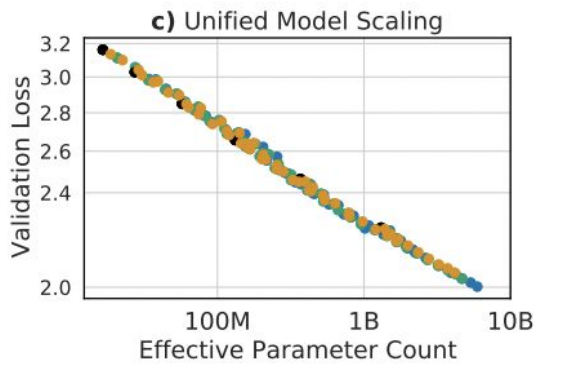
Before, **N** and **E** represented two different axes of improvement.
Increasing **N** improves performance very rapidly
Increasing **E** improves performance more slowly



Application One: Effective Parameter Count

With EPC, there is just a single axis of scaling.

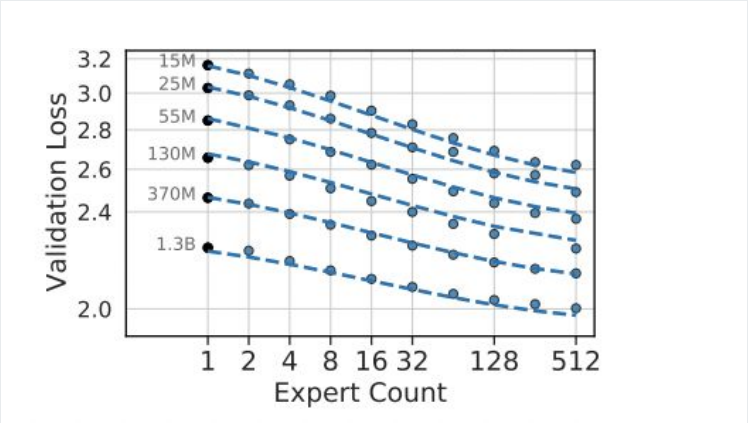


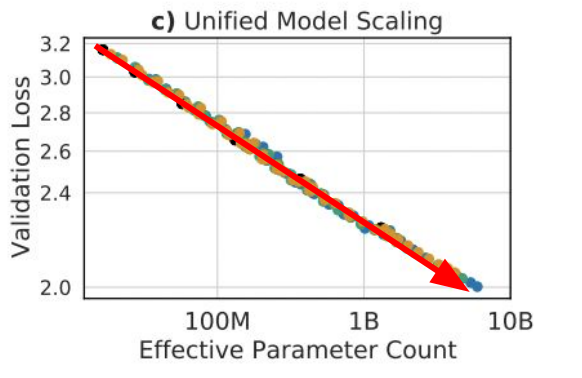


Application One: Effective Parameter Count

With EPC, there is just a single axis of scaling.

To know how much better an {**N**, **E**}-shaped model will be, just calculate the equivalent EPC!
All models, routed and dense, obey a simple scaling law of power-law performance in terms of EPC!





Application Two: Limitations to Scaling

Q: How will the improvement from routing change if I increase the model's size?

Application Two: Limitations to Scaling

Q: How will the improvement from routing change if I increase the model's size?

A: We can solve for this too!

Application Two: Limitations to Scaling

The key part is the third term of the scaling law, which models the interaction between **N** and **E** via a single coefficient c:

$$\log L(N, E) \triangleq a \log N + b \log \widehat{E} + c \log N \log \widehat{E} + d$$

Application Two: Limitations to Scaling

The key part is the third term of the scaling law, which models the interaction between **N** and **E** via a single coefficient c:

$$\log L(N, E) \triangleq a \log N + b \log \widehat{E} + c \log N \log \widehat{E} + d$$

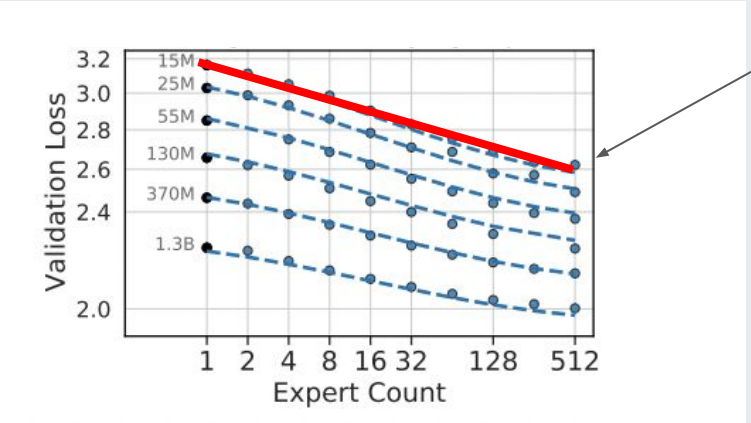
This term implies that the slope of improvement from increasing **E** varies linearly with **N**, meaning that as you change **N**, the slope of improvement to be expected from increasing **E** will change by a linear (in log-space) amount.

Application Two: Limitations to Scaling

This linear interaction isn’t just a quirk of the scaling law: it’s very visible in the data.

Application Two: Limitations to Scaling

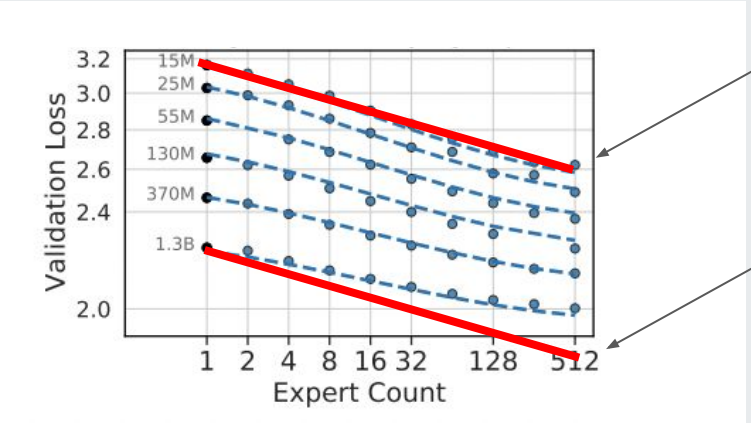
This linear interaction isn't just a quirk of the scaling law: it's very visible in the data.



Starting <u>at a small **N**</u>, you can see that increasing E leads to some slope of improvement.

Application Two: Limitations to Scaling

This linear interaction isn't just a quirk of the scaling law: it's very visible in the data.

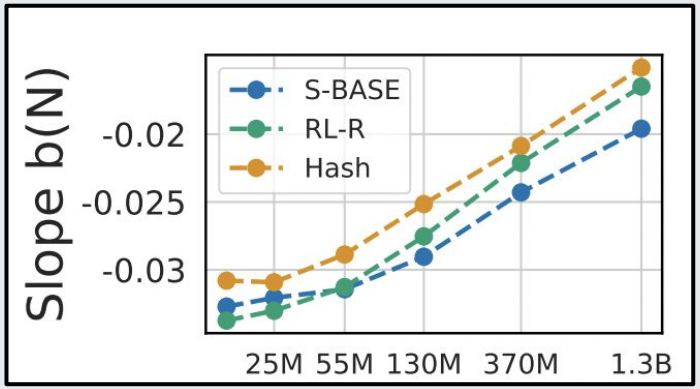


Starting at a small **N**, you can see that increasing E leads to some slope of improvement.

But starting from a larger **N**, that slope overestimates the performance substantially! The true slope of improvement has decreased.

Application Two: Limitations to Scaling

It turns out that a *positive* value of c is crucial for a good fit to the collected data, which implies that the benefit from routing is slowly reducing (even at a log-log scale!) as you increase **N**.



Application Two: Limitations to Scaling

Q: How will the improvement from routing change if I increase the model's size?

A: … the benefit from routing begins to decrease!

Application Two: Limitations to Scaling

Q: How will the improvement from routing change if I increase the model's size?

A: … the benefit from routing begins to decrease!

We take this analysis even further and ask: what happens in the limit as we increase the model size that we're routing?

Application Two: Limitations to Scaling

Q: How will the improvement from routing change if I increase the model's size to infinity?

A: We can solve for this too!

Application Two: Limitations to Scaling

Q: How will the improvement from routing change if I increase the model's size to infinity?

A: We can solve for this too!

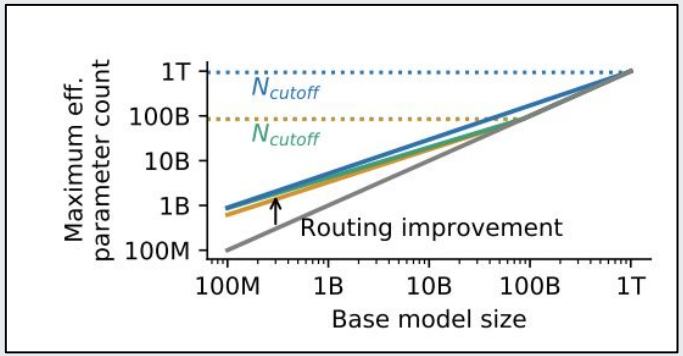
> Next we consider $\bar{N}_{\max}(N) \triangleq \max_E \bar{N}(N, E)$, i.e. the maximal effective parameter count that a routing network can reach. Eq. (11) predicts that $\log \bar{N}$ is an affine function of $\log N$ for any fixed E, and $\bar{N}_{\max}(N) = N$ for $N > N_{\text{cutoff}}$. Therefore $\log \bar{N}_{\max}$ is piecewise-affine in $\log N$, as displayed in Fig. 6:
>
> $$\forall N \leqslant N_{\text{cutoff}} = 10^{-\frac{b}{c}}, \quad \bar{N}_{\max}(N) = \bar{N}(N, E_{\max}),$$
> $$\forall N \geqslant N_{\text{cutoff}}, \bar{N}_{\max}(N) = N. \tag{13}$$

Because the slope of improvement from routing is reducing by a constant factor, we can solve for the point at which routing is predicted to stop giving any benefit at all!

Application Two: Limitations to Scaling

Q: How will the improvement from routing change if I increase the model's size to infinity?

A: We can solve for this too!

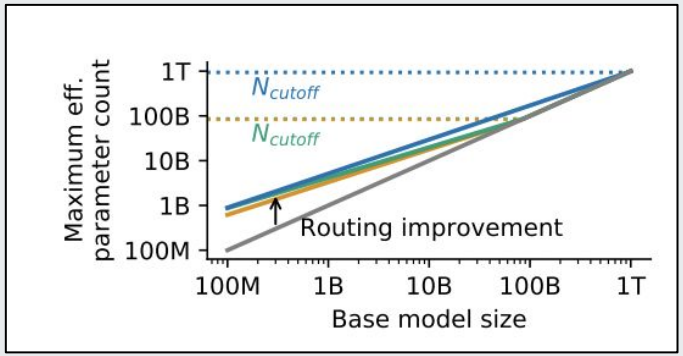


We call this value N_{cutoff}: the size at which routing is predicted to stop helping!
For RL-R, this value occurs at 90B. For S-BASE, routing helps up to **N** = 900B!

Application Two: Limitations to Scaling

Q: How will the improvement from routing change if I increase the model's size to infinity?

A: We can solve for this too!



This isn't some fundamental limitation of routing. These limits are tightly connected to the details of the training setup, and we believe changes (e.g., increasing the number of tokens trained on) might substantially effect these values. Rather, there are particular diminishing returns in the routing we observe, and we can calculate exactly where those losses catch up to you.

Conclusion

Conclusion

Our paper says that:

1. The performance of routing networks can be modelled via simple scaling laws which we have proposed, and we have provided a wide range of experimental data supporting these laws.

$$\log L(N,E) \triangleq a\log N + b\log \widehat{E} + c\log N\log \widehat{E} + d$$
$$\text{where} \quad \frac{1}{\widehat{E}} \triangleq \frac{1}{E-1+\left(\frac{1}{E_{\text{start}}}-\frac{1}{E_{\max}}\right)^{-1}} + \frac{1}{E_{\max}}.$$

2. Scaling laws for routing allows us to conduct a bunch of interesting analyses, especially in developing a unified scaling perspective (Effective Parameter Count!) and modelling the behavior of routing as the base size of the model is changed (predicting N_{cutoff}!).

Conclusion

Our paper says that:

1. The performance of routing networks can be modelled via simple scaling laws which we have proposed, and we have provided a wide range of experimental data supporting these laws.

$$\log L(N,E) \triangleq a\log N + b\log \widehat{E} + c\log N\log \widehat{E} + d$$

$$\text{where} \quad \frac{1}{\widehat{E}} \triangleq \frac{1}{E-1+\left(\frac{1}{E_{\text{start}}}-\frac{1}{E_{\max}}\right)^{-1}} + \frac{1}{E_{\max}}.$$

2. Scaling laws for routing allows us to conduct a bunch of interesting analyses, especially in developing a unified scaling perspective (Effective Parameter Count!) and modelling the behavior of routing as the base size of the model is changed (predicting N_{cutoff}!).

I want to leave you with two parting comments…..

Final Point #1

While it is true that our scaling laws predict the performance from routing will decrease, the decrease in effectiveness is very slow, and for the model sizes we have explored, routing is still extremely effective. **It is highly likely the transformers you-the-audience are training are in the regime where routing is extremely helpful, so you should consider using it!**

That said….

Final Point #1

While it is true that our scaling laws predict the performance from routing will decrease, the decrease in effectiveness is very slow, and for the model sizes we have explored, routing is still extremely effective. **It is highly likely the transformers you-the-audience are training are in the regime where routing is extremely helpful, so you should consider using it!**

That said….

I've presented routing as a rosy alternative to scaling, where increased experts means increased performance at no cost. The reality is more complicated. Building a fast and effective software stack for routing networks is surprisingly difficult, and best practices are highly dependent on details of the specific hardware and software.

Users be warned!

Final Point #2

We considered **a lot** of details to the routing and variants of the techniques I've mentioned. We have some really detailed appendices talking about them, including:

- (Many!) more variants of routing techniques
- Architectural changes and ablations to the routing architectures
- Extensive zero-shot evaluation on different downstream datasets
- Experimentation on larger ($\mathbf{N}$ = 7B) routed transformers
- Some exploration of the interaction between training tokens and validation loss

If you're interested in this work, don't skip our appendices! We have some interesting stuff in there :)

Fin :)

Thank you all for listening, and infinite thanks to all the co-authors on this work.

Our Paper: https://arxiv.org/abs/2202.01169
Data is open-sourced at: https://github.com/deepmind/scaling_laws_for_routing