Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

Ritik Dutta
Highlights

Paper presents a collection of techniques/tips that seem to work well for LM

- Simplifies MoE architecture (each token routed to 1 expert)
- Scaling results -> scaling model params while keeping FLOPs per token constant leads to better results
- Sparse MoE works great:
  - Switch is faster to train for similar performance to dense model (T5)
  - Finetuning performance is better
  - Distillation is also effective in retaining teacher’s performance while compressing model size
Why is this paper important?

- From scaling laws, we know that larger models are more sample-efficient
  - Switch transformers helps scale while keeping inference cost the same
- Dense models are harder to train, MoE is complicated
  - Shows that a single expert is sufficient, thus reducing complexity of MoE
  - Provide tips and tricks for stable training and distillation
- Adds a new dimension for scaling laws: increasing model size while keeping FLOPs constant also improves performance (i.e., sparsity)
  - Different from just increasing param. count (which is equivalent to more compute)
How is sparsity achieved?

- Sparsity due to activation of only one expert
Routing

Normal MoE

\[ p_i(x) = \frac{e^{h(x)_i}}{\sum_j^N e^{h(x)_j}}. \]

Gate value for \( i \)th expert

\[ y = \sum_{i \in T} p_i(x) E_i(x). \]

Layer output

(lin. combination of experts in \( T \))

For switch transformers, we just take \( \text{argmax}(p_i) \)
Why just one expert?

- Reduced routing computation
- Batch size of each expert can be at least halved
  - For top-2, each token is processed twice, for top-3, each token is processed thrice…
- Routing implementation is simplified & communication cost reduced
- Since forward pass FLOPs is now constant, you can scale without worrying about increased inference computation
  - Can leverage scaling -> better performance
Distributed Switch Implementation

- Tensor shapes are statically determined at compile time
- Use expert capacity -> number of tokens each expert computes

\[
\text{expert capacity} = \left( \frac{\text{tokens per batch}}{\text{number of experts}} \right) \times \text{capacity factor.}
\]

- Low capacity factor -> dropped tokens
- High capacity factor -> increased computation/communication cost
- Dropped tokens passed to next layer through residual connections
Distributed Switch Implementation

**Terminology**

- **Experts**: Split across devices, each having their own unique parameters. Perform standard feed-forward computation.

- **Expert Capacity**: Batch size of each expert. Calculated as 
  \[(\text{tokens\_per\_batch} / \text{num\_experts}) \times \text{capacity\_factor}\]

- **Capacity Factor**: Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.
Load Balancing Loss

- Auxiliary loss added to encourage balanced token routing for $N$ experts

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$

$$f_i = \frac{1}{T} \sum_{x \in B} \mathbb{1}\{\arg\max_x p(x) = i\} \quad P_i = \frac{1}{T} \sum_{x \in B} p_i(x).$$

fraction of tokens to expert $i$  
router probability to expert $i$

- Authors claim this loss encourages uniform routing since it is minimized under uniform distribution (not necessarily true!)
Load Balancing Loss

- Authors claim loss is minimized value for both $f_i$ and $P_i$ is $1/N$. But this is not true. Consider $N=2$, $T=3$

|         | Expert 1 | Expert 2 |
|---------|----------|----------|
| Token 1 | 0.51     | 0.49     |
| Token 2 | 0.51     | 0.49     |
| Token 3 | 0        | 1        |

$$\sum_{i=1}^{N} (f_i \cdot P_i) = \sum_{i=1}^{N} \left( \frac{1}{N} \cdot \frac{1}{N} \right) = \frac{1}{N}$$

$$\sum_{i=1}^{N} f_i \cdot P_i = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2}$$

Table of softmax values ($p_i$)

$$f = \left( \frac{2}{3}, \frac{1}{3} \right), \quad P = (0.34, 0.66), \quad \langle f, P \rangle = 0.447 < \frac{1}{2}$$

Credit: https://www.cs.princeton.edu/courses/archive/fall22/cos597G/lectures/lec16.pdf
Baselines

- **T5**
  - Released in 2020
  - Family of models: 60M, 220M (T5-Base), 740M (T5-Large), 3B, 11B
  - Represents dense models

- **MoE Transformers**
  - Released in 2017
  - In this paper, top-2 routing is used for comparison
  - FLOPs is larger than Switch Transformers because each expert applies its own FFN
Results

- Masked language modelling task where 15% of tokens are masked
- C4 dataset

| Model       | Capacity Factor | Quality after 100k steps (↑) (Neg. Log Perp.) | Time to Quality Threshold (↓) (hours) | Speed (↑) (examples/sec) |
|-------------|----------------|--------------------------------------------|------------------------------------|--------------------------|
| T5-Base     | —              | -1.731                                     | Not achieved†                      | 1600                     |
| T5-Large    | —              | -1.550                                     | 131.1                              | 470                      |
| MoE-Base    | 2.0            | -1.547                                     | 68.7                               | 840                      |
| Switch-Base | 2.0            | -1.554                                     | 72.8                               | 860                      |
| MoE-Base    | 1.25           | -1.559                                     | 80.7                               | 790                      |
| Switch-Base | 1.25           | -1.553                                     | 65.0                               | 910                      |
| MoE-Base    | 1.0            | -1.572                                     | 80.1                               | 860                      |
| Switch-Base | 1.0            | -1.561                                     | 62.8                               | 1000                     |
| Switch-Base+| 1.0            | **-1.534**                                 | 67.6                               | 780                      |

- Switch Transformers outperforms MoE and T5 on speed-quality basis
- Switch has smaller computational footprint
- Switch performs better at lower capacity factors
## Results

| Model      | Capacity Factor | Quality after 100k steps (↑) (Neg. Log Perp.) | Time to Quality Threshold (↓) (hours) | Speed (↑) (examples/sec) |
|------------|----------------|-----------------------------------------------|-------------------------------------|-------------------------|
| T5-Base    | —              | -1.731                                        | Not achieved†                       | 1600                    |
| T5-Large   | —              | -1.550                                        | 131.1                               | 470                     |
| MoE-Base   | 2.0            | -1.547                                        | 68.7                                | 840                     |
| Switch-Base| 2.0            | -1.554                                        | 72.8                                | 860                     |
| MoE-Base   | 1.25           | -1.559                                        | 80.7                                | 790                     |
| Switch-Base| 1.25           | -1.553                                        | 65.0                                | 910                     |
| MoE-Base   | 1.0            | -1.572                                        | 80.1                                | 860                     |
| Switch-Base| 1.0            | -1.561                                        | 62.8                                | 1000                    |
| Switch-Base+ | 1.0           | **-1.534**                                    | 67.6                                | 780                     |

MoE beats Switch here
## Results

| Model          | Capacity Factor | Quality after 100k steps (↑) (Neg. Log Perp.) | Time to Quality Threshold (↓) (hours) | Speed (↑) (examples/sec) |
|----------------|-----------------|-----------------------------------------------|---------------------------------------|--------------------------|
| T5-Base        | —               | -1.731                                        | Not achieved†                        | 1600                     |
| T5-Large       | —               | -1.550                                        | 131.1                                 | 470                      |
| MoE-Base       | 2.0             | -1.547                                        | 68.7                                  | 840                      |
| Switch-Base    | 2.0             | -1.554                                        | 72.8                                  | 860                      |
| MoE-Base       | 1.25            | -1.559                                        | 80.7                                  | 790                      |
| Switch-Base    | 1.25            | -1.553                                        | 65.0                                  | 910                      |
| MoE-Base       | 1.0             | -1.572                                        | 80.1                                  | 860                      |
| Switch-Base    | 1.0             | -1.561                                        | **62.8**                              | **1000**                 |
| Switch-Base+   | 1.0             | **-1.534**                                    | 67.6                                  | 780                      |

Increase computation elsewhere (non-expert part) to match MoE compute speed, and performance is better.
## Results

| Model       | Capacity Factor | Quality after 100k steps (↑) (Neg. Log Perp.) | Time to Quality Threshold (↓) (hours) | Speed (↑) (examples/sec) |
|-------------|----------------|----------------------------------------------|---------------------------------------|--------------------------|
| T5-Base     | —              | -1.731                                       | Not achieved†                         | 1600                     |
| T5-Large    | —              | -1.550                                       | 131.1                                 | 470                      |
| MoE-Base    | 2.0            | -1.547                                       | 68.7                                  | 840                      |
| Switch-Base | 2.0            | -1.554                                       | 72.8                                  | 860                      |
| MoE-Base    | 1.25           | -1.559                                       | 80.7                                  | 790                      |
| Switch-Base | 1.25           | -1.553                                       | 65.0                                  | 910                      |
| MoE-Base    | 1.0            | -1.572                                       | 80.1                                  | 860                      |
| Switch-Base | 1.0            | -1.561                                       | 62.8                                  | 1000                     |
| Switch-Base+| 1.0            | **-1.534**                                   | 67.6                                  | 780                      |

Speed-quality pareto optimality is somewhere here (reduce capacity factor), increase compute in non-expert layers.
Techniques for improving training

- Selective precision (float32 is slower to compute, also means you transferring more data between layers on potentially different devices)
- Reduced initialization scale
- Higher regularization of experts
Selective Precision

- Using only bfloat16 leads to instability (esp. during exponentiation)
- Using only float32 will increase costs
- Selectively cast router input to float32 precision - float32 only used within body of router function

```python
# Convert input to softmax operation from bfloat16 to float32 for stability.
router_logits = mtf.to_float32(router_logits)

# Probabilities for each token of what expert it should be sent to.
router_probs = mtf.softmax(router_logits, axis=-1)

# Cast back outputs to bfloat16 for the rest of the layer.
combine_tensor = mtf.to_bfloat16(combine_tensor)
```

| Model (precision)                  | Quality (Neg. Log Perp.) (†) | Speed (Examples/sec) (†) |
|------------------------------------|------------------------------|--------------------------|
| Switch-Base (float32)              | -1.718                       | 1160                     |
| Switch-Base (bfloat16)             | -3.780 [diverged]            | 1390                     |
| Switch-Base (Selective precision)  | -1.716                       | 1390                     |
Smaller Parameter Initialization

- Weight matrices initialized by sampling from a truncated normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{s/n}$

- Reduce default initialization scale $s = 1.0$ to $0.1$

| Model (Initialization scale) | Average Quality (Neg. Log Perp.) | Std. Dev. of Quality (Neg. Log Perp.) |
|-----------------------------|----------------------------------|--------------------------------------|
| Switch-Base (0.1x-init)     | -2.72                            | 0.01                                 |
| Switch-Base (1.0x-init)     | -3.60                            | 0.68                                 |

Table 3: Reduced initialization scale improves stability. Reducing the initialization scale results in better model quality and more stable training of Switch Transformer. Here we record the average and standard deviation of model quality, measured by the negative log perplexity, of a 32 expert model after 3.5k steps (3 random seeds each).
Higher Regularization of Experts

- Many finetuning tasks have very few examples -> leads to overfitting
- Switch Transformers have more parameters -> severe overfitting
- Increase dropout inside experts
  - Increasing dropout across all layers leads to worse performance

| Model (dropout)         | GLUE | CNNDM | SQuAD | SuperGLUE |
|-------------------------|------|-------|-------|-----------|
| T5-Base (d=0.1)        | 82.9 | 19.6  | 83.5  | 72.4      |
| Switch-Base (d=0.1)    | 84.7 | 19.1  | **83.7** | **73.0** |
| Switch-Base (d=0.2)    | 84.4 | 19.2  | **83.9** | **73.2** |
| Switch-Base (d=0.3)    | 83.9 | 19.6  | 83.4  | 70.7      |
| Switch-Base (d=0.1, ed=0.4) | **85.2** | **19.6** | **83.7** | **73.0** |

Table 4: Fine-tuning regularization results. A sweep of dropout rates while fine-tuning Switch Transformer models pre-trained on 34B tokens of the C4 data set (higher numbers are better). We observe that using a lower standard dropout rate at all non-expert layer, with a much larger dropout rate on the expert feed-forward layers, to perform the best.
Scaling on a step-basis

- Scaling experts (more params.) when training for fixed number of steps

![Graph showing test loss vs. sparse model parameters and training step vs. negative log perplexity. The graph illustrates the relationship between model parameters and perplexity, with a focus on scaling experts for fixed number of steps. There is a note indicating that the graph doesn’t account for communication time.]
Scaling properties on time basis

- Switch has more communication costs than T5
- For fixed training duration and comp. budget, Switch is better
Scaling vs larger dense model

- Increase sparsity (num. experts) or model density?

![Graph showing comparison between different models and their performance over time.](image-url)
Downstream Experiments

- Finetuning
- Distillation
- Multilingual learning
### Downstream: Finetuning results

- Switch is better

| Model         | GLUE | SQuAD | SuperGLUE | Winogrande (XL) |
|---------------|------|-------|-----------|-----------------|
| T5-Base       | 84.3 | 85.5  | 75.1      | 66.6            |
| Switch-Base   | **86.7** | **87.2** | **79.5**   | 73.3            |
| T5-Large      | 87.8 | 88.1  | 82.7      | 79.1            |
| Switch-Large  | **88.5** | **88.6** | **84.7**   | **83.0**        |

| Model         | XSum | ANLI (R3) | ARC Easy | ARC Chal. |
|---------------|------|-----------|----------|-----------|
| T5-Base       | 18.7 | 51.8      | 56.7     | 35.5      |
| Switch-Base   | **20.3** | **54.0**  | **61.3** | 32.8      |
| T5-Large      | 20.9 | 56.6      | 68.8     | 35.5      |
| Switch-Large  | **22.3** | **58.6**  | 66.0     | 35.5      |

| Model         | CB Web QA | CB Natural QA | CB Trivia QA |
|---------------|------------|---------------|--------------|
| T5-Base       | 26.6       | 25.8          | 24.5         |
| Switch-Base   | **27.4**   | **26.8**      | **30.7**     |
| T5-Large      | 27.7       | 27.6          | 29.5         |
| Switch-Large  | **31.3**   | **29.5**      | **36.9**     |
## Distillation Results

### Perplexity

| Technique                                    | Parameters | Quality (↑) |
|----------------------------------------------|------------|-------------|
| T5-Base                                      | 223M       | -1.636      |
| Switch-Base                                  | 3,800M     | -1.444      |
| Distillation + Init. non-expert weights from teacher | 223M       | (3%) -1.631 |
| Distillation + 0.75 mix of hard and soft loss | 223M       | (20%) -1.598 |
| Initialization Baseline (no distillation)     | 223M       | (29%) -1.580 |
| Init. non-expert weights from teacher         | 223M       | -1.639      |

Non-expert layers have same dimensions

Ground truth loss

Matching output logits of teacher model (Switch)

Loss = \(a \cdot L(\text{hard}) + (1-a) \cdot L(\text{soft})\)
## Distillation Results

| Parameters                               | Dense | Sparse |
|------------------------------------------|-------|--------|
| Parameters                               | 223M  | 1.1B   |
| Pre-trained Neg. Log Perp. (↑)           | -1.636| -1.505 |
| Distilled Neg. Log Perp. (↑)             | —     | -1.587 |
| Percent of Teacher Performance           | —     | 37%    |
| Compression Percent                      | —     | 82 %   |
|                                          | 1.1B  | 2.0B   |
|                                          | 2.0B  | 3.8B   |
|                                          | 3.8B  | 7.4B   |
|                                          | 7.4B  | 14.7B  |
|                                          | -1.474| -1.444 |
|                                          | -1.432| -1.427 |
|                                          | -1.585| -1.579 |
|                                          | -1.582| -1.578 |
|                                          | 32%   | 27 %   |
|                                          | 82 %  | 99 %   |
|                                          | 90 %  | 95 %   |
|                                          | 97 %  | 99 %   |
Multilingual learning

- Train on multilingual variant of C4 with 101 languages
- mSwitch is better than mT5 on all languages
Scaling strategies - Data, Model, Expert parallelism

Only scaling experts will give diminishing returns

How the *model weights* are split over cores

How the *data* is split over cores
Scaling strategies - Data, Model, Expert parallelism

How the *model weights* are split over cores

Data Parallelism  | Model Parallelism  | Model and Data Parallelism  | Expert and Data Parallelism

How the *data* is split over cores

Data Parallelism  | Model Parallelism  | Model and Data Parallelism  | Expert and Data Parallelism

Swich Base  
Switch Large

Param count increased by increasing intermediate dimensions
Scaling to trillion parameters

| Model       | Parameters | FLOPs/seq | $d_{model}$ | $\text{FFN}_{\text{GELU}}$ | $d_{ff}$ | $d_{kv}$ | Num. Heads |
|-------------|------------|-----------|-------------|----------------------------|----------|----------|-------------|
| T5-Base     | 0.2B       | 124B      | 768         | ✓                          | 2048     | 64       | 12          |
| T5-Large    | 0.7B       | 425B      | 1024        | ✓                          | 2816     | 64       | 16          |
| T5-XXL      | 11B        | 6.3T      | 4096        | ✓                          | 10240    | 64       | 64          |
| Switch-Base | 7B         | 124B      | 768         | ✓                          | 2048     | 64       | 12          |
| Switch-Large| 26B        | 425B      | 1024        | ✓                          | 2816     | 64       | 16          |
| Switch-XXL  | 395B       | 6.3T      | 4096        | ✓                          | 10240    | 64       | 64          |
| Switch-C    | 1571B      | 890B      | 2080        |                            | 6144     | 64       | 32          |

| Model       | Expert Freq. | Num. Layers | Num Experts | Neg. Log Perp. @250k | Neg. Log Perp. @ 500k |
|-------------|--------------|-------------|-------------|-----------------------|------------------------|
| T5-Base     | –            | 12          | –           | -1.599                | -1.556                 |
| T5-Large    | –            | 24          | –           | -1.402                | -1.350                 |
| T5-XXL      | –            | 24          | –           | -1.147                | -1.095                 |
| Switch-Base | 1/2          | 12          | 128         | -1.370                | -1.306                 |
| Switch-Large| 1/2          | 24          | 128         | -1.248                | -1.177                 |
| Switch-XXL  | 1/2          | 24          | 64          | **-1.086**            | **-1.008**             |
| Switch-C    | 1            | 15          | 2048        | -1.096                | -1.043                 |
## Scaling to trillion parameters

| Model      | Parameters | FLOPs/seq | $d_{model}$ | $FFN_{GELU}$ | $d_{ff}$ | $d_{kv}$ | Num. Heads |
|------------|------------|-----------|-------------|--------------|----------|----------|------------|
| T5-Base    | 0.2B       | 124B      | 768         | ✓            | 2048     | 64       | 12         |
| T5-Large   | 0.7B       | 425B      | 1024        | ✓            | 2816     | 64       | 16         |
| T5-XXL     | 11B        | 6.3T      | 4096        | ✓            | 10240    | 64       | 64         |
| Switch-Base| 7B         | 124B      | 768         | ✓            | 2048     | 64       | 12         |
| Switch-Large| 26B       | 425B      | 1024        | ✓            | 2816     | 64       | 16         |
| Switch-XXL | 395B       | 6.3T      | 4096        | ✓            | 10240    | 64       | 64         |
| Switch-C   | 1571B      | 890B      | 2080        |              | 6144     | 64       | 32         |

| Model      | Expert Freq. | Num. Layers | Num. Experts | Neg. Log Perp. @250k | Neg. Log Perp. @ 500k |
|------------|--------------|-------------|--------------|-----------------------|------------------------|
| T5-Base    | –            | 12          | –            | -1.599                | -1.556                 |
| T5-Large   | –            | 24          | –            | -1.402                | -1.350                 |
| T5-XXL     | –            | 24          | –            | -1.147                | -1.095                 |
| Switch-Base| 1/2          | 12          | 128          | -1.370                | -1.306                 |
| Switch-Large| 1/2       | 24          | 128          | -1.248                | -1.177                 |
| Switch-XXL | 1/2          | 24          | 64           | -1.086                | -1.008                 |
| Switch-C   | 1            | 15          | 2048         | -1.096                | -1.043                 |

- Size achieved by increasing experts, but worse performance than XXL
- ~10x less FLOPs
Additional Discussion

- Switch transformers works for smaller models too
- Even with 2 cores (1 expert per core), Switch is a better choice
Summary

- Bigger models are just better
- Don’t need multiple experts, a single expert is sufficient
- Use the following techniques for better training:
  - Mixed precision training (higher precision when doing exponentiation, etc.)
  - Smarter initialization with smaller values
  - Regularize using dropout
Additional Discussion

- Load balancing loss assumption is wrong (uniform routing does not minimize the loss), but still seems to work
  - This could mean that many tokens must pass through residual connection under optimal training
- How to reconcile uniform load distribution with expert specialization?
  - Specialization (polysemancticity) is observed in neurons
  - Do experts specialize in certain tasks (nouns, areas like english, grammar, etc.)?
  - If so, wouldn’t router probability depend on input distribution? Inputs dealing with math might get routed to expert 1 more frequently, english reasoning to expert 2, etc.
- Simply scaling experts leads to diminishing returns (Switch-C)
  - Increasing sparsity by just increasing expert count leads to diminishing returns even at 1T
  - Human brain has over 100T synapses (parameters)!
  - Would the training suggestions provided by authors scale to even larger sizes?
Thank You!