Mixture of Experts

Motivation

- Solving Complex Problems
- Enhancing Performance
- Scalability
- Diverse Applications

Motivation

- Solving Complex Problems
- Enhancing Performance
- Scalability
- Diverse Applications

Motivation

Solving Complex Problems

- By utilizing multiple experts, MoE can effectively handle complex and diverse problems with various aspects.
- Each expert can focus on a specific part of the problem, enhancing overall performance.

Motivation

Enhancing Performance

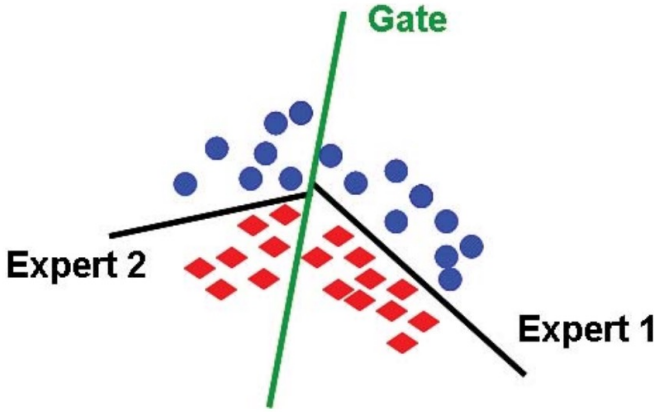
- MoE uses a gating mechanism to select the most suitable expert for each specific case, ensuring that each part of the problem is tackled by the best expert.
- Minimizes resource usage by activating only the necessary experts, saving time and memory.

Motivation

Scalability

- The system can be expanded by adding new experts without changing the existing structure.
- MoE allows for the distribution of work among experts, enhancing the ability to process large datasets and improve scalability.

MoE



- Simplified classification example for ME
- The blue circles and the red diamonds belong to classes 1 and 2, respectively, and they present a nonlinear classification

example

MoE

- Let $D = X, Y$ denote the data where $X = {x^{(n)}}_{n=1}^{N}$ is the input, $Y = {y^{(n)}}_{n=1}^{N}$ is the target, and N is the number of training points. Also, let $\theta = \{\theta_g, \theta_e\}$ denote the set of all parameters where θ_e is the set of expert parameters
- Given an input vector x and a target vector y, the total probability of observing y can be written in terms of the experts, as

$$
\begin{aligned}
P(y|x,\theta) &= \sum_{i=1}^{I} P(y,i|x,\theta) \\
&= \textstyle\sum_{i=1}^{I} P\left(i\middle|x,\theta_g\right) P(y|i,x,\theta_e) \\
&= \textstyle\sum_{i=1}^{I} g_i\left(x,\theta_g\right) P(y|i,x,\theta_e\ )
\end{aligned}
$$

MoE

$$P(y|x,\theta) = \sum_{i=1}^{I} P(y,i|x,\theta)$$
$$= \sum_{i=1}^{I} P\left(i\middle|x,\theta_g\right)P(y|i,x,\theta_e)$$
$$= \sum_{i=1}^{I} g_i\left(x,\theta_g\right)P(y|i,x,\theta_e)$$

- Where I is the number of experts, the function $g_i\left(x,\theta_g\right) = P\left(i\middle|x,\theta_g\right)$, represents the gate's rating, i.e., the probability of the ith expert given x, and $P(y|i,x,\theta_e)$ is the probability of the ith expert generating y given x. The latter will be denoted by $P_i(y)$ from now on.

MoE

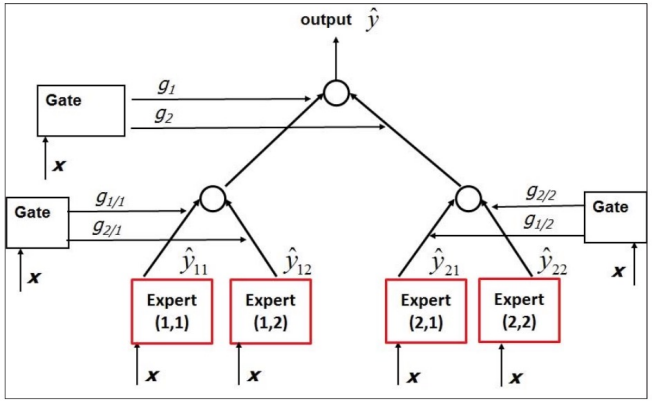
- During the training of ME, the gate and experts get decoupled, so the architecture attains a modular structure. Using this property

$$P(y|x,\theta) = \sum_{i=1}^{I} g_i\left(x,\theta_{g_i}\right) \sum_{j=1}^{J_i} g_{(j|i)}\left(x,\theta_{g_{(i|j)}}\right) P_{ij}(y,\theta_e)$$

Where I is the number of nodes connected to the gate at the top layer, and J_i is the number of nodes connected to the ith lower-level gating network, g_i is the output of the gate in the top layer, $g_{(j|i)}$ is the output of the jth gate connected to the ith gate of the top layer, and θ_{g_i} and $\theta_{g_{(i|j)}}$ are their parameters, respectively.

MoE

$$P(y|x,\theta) = \sum_{i=1}^{I} g_i\left(x,\theta_{g_i}\right) \sum_{j=1}^{J_i} g_{(j|i)}\left(x,\theta_{g_{(i|j)}}\right) P_{ij}(y,\theta_e)$$



MoE

For both classification and regression, the gate is defined by the softmax function

$$g_i(\mathbf{x}, \mathbf{v}) = \frac{\exp(\beta_i(\mathbf{x}, \mathbf{v}))}{\sum_{j=1}^{I} \exp(\beta_j(\mathbf{x}, \mathbf{v}))}$$

$$\beta_i(\mathbf{x}, \mathbf{v}) = \mathbf{v}_i^T[\mathbf{x}, 1].$$

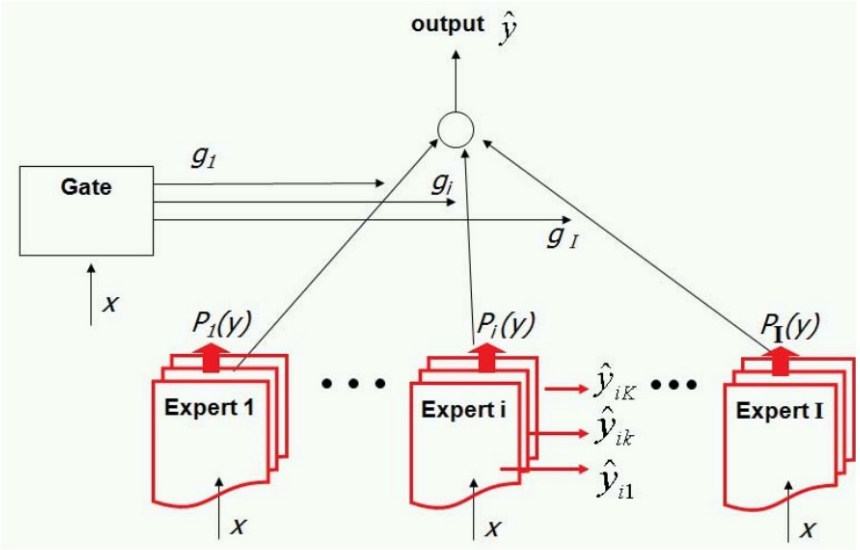
MoE Classification Model

K is the number of classes,
$\{\mathbf{w}_{ik}\}_{k=1}^{K}$ is parameter of expert ith

$$\hat{y}_{ik} = \frac{\exp(\mathbf{w}_{ik}^T[\mathbf{x}, 1])}{\sum_{r=1}^{K} \exp(\mathbf{w}_{ir}^T[\mathbf{x}, 1])}$$

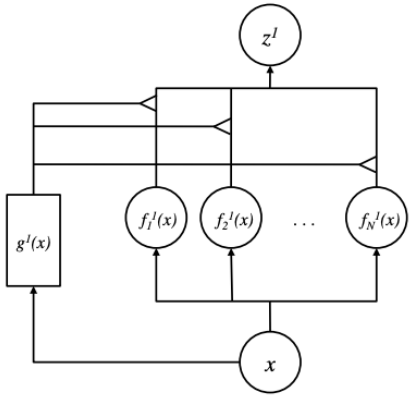
$$P_i(\mathbf{y}) = \prod_k \hat{y}_{ik}^{y_k}.$$

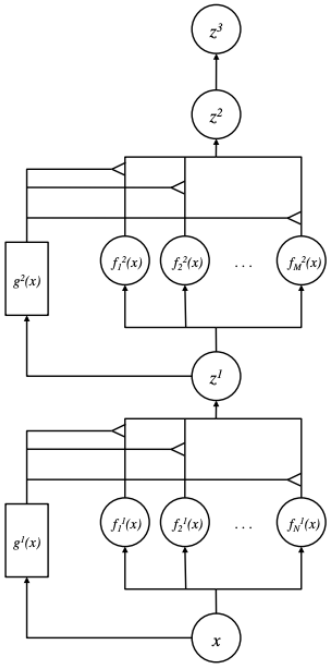
$$\hat{y}_k = \sum g_i(\mathbf{x}, \mathbf{v})\hat{y}_{ik}$$



Deep Mixture of Experts

$$
\begin{aligned}
z^1 &= \sum_{i=1}^{N} g_i^1(x) f_i^1(x) \\
z^2 &= \sum_{j=1}^{M} g_j^2(z^1) f_j^2(z^1) \\
F(x) = z^3 &= \mathrm{softmax}(f^3(z^2))
\end{aligned}
$$





Deep Mixture of Experts

Note:

The experts at each layer that perform best for the first few examples end up overpowering the remaining experts. This happens because the first examples increase the gating weights of these experts, which in turn causes them to be selected with high gating weights more frequently. This causes them to train more, and their gating weights to increase again, ad infinitum.

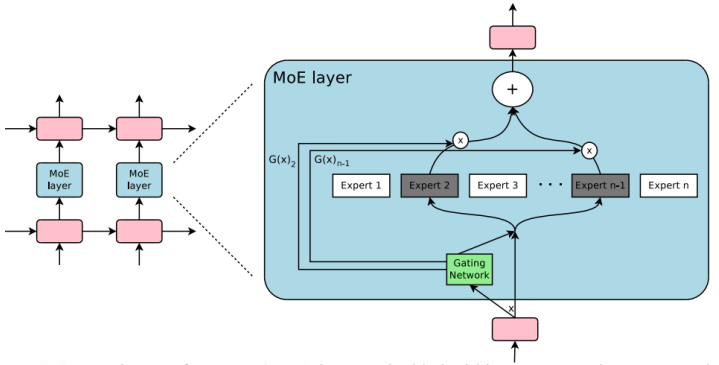
Let $G_i^l(t) = \sum_{(t'=1)}^{t} g_i^l(x_t')$ be the running total assignment to expert i of layer l at step t, and let $\overline{G^l(t)} = \frac{1}{N}\sum_{i=1}^{N} G_i^l(t)$ be their mean (here, $x_{t'}$ is the training example at step t'). Then for each expert i, we set $g_i^l(x_t) = 0$ if $G_i^l(t) - \overline{G^l(t)} > m$ for a margin threshold m.

THE SPARSELY GATED MIXTURE OF EXPERTS LAYER

$G(x)$ and $E_i(x)$ the output of the gating network and the output of the $i - th$ expert network for a given input x.

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

$$G_\sigma(x) = Softmax(x \cdot W_g)$$



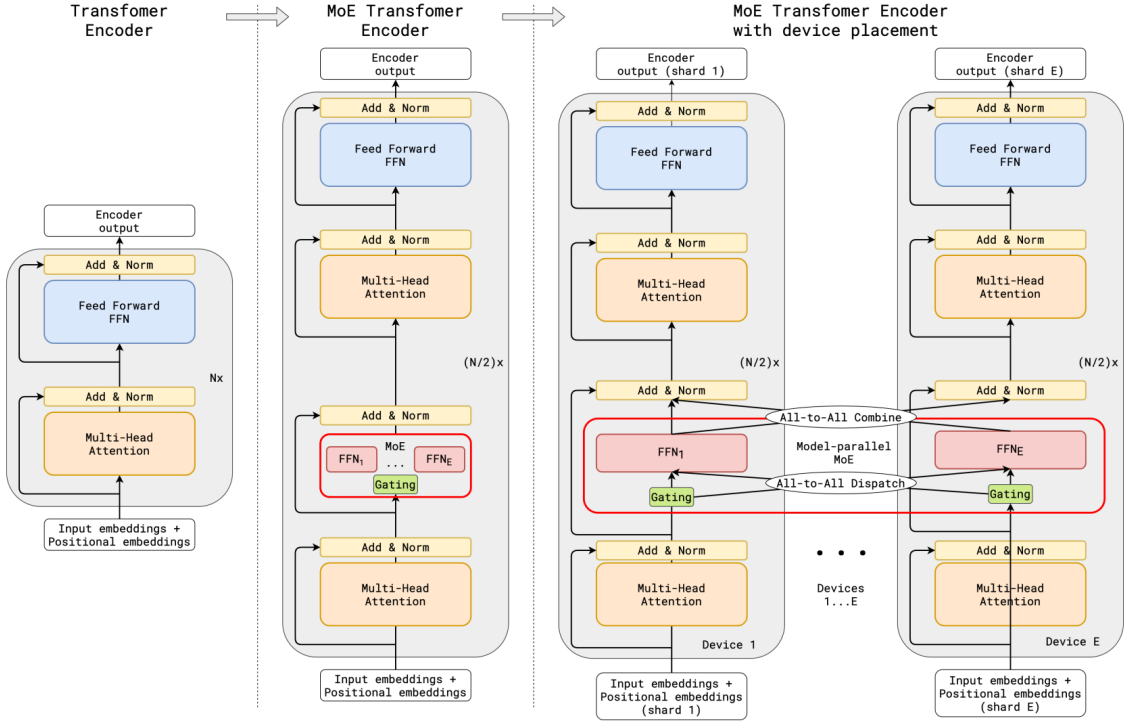
$$G(x) = Softmax(KeepTopK(H(x), k))$$

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

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

Shazeer, Noam, et al. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer." arXiv preprint arXiv:1701.06538 (2017)

GShards



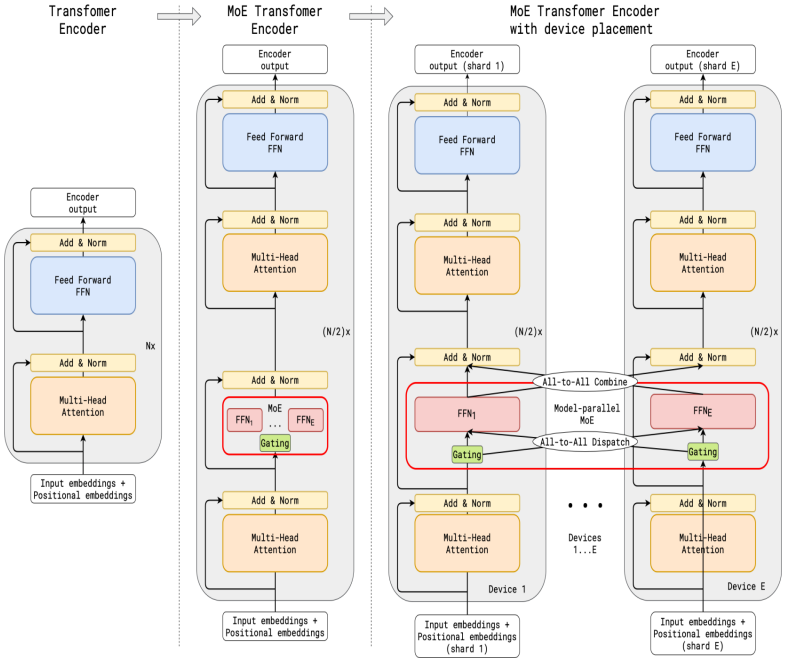
Lepikhin, Dmitry, et al. "Gshard: Scaling giant models with conditional computation and automatic sharding." arXiv preprint arXiv:2006.16668 (2020).

GShards

Algorithm 1: Group-level top-2 gating with auxiliary loss

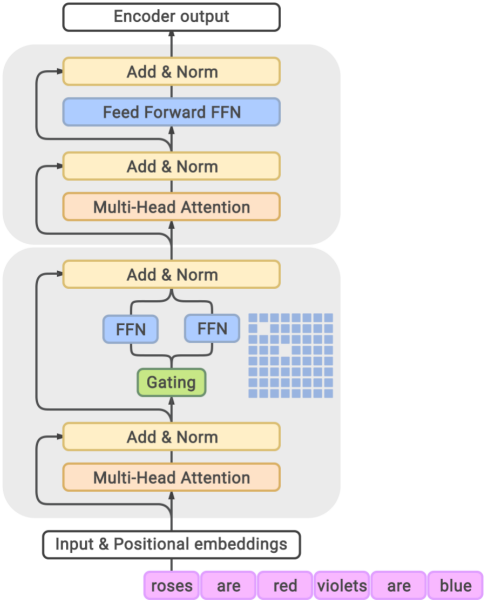
Data: x_S, a group of tokens of size S
Data: C, Expert capacity allocated to this group
Result: $\mathcal{G}_{S,E}$, group combine weights
Result: ℓ_{aux}, group auxiliary loss

(1) $c_E \leftarrow 0$ ▷ gating decisions per expert
(2) $g_{S,E} \leftarrow softmax(wg \cdot x_S)$ ▷ gates per token per expert, wg are trainable weights
(3) $m_E \leftarrow \frac{1}{S}\sum_{s=1}^{s} g_{s,E}$ ▷ mean gates per expert
(4) **for** $s \leftarrow 1$ **to** S **do**
(5) $\quad g1, e1, g2, e2 = top\_2(g_{s,E})$ ▷ top-2 gates and expert indices
(6) $\quad g1 \leftarrow g1/(g1+g2)$ ▷ normalized $g1$
(7) $\quad c \leftarrow c_{e1}$ ▷ position in $e1$ expert buffer
(8) $\quad$ **if** $c_{e1} < C$ **then**
(9) $\quad\quad \mathcal{G}_{s,e1} \leftarrow g1$ ▷ $e1$ expert combine weight for x_s
(10) $\quad$ **end**
(11) $\quad c_{e1} \leftarrow c+1$ ▷ incrementing $e1$ expert decisions count
(12) **end**
(13) $\ell_{aux} = \frac{1}{E}\sum_{e=1}^{E} \frac{c_e}{S} \cdot m_e$
(14) **for** $s \leftarrow 1$ **to** S **do**
(15) $\quad g1, e1, g2, e2 = top\_2(g_{s,E})$ ▷ top-2 gates and expert indices
(16) $\quad g2 \leftarrow g2/(g1+g2)$ ▷ normalized $g2$
(17) $\quad rnd \leftarrow uniform(0,1)$ ▷ dispatch to second-best expert with probability $\propto 2 \cdot g2$
(18) $\quad c \leftarrow c_{e2}$ ▷ position in $e2$ expert buffer
(19) $\quad$ **if** $c < C \wedge 2 \cdot g2 > rnd$ **then**
(20) $\quad\quad \mathcal{G}_{s,e2} \leftarrow g2$ ▷ $e2$ expert combine weight for x_s
(21) $\quad$ **end**
(22) $\quad c_{e2} \leftarrow c+1$
(23) **end**

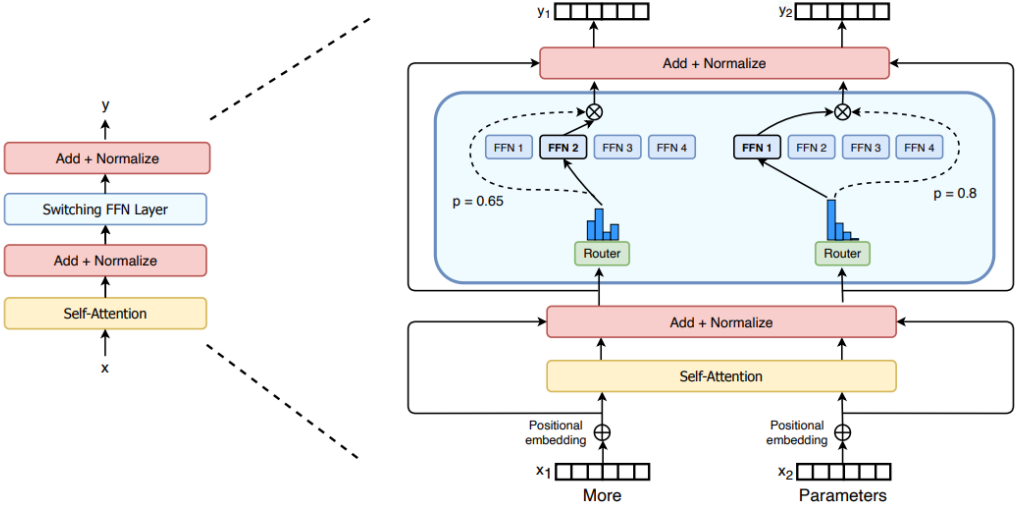


Lepikhin, Dmitry, et al. "Gshard: Scaling giant models with conditional computation and automatic sharding." arXiv preprint arXiv:2006.16668 (2020).

GLaM and SwitchTransformers



GLaM



Switch transformers

Du, Nan, et al. "Glam: Efficient scaling of language models with mixture-of-experts." International Conference on Machine Learning. PMLR, 2022.
Fedus, William, Barret Zoph, and Noam Shazeer. "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity." Journal of Machine Learning Research 23.120 (2022)