WIKIPEDIA
The Free Encyclopedia

Mixture of experts

Mixture of experts (**MoE**) is a machine learning technique where multiple expert networks (learners) are used to divide a problem space into homogeneous regions.[1] It differs from ensemble techniques in that for MoE, typically only one or a few expert models are run for each input, whereas in ensemble techniques, all models are run on every input.

Basic theory

In mixture of experts, we always have the following ingredients, but they are constructed and combined differently.

- There are experts $f_1, \dots, f_n$, each taking in the same input x, and produces outputs $f_1(x), \dots, f_n(x)$.
- There is a single weighting function (aka gating function) w, which takes in x and produces a vector of outputs $(w(x)_1, \dots, w(x)_n)$.
- $\theta = (\theta_0, \theta_1, \dots, \theta_n)$ is the set of parameters. The parameter θ_0 is for the weighting function.
- Given an input x, the mixture of experts produces a single combined output by combining $f_1(x), \dots, f_n(x)$ according to the weights $w(x)_1, \dots, w(x)_n$ in some way.

Both the experts and the weighting function are trained by minimizing some form of loss function, generally by gradient descent. There is a lot of freedom in choosing the precise form of experts, the weighting function, and the loss function.

Meta-pi network

The **meta-pi network**, reported by Hampshire and Waibel,[2] uses $f(x) = \sum_i w(x)_i f_i(x)$ as the output. The model is trained by performing gradient descent on the mean-squared error loss $L := \frac{1}{N} \sum_k \|y_k - f(x_k)\|^2$. The experts may be arbitrary functions.

In their original publication, they were solving the problem of classifying phonemes in speech signal from 6 different Japanese speakers, 2 females and 4 males. They trained 6 experts, each being a "time-delayed neural network"[3] (essentially a multilayered convolution network over the mel spectrogram). They found that the resulting mixture of experts dedicated 5 experts for 5 of the speakers, but the 6th (male) speaker does not have a dedicated expert, instead his voice was classified by a linear combination of the experts for the other 3 male speakers.

Adaptive mixtures of local experts

The **adaptive mixtures of local experts** [4][5] uses a gaussian mixture model. Each expert simply predicts a gaussian distribution, and totally ignores the input. Specifically, the i-th expert predicts that the output is $y \sim N(\mu_i, I)$, where μ_i is a learnable parameter. The weighting function is a linear-softmax function:

$$w(x)_i = \frac{e^{k_i^T x + b_i}}{\sum_j e^{k_j^T x + b_j}}$$

The mixture of experts predict that the output is distributed according to the probability density function:

$$f_\theta(y|x) = \ln\left[\sum_i \frac{e^{k_i^T x + b_i}}{\sum_j e^{k_j^T x + b_j}} N(y|\mu_i, I)\right] = \ln\left[(2\pi)^{-d/2} \sum_i \frac{e^{k_i^T x + b_i}}{\sum_j e^{k_j^T x + b_j}} e^{-\frac{1}{2}\|y-\mu_i\|^2}\right]$$

It is trained by maximal likelihood estimation, that is, gradient ascent on $f(y|x)$. The gradient for the i-th expert is

$$\nabla_{\mu_i} f_\theta(y|x) = \frac{w(x)_i N(y|\mu_i, I)}{\sum_j w(x)_j N(y|\mu_j, I)} (y - \mu_i)$$

and the gradient for the weighting function is

$$\nabla_{[k_i, b_i]} f_\theta(y|x) = \begin{bmatrix} x \\ 1 \end{bmatrix} \frac{w(x)_i}{\sum_j w(x)_j N(y|\mu_j, I)} (f_i(x) - f_\theta(y|x))$$

For each input-output pair (x, y), the weighting function is changed to increase the weight on all experts that performed above average, and decrease the weight on all experts that performed below average. This encourages the weighting function to learn to select only the experts that make the right predictions for each input.

The i-th expert is changed to make its prediction closer to y, but the amount of change is proportional to $w(x)_i N(y|\mu_i, I)$. This has a Bayesian interpretation. Given input x, the prior probability that expert i is the right one is $w(x)_i$, and $N(y|\mu_i, I)$ is the likelihood of evidence y. So, $\frac{w(x)_i N(y|\mu_i, I)}{\sum_j w(x)_j N(y|\mu_j, I)}$ is the posterior probability for expert i, and so the rate of change for the i-th expert is proportional to its posterior probability.

In words, the experts that, in hindsight, seemed like the good experts to consult, are asked to learn on the example. The experts that, in hindsight, were not, are left alone.

The combined effect is that the experts become specialized: Suppose two experts are both good at predicting a certain kind of input, but one is slightly better, then the weighting function would eventually learn to favor the better one. After that happens, the lesser expert is unable to obtain a high gradient signal, and becomes even worse at predicting such kind of input. Conversely, the lesser expert can become better at predicting other kinds of input, and increasingly pulled away into another region. This has a positive feedback effect, causing each expert to move apart from the rest and take care of a local region alone (thus the name "*local* experts").

Hierarchical MoE

Hierarchical mixtures of experts[6][7] uses multiple levels of gating in a tree. Each gating is a probability distribution over the next level of gatings, and the experts are on the leaf nodes of the tree. They are similar to decision trees.

For example, a 2-level hierarchical MoE would have a first order gating function w_i, and second order gating functions $w_{j|i}$ and experts $f_{j|i}$. The total prediction is then $\sum_i w_i(x) \sum_j w_{j|i}(x) f_{j|i}(x)$.

Variants

The mixture of experts, being similar to the gaussian mixture model, can also be trained by the expectation-maximization algorithm, just like gaussian mixture models. Specifically, during the expectation step, the "burden" for explaining each data point is assigned over the experts, and during the maximization step, the experts are trained to improve the explanations they got a high burden for, while the gate is trained to improve its burden assignment. This can converge faster than gradient ascent on the log-likelihood.[7][8]

The choice of gating function is often a softmax gating. Other than that, [9] proposed using gaussian distributions, and [8] proposed using exponential families.

Instead of performing a weighted sum of all the experts, in hard MoE [10] only the highest ranked expert is chosen. That is, $f(x) = f_{\arg\max_i w_i(x)}(x)$. This can accelerate training and inference time.[11]

The experts can use more general forms of multivariant gaussian distributions. For example, [6] proposed $f_i(y|x) = N(y|A_i x + b_i, \Sigma_i)$, where A_i, b_i, Σ_i are learnable parameters. In words, each expert learns to do linear regression, with a learnable uncertainty estimate.

One can use different experts than gaussian distributions. For example, one can use Laplace distribution,[12] or Student's t-distribution.[13] For binary classification, it also proposed logistic regression experts, with

$$f_i(y|x) = \begin{cases} \frac{1}{1+e^{\beta_i^T x + \beta_{i,0}}}, & y = 0 \\ 1 - \frac{1}{1+e^{\beta_i^T x + \beta_{i,0}}}, & y = 1 \end{cases}$$

where $\beta_i, \beta_{i,0}$ are learnable parameters. This is later generalized for multi-class classification, with multinomial logistic regression experts.[14]

Deep learning

The previous section described MoE as it was used before the era of deep learning. After deep learning, MoE found applications in running the largest models, as a simple way to perform *conditional computation*: only parts of the model are used, the parts chosen according to what the input is.[15]

The earliest paper that applies MoE to deep learning is "Learning Factored Representations in a Deep Mixture of Experts" (Eigen, Ranzato, Sutskever) [16] which proposes to use a different gating network at each layer in a deep neural network. Specifically, each gating is a linear-ReLU-linear-softmax network, and each expert is a linear-ReLU network.

The key design desideratum for MoE in deep learning is to reduce computing cost. Consequently, for each query, only a small subset of the experts should be queried. This makes MoE in deep learning different from classical MoE. In classical MoE, the output for each query is a weighted sum of *all* experts' outputs. In deep learning MoE, the output for each query can only involve a few experts' outputs. Consequently, the key design choice in MoE becomes routing: given a batch of queries, how to route the queries to the best experts.

Sparsely-gated MoE layer

The **sparsely-gated MoE layer**,[17] published by researchers from Google Brain, uses feedforward networks as experts, and linear-softmax gating. Similar to the previously proposed hard MoE, they achieve sparsity by a weighted sum of only the top-k experts, instead of the weighted sum of all of them. Specifically, in a MoE layer, there are feedforward networks $f_1, \ldots, f_n$, and a gating network w. The gating network is defined by $w(x) = \mathrm{softmax}(\mathrm{top}_k(Wx + \text{noise}))$, where top_k is a function that keeps the top-k entries of a vector the same, but sets all other entries to $-\infty$. The addition of noise helps with load balancing.

The choice of k is a hyperparameter that is chosen according to application. Typical values are $k = 1, 2$. The $k = 1$ version is also called the Switch Transformer.[18]

As demonstration, they trained a series of models for machine translation with alternating layers of MoE and LSTM, and compared with deep LSTM models.[19] Table 3 shows that the MoE models used less inference time compute, despite having 30x more parameters.

Vanilla MoE tend to have issues of load balancing: some experts are consulted often, while other experts rarely or not at all. To encourage the gate to select each expert with equal frequency (proper load balancing) within each batch, each MoE layer has two auxiliary loss functions. This is improved by [18] into a single auxiliary loss function. Specifically, let n be the number of experts, then for a given batch of queries $\{x_1, x_2, \ldots, x_T\}$, the auxiliary loss for the batch is

$$n \sum_{i=1}^{n} f_i P_i$$

Here, $f_i = \frac{1}{T}\#(\text{queries sent to expert } i)$ is the fraction of time where expert i is ranked highest, and $P_i = \frac{1}{T}\sum_{j=1}^{T} w_i(x_j)$ is the fraction of weight on expert i. This loss is minimized at 1, precisely when every expert has equal weight $1/n$ in all situations.

Routing

In sparsely-gated MoE, only the top-k experts are queried, and their outputs are weighted-summed. There are other methods.[20]

In Hash MoE,[21] routing is performed deterministically by a hash function, fixed before learning begins. For example, if the model is a 4-layered Transformer, and input is a token for word "eat", and the hash of "eat" is $(1, 4, 2, 3)$, then the token would be routed to the 1st expert in layer 1, 4th expert in layer 2, etc. Despite its simplicity, it achieves competitive performance as sparsely gated MoE with $k = 1$.

In soft MoE, suppose in each batch, each expert can process p queries, then there are $n \times p$ queries that can be assigned per batch. Now for each batch of queries $\{x_1, x_2, \dots, x_T\}$, the soft MoE layer computes an array $w_{i,j,k}$, such that $(w_{i,j,1}, \dots, w_{i,j,T})$ is a probability distribution over queries, and the i-th expert's j-th query is $\sum_k w_{i,j,k} x_k$.[22] However, this does not work with autoregressive modelling, since the weights $w_{i,j,k}$ over one token depends on all other tokens'.[23]

Other approaches include solving it as a constrained linear programming problem,[24] making each expert choose the top-k queries it wants (instead of each query choosing the top-k experts for it),[25] using reinforcement learning to train the routing algorithm (since picking an expert is a discrete action, like in RL).[26]

Capacity factor

Suppose there are n experts in a layer. For a given batch of queries $\{x_1, x_2, \dots, x_T\}$, each query is routed to one or more experts. For example, if each query is routed to one expert as in Switch Transformers, and if the experts are load-balanced, then each expert should expect on average T/n queries in a batch. In practice, the experts cannot expect perfect load balancing: in some batches, one expert might be underworked, while in other batches, it would be overworked.

Since the inputs cannot move through the layer until every expert in the layer has finished the queries it is assigned, load balancing is important. As a hard constraint on load balancing, there is the **capacity factor**: each expert is only allowed to process up to $c \cdot T/n$ queries in a batch. [20] found $c \in [1.25, 2]$ to work in practice.

Applications to transformer models

MoE layers are used in the largest transformer models, for which learning and inferring over the full model is too costly. They are typically sparsely-gated, with sparsity 1 or 2. In Transformer models, the MoE layers are often used to select the feedforward layers (typically a linear-ReLU-linear network), appearing in each Transformer block after the multiheaded attention. This is because the feedforward layers take up an increasing portion of the computing cost as models grow larger. For example, in the Palm-540B model, 90% of parameters are in its feedforward layers.[27]

As of 2023, models large enough to use MoE tend to be large language models, where each expert has on the order of 10 billion parameters. Other than language models, Vision MoE[28] is a Transformer model with MoE layers. They demonstrated it by training a model with 15 billion parameters.

A series of large language models from Google used MoE. GShard[29] uses MoE with up to top-2 experts per layer. Specifically, the top-1 expert is always selected, and the top-2th expert is selected with probability proportional to that experts' weight according to the gating function. Later, GLaM[30] demonstrated a language model with 1.2 trillion parameters, each MoE layer using top-2 out of 64 experts. Switch Transformers[18] use top-1 in all MoE layers.

The NLLB-200 by Meta AI is a machine translation model for 200 languages.[31] Each MoE layer uses a hierarchical MoE with two levels. On the first level, the gating function chooses to use either a "shared" feedforward layer, or to use the experts. If using the experts, then another gating function computes the weights and chooses the top-2 experts.[32]

MoE large language models can be adapted for downstream tasks by instruction tuning.[33]

In December 2023, Mistral AI released Mixtral 8x7B under Apache 2.0 license. It is a MoE language model with 46.7B parameters, 8 experts, and sparsity 2. They also released a version finetuned for instruction following.[34][35]

In March 2024, Databricks released DBRX. It is a MoE language model with 132B parameters, 16 experts, and sparsity 4. They also released a version finetuned for instruction following.[36][37]

Further reading

- Before deep learning era
  - McLachlan, Geoffrey J.; Peel, David (2000). *Finite mixture models*. Wiley series in probability and statistics applied probability and statistics section. New York Chichester Weinheim Brisbane Singapore Toronto: John Wiley & Sons, Inc. ISBN 978-0-471-00626-8.
  - Yuksel, S. E.; Wilson, J. N.; Gader, P. D. (August 2012). "Twenty Years of Mixture of Experts" (https://ieeexplore.ieee.org/document/6215056). *IEEE Transactions on Neural Networks and Learning Systems*. **23** (8): 1177–1193. doi:10.1109/TNNLS.2012.2200299 (https://doi.org/10.1109%2FTNNLS.2012.2200299). ISSN 2162-237X (https://www.worldcat.org/issn/2162-237X). PMID 24807516 (https://pubmed.ncbi.nlm.nih.gov/24807516). S2CID 9922492 (https://api.semanticscholar.org/CorpusID:9922492).
  - Masoudnia, Saeed; Ebrahimpour, Reza (12 May 2012). "Mixture of experts: a literature survey". *Artificial Intelligence Review*. **42** (2): 275–293. doi:10.1007/s10462-012-9338-y (https://doi.org/10.1007%2Fs10462-012-9338-y). S2CID 3185688 (https://api.semanticscholar.org/CorpusID:3185688).

- Nguyen, Hien D.; Chamroukhi, Faicel (July 2018). "Practical and theoretical aspects of mixture-of-experts modeling: An overview" (https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1246). *WIREs Data Mining and Knowledge Discovery*. **8** (4). doi:10.1002/widm.1246 (https://doi.org/10.1002%2Fwidm.1246). ISSN 1942-4787 (https://www.worldcat.org/issn/1942-4787). S2CID 49301452 (https://api.semanticscholar.org/CorpusID:49301452).
- Deep learning era
  - Zoph, Barret; Bello, Irwan; Kumar, Sameer; Du, Nan; Huang, Yanping; Dean, Jeff; Shazeer, Noam; Fedus, William (2022). "ST-MoE: Designing Stable and Transferable Sparse Expert Models". arXiv:2202.08906 (https://arxiv.org/abs/2202.08906) [cs.CL (https://arxiv.org/archive/cs.CL)].

See also

- Product of experts
- Mixture models
- Mixture of gaussians
- Ensemble learning

References

1. Baldacchino, Tara; Cross, Elizabeth J.; Worden, Keith; Rowson, Jennifer (2016). "Variational Bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems". *Mechanical Systems and Signal Processing*. 66–67: 178–200. Bibcode:2016MSSP...66..178B (https://ui.adsabs.harvard.edu/abs/2016MSSP...66..178B). doi:10.1016/j.ymssp.2015.05.009 (https://doi.org/10.1016%2Fj.ymssp.2015.05.009).
2. Hampshire, J.B.; Waibel, A. (July 1992). "The Meta-Pi network: building distributed knowledge representations for robust multisource pattern recognition" (https://isl.anthropomatik.kit.edu/downloads/The_Meta-Pi_Network-_Building_Distributed_Knowledge_Representations_for_Robust_Multi-Source_Pattern_Recognition.pdf) (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **14** (7): 751–769. doi:10.1109/34.142911 (https://doi.org/10.1109%2F34.142911).
3. Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, Kevin J. Lang (1995). "Phoneme Recognition Using Time-Delay Neural Networks*" (https://www.taylorfrancis.com/chapters/edit/10.4324/9780203763247-2/phoneme-recognition-using-time-delay-neural-networks-alexander-waibel-toshiyuki-hanazawa-geoffrey-hinton-kiyohiro-shikano-kevin-lang). In Chauvin, Yves; Rumelhart, David E. (eds.). *Backpropagation*. Psychology Press. doi:10.4324/9780203763247 (https://doi.org/10.4324%2F9780203763247). ISBN 978-0-203-76324-7.
4. Nowlan, Steven; Hinton, Geoffrey E (1990). "Evaluation of Adaptive Mixtures of Competing Experts" (https://proceedings.neurips.cc/paper/1990/hash/432aca3a1e345e339f35a30c8f65edce-Abstract.html). *Advances in Neural Information Processing Systems*. **3**. Morgan-Kaufmann.
5. Jacobs, Robert A.; Jordan, Michael I.; Nowlan, Steven J.; Hinton, Geoffrey E. (February 1991). "Adaptive Mixtures of Local Experts" (https://direct.mit.edu/neco/article/3/1/79-87/5560). *Neural Computation*. **3** (1): 79–87. doi:10.1162/neco.1991.3.1.79 (https://doi.org/10.1162%2Fneco.1991.3.1.79). ISSN 0899-7667 (https://www.worldcat.org/issn/0899-7667). PMID 31141872 (https://pubmed.ncbi.nlm.nih.gov/31141872). S2CID 572361 (https://api.semanticscholar.org/CorpusID:572361).
6. Jordan, Michael; Jacobs, Robert (1991). "Hierarchies of adaptive experts" (https://proceedings.neurips.cc/paper_files/paper/1991/hash/59b90e1005a220e2ebc542eb9d950b1e-Abstract.html). *Advances in Neural Information Processing Systems*. **4**. Morgan-Kaufmann.

7. Jordan, Michael I.; Jacobs, Robert A. (March 1994). "Hierarchical Mixtures of Experts and the EM Algorithm" (https://direct.mit.edu/neco/article/6/2/181-214/5779). *Neural Computation*. **6** (2): 181–214. doi:10.1162/neco.1994.6.2.181 (https://doi.org/10.1162%2Fneco.1994.6.2.181). hdl:1721.1/7206 (https://hdl.handle.net/1721.1%2F7206). ISSN 0899-7667 (https://www.worldcat.org/issn/0899-7667).
8. Jordan, Michael I.; Xu, Lei (1995-01-01). "Convergence results for the EM approach to mixtures of experts architectures" (https://dx.doi.org/10.1016/0893-6080%2895%2900014-3). *Neural Networks*. **8** (9): 1409–1431. doi:10.1016/0893-6080(95)00014-3 (https://doi.org/10.1016%2F0893-6080%2895%2900014-3). hdl:1721.1/6620 (https://hdl.handle.net/1721.1%2F6620). ISSN 0893-6080 (https://www.worldcat.org/issn/0893-6080).
9. Xu, Lei; Jordan, Michael; Hinton, Geoffrey E (1994). "An Alternative Model for Mixtures of Experts" (https://proceedings.neurips.cc/paper/1994/hash/c8fbbc86abe8bd6a5eb6a3b4d0411301-Abstract.html). *Advances in Neural Information Processing Systems*. **7**. MIT Press.
10. Collobert, Ronan; Bengio, Samy; Bengio, Yoshua (2001). "A Parallel Mixture of SVMs for Very Large Scale Problems" (https://proceedings.neurips.cc/paper_files/paper/2001/hash/36ac8e558ac7690b6f44e2cb5ef93322-Abstract.html). *Advances in Neural Information Processing Systems*. **14**. MIT Press.
11. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "12: Applications". *Deep learning*. Adaptive computation and machine learning. Cambridge, Mass: The MIT press. ISBN 978-0-262-03561-3.
12. Nguyen, Hien D.; McLachlan, Geoffrey J. (2016-01-01). "Laplace mixture of linear experts" (https://www.sciencedirect.com/science/article/pii/S0167947314003089). *Computational Statistics & Data Analysis*. **93**: 177–191. doi:10.1016/j.csda.2014.10.016 (https://doi.org/10.1016%2Fj.csda.2014.10.016). ISSN 0167-9473 (https://www.worldcat.org/issn/0167-9473).
13. Chamroukhi, F. (2016-07-01). "Robust mixture of experts modeling using the t distribution" (https://www.sciencedirect.com/science/article/pii/S0893608016000435). *Neural Networks*. **79**: 20–36. arXiv:1701.07429 (https://arxiv.org/abs/1701.07429). doi:10.1016/j.neunet.2016.03.002 (https://doi.org/10.1016%2Fj.neunet.2016.03.002). ISSN 0893-6080 (https://www.worldcat.org/issn/0893-6080). PMID 27093693 (https://pubmed.ncbi.nlm.nih.gov/27093693). S2CID 3171144 (https://api.semanticscholar.org/CorpusID:3171144).
14. Chen, K.; Xu, L.; Chi, H. (1999-11-01). "Improved learning algorithms for mixture of experts in multiclass classification" (https://www.sciencedirect.com/science/article/pii/S089360809900043X). *Neural Networks*. **12** (9): 1229–1252. doi:10.1016/S0893-6080(99)00043-X (https://doi.org/10.1016%2FS0893-6080%2899%2900043-X). ISSN 0893-6080 (https://www.worldcat.org/issn/0893-6080). PMID 12662629 (https://pubmed.ncbi.nlm.nih.gov/12662629).
15. Bengio, Yoshua; Léonard, Nicholas; Courville, Aaron (2013). "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". arXiv:1308.3432 (https://arxiv.org/abs/1308.3432) [cs.LG (https://arxiv.org/archive/cs.LG)].
16. Eigen, David; Ranzato, Marc'Aurelio; Sutskever, Ilya (2013). "Learning Factored Representations in a Deep Mixture of Experts". arXiv:1312.4314 (https://arxiv.org/abs/1312.4314) [cs.LG (https://arxiv.org/archive/cs.LG)].
17. Shazeer, Noam; Mirhoseini, Azalia; Maziarz, Krzysztof; Davis, Andy; Le, Quoc; Hinton, Geoffrey; Dean, Jeff (2017). "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". arXiv:1701.06538 (https://arxiv.org/abs/1701.06538) [cs.LG (https://arxiv.org/archive/cs.LG)].
18. Fedus, William; Zoph, Barret; Shazeer, Noam (2022-01-01). "Switch transformers: scaling to trillion parameter models with simple and efficient sparsity" (https://dl.acm.org/doi/abs/10.5555/3586589.3586709). *The Journal of Machine Learning Research*. **23** (1): 5232–5270. arXiv:2101.03961 (https://arxiv.org/abs/2101.03961). ISSN 1532-4435 (https://www.worldcat.org/issn/1532-4435).

19. Wu, Yonghui; Schuster, Mike; Chen, Zhifeng; Le, Quoc V.; Norouzi, Mohammad; Macherey, Wolfgang; Krikun, Maxim; Cao, Yuan; Gao, Qin; Macherey, Klaus; Klingner, Jeff; Shah, Apurva; Johnson, Melvin; Liu, Xiaobing; Kaiser, Łukasz (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". arXiv:1609.08144 (https://arxiv.org/abs/1609.08144) [cs.CL (https://arxiv.org/archive/cs.CL)].
20. Zoph, Barret; Bello, Irwan; Kumar, Sameer; Du, Nan; Huang, Yanping; Dean, Jeff; Shazeer, Noam; Fedus, William (2022). "ST-MoE: Designing Stable and Transferable Sparse Expert Models". arXiv:2202.08906 (https://arxiv.org/abs/2202.08906) [cs.CL (https://arxiv.org/archive/cs.CL)].
21. Roller, Stephen; Sukhbaatar, Sainbayar; szlam, arthur; Weston, Jason (2021). "Hash Layers For Large Sparse Models" (https://proceedings.neurips.cc/paper_files/paper/2021/hash/92bf5e6240737e0326ea59846a83e076-Abstract.html). *Advances in Neural Information Processing Systems*. **34**. Curran Associates: 17555–17566. arXiv:2106.04426 (https://arxiv.org/abs/2106.04426).
22. Puigcerver, Joan; Riquelme, Carlos; Mustafa, Basil; Houlsby, Neil (2023). "From Sparse to Soft Mixtures of Experts". arXiv:2308.00951 (https://arxiv.org/abs/2308.00951) [cs.LG (https://arxiv.org/archive/cs.LG)].
23. Wang, Phil (2023-10-04), *lucidrains/soft-moe-pytorch* (https://github.com/lucidrains/soft-moe-pytorch), retrieved 2023-10-08
24. Lewis, Mike; Bhosale, Shruti; Dettmers, Tim; Goyal, Naman; Zettlemoyer, Luke (2021-07-01). "BASE Layers: Simplifying Training of Large, Sparse Models" (https://proceedings.mlr.press/v139/lewis21a.html). *Proceedings of the 38th International Conference on Machine Learning*. PMLR: 6265–6274. arXiv:2103.16716 (https://arxiv.org/abs/2103.16716).
25. Zhou, Yanqi; Lei, Tao; Liu, Hanxiao; Du, Nan; Huang, Yanping; Zhao, Vincent; Dai, Andrew M.; Chen, Zhifeng; Le, Quoc V.; Laudon, James (2022-12-06). "Mixture-of-Experts with Expert Choice Routing" (https://proceedings.neurips.cc/paper_files/paper/2022/hash/2f00ecd787b432c1d36f3de9800728eb-Abstract-Conference.html). *Advances in Neural Information Processing Systems*. **35**: 7103–7114. arXiv:2202.09368 (https://arxiv.org/abs/2202.09368).
26. Bengio, Emmanuel; Bacon, Pierre-Luc; Pineau, Joelle; Precup, Doina (2015). "Conditional Computation in Neural Networks for faster models". arXiv:1511.06297 (https://arxiv.org/abs/1511.06297) [cs.LG (https://arxiv.org/archive/cs.LG)].
27. "Transformer Deep Dive: Parameter Counting" (https://orenleung.com/transformer-parameter-counting). *Transformer Deep Dive: Parameter Counting*. Retrieved 2023-10-10.
28. Riquelme, Carlos; Puigcerver, Joan; Mustafa, Basil; Neumann, Maxim; Jenatton, Rodolphe; Susano Pinto, André; Keysers, Daniel; Houlsby, Neil (2021). "Scaling Vision with Sparse Mixture of Experts" (https://proceedings.neurips.cc/paper/2021/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html). *Advances in Neural Information Processing Systems*. **34**: 8583–8595. arXiv:2106.05974 (https://arxiv.org/abs/2106.05974).
29. Lepikhin, Dmitry; Lee, HyoukJoong; Xu, Yuanzhong; Chen, Dehao; Firat, Orhan; Huang, Yanping; Krikun, Maxim; Shazeer, Noam; Chen, Zhifeng (2020). "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding". arXiv:2006.16668 (https://arxiv.org/abs/2006.16668) [cs.CL (https://arxiv.org/archive/cs.CL)].
30. Du, Nan; Huang, Yanping; Dai, Andrew M.; Tong, Simon; Lepikhin, Dmitry; Xu, Yuanzhong; Krikun, Maxim; Zhou, Yanqi; Yu, Adams Wei; Firat, Orhan; Zoph, Barret; Fedus, Liam; Bosma, Maarten; Zhou, Zongwei; Wang, Tao (2021). "GLaM: Efficient Scaling of Language Models with Mixture-of-Experts". arXiv:2112.06905 (https://arxiv.org/abs/2112.06905) [cs.CL (https://arxiv.org/archive/cs.CL)].
31. "200 languages within a single AI model: A breakthrough in high-quality machine translation" (https://web.archive.org/web/20230109051700/https://ai.facebook.com/blog/nllb-200-high-quality-machine-translation/). *ai.facebook.com*. 2022-06-19. Archived from the original (https://ai.facebook.com/blog/nllb-200-high-quality-machine-translation/) on 2023-01-09.

32. NLLB Team; Costa-jussà, Marta R.; Cross, James; Çelebi, Onur; Elbayad, Maha; Heafield, Kenneth; Heffernan, Kevin; Kalbassi, Elahe; Lam, Janice; Licht, Daniel; Maillard, Jean; Sun, Anna; Wang, Skyler; Wenzek, Guillaume; Youngblood, Al (2022). "No Language Left Behind: Scaling Human-Centered Machine Translation". arXiv:2207.04672 (https://arxiv.org/abs/2207.04672) [cs.CL (https://arxiv.org/archive/cs.CL)].
33. Shen, Sheng; Hou, Le; Zhou, Yanqi; Du, Nan; Longpre, Shayne; Wei, Jason; Chung, Hyung Won; Zoph, Barret; Fedus, William; Chen, Xinyun; Vu, Tu; Wu, Yuexin; Chen, Wuyang; Webson, Albert; Li, Yunxuan (2023). "Mixture-of-Experts Meets Instruction Tuning:A Winning Combination for Large Language Models". arXiv:2305.14705 (https://arxiv.org/abs/2305.14705) [cs.CL (https://arxiv.org/archive/cs.CL)].
34. AI, Mistral (2023-12-11). "Mixtral of experts" (https://mistral.ai/news/mixtral-of-experts/). *mistral.ai*. Retrieved 2024-02-04.
35. Jiang, Albert Q.; Sablayrolles, Alexandre; Roux, Antoine; Mensch, Arthur; Savary, Blanche; Bamford, Chris; Chaplot, Devendra Singh; Casas, Diego de las; Hanna, Emma Bou (2024-01-08), *Mixtral of Experts* (http://arxiv.org/abs/2401.04088), arXiv:2401.04088 (https://arxiv.org/abs/2401.04088), retrieved 2024-02-04
36. "Introducing DBRX: A New State-of-the-Art Open LLM" (https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm). *Databricks*. 2024-03-27. Retrieved 2024-03-28.
37. Knight, Will. "Inside the Creation of the World's Most Powerful Open Source AI Model" (https://www.wired.com/story/dbrx-inside-the-creation-of-the-worlds-most-powerful-open-source-ai-model/). *Wired*. ISSN 1059-1028 (https://www.worldcat.org/issn/1059-1028). Retrieved 2024-03-28.
