Title: Mixture of experts
URL: https://en.wikipedia.org/wiki/Mixture_of_experts
PageID: 54238535
Categories: Category:Machine learning algorithms

Supervised learning
Unsupervised learning
Semi-supervised learning
Self-supervised learning
Reinforcement learning
Meta-learning
Online learning
Batch learning
Curriculum learning
Rule-based learning
Neuro-symbolic AI
Neuromorphic engineering
Quantum machine learning
Classification
Generative modeling
Regression
Clustering
Dimensionality reduction
Density estimation
Anomaly detection
Data cleaning
AutoML
Association rules
Semantic analysis
Structured prediction
Feature engineering
Feature learning
Learning to rank
Grammar induction
Ontology learning
Multimodal learning

Apprenticeship learning

Decision trees

Ensembles Bagging Boosting Random forest

Bagging

Boosting

Random forest

k -NN

Linear regression

Naive Bayes

Artificial neural networks

Logistic regression

Perceptron

Relevance vector machine (RVM)

Support vector machine (SVM)

BIRCH

CURE

Hierarchical

k -means

Fuzzy

Expectation–maximization (EM)

DBSCAN

OPTICS

Mean shift

Factor analysis

CCA

ICA

LDA

NMF

PCA

PGD

t-SNE

SDL

Graphical models Bayes net Conditional random field Hidden Markov

Bayes net

Conditional random field

Hidden Markov

RANSAC

k -NN

Local outlier factor

Isolation forest

Autoencoder

Deep learning

Feedforward neural network

Recurrent neural network LSTM GRU ESN reservoir computing

LSTM

GRU

ESN

reservoir computing

Boltzmann machine Restricted

Restricted

GAN

Diffusion model

SOM

Convolutional neural network U-Net LeNet AlexNet DeepDream

U-Net

LeNet

AlexNet

DeepDream

Neural field Neural radiance field Physics-informed neural networks

Neural radiance field

Physics-informed neural networks

Transformer Vision

Vision

Mamba

Spiking neural network

Memtransistor

Electrochemical RAM (ECRAM)

Q-learning

Policy gradient

SARSA

Temporal difference (TD)

Multi-agent Self-play

Self-play

Active learning

Crowdsourcing

Human-in-the-loop

Mechanistic interpretability

RLHF

Coefficient of determination

Confusion matrix

Learning curve

ROC curve

Kernel machines

Bias–variance tradeoff

Computational learning theory

Empirical risk minimization

Occam learning

PAC learning

Statistical learning

VC theory

Topological deep learning

AAAI

ECML PKDD

NeurIPS

ICML

ICLR

IJCAI

ML

JMLR

Glossary of artificial intelligence

List of datasets for machine-learning research List of datasets in computer vision and image processing

List of datasets in computer vision and image processing

Outline of machine learning

v

t

e

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] MoE represents a form of ensemble learning . [2] They were also called committee machines . [3]

Basic theory

MoE always has the following components, but they are implemented and combined differently according to the problem being solved:

Experts f 1 , . . . , f n {\displaystyle f_{1},...,f_{n}} , each taking the same input x {\displaystyle x} , and producing outputs f 1 (x) , . . . , f n (x) {\displaystyle f_{1}(x),...,f_{n}(x)} .

A weighting function (also known as a gating function) w {\displaystyle w} , which takes input x {\displaystyle x} and produces a vector of outputs (w (x) 1 , . . . , w (x) n) {\displaystyle (w(x)_{1},...,w(x)_{n})} . This may or may not be a probability distribution, but in both cases, its entries are non-negative.

θ = (θ 0 , θ 1 , . . . , θ n) {\displaystyle \theta =(\theta _{0},\theta _{1},...,\theta _{n})} is the set of parameters. The parameter θ 0 {\displaystyle \theta _{0}} is for the weighting function. The parameters θ 1 , … , θ n {\displaystyle \theta _{1},\dots ,\theta _{n}} are for the experts.

Given an input x {\displaystyle x} , the mixture of experts produces a single output by combining f 1 (x) , . . . , f n (x) {\displaystyle f_{1}(x),...,f_{n}(x)} according to the weights w (x) 1 , . . . , w (x) n {\displaystyle w(x)_{1},...,w(x)_{n}} in some way, usually by f (x) = ∑ i w (x) i f i (x) {\displaystyle f(x)=\sum _{i}w(x)_{i}f_{i}(x)} .

Both the experts and the weighting function are trained by minimizing some loss function , generally via gradient descent . There is much 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, [4] uses f (x) = ∑ i w (x) i f i (x) {\displaystyle 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 := 1 N ∑ k ■ y k – f (x k) ■ 2 {\displaystyle 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" [5] (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 [6] [7] uses a Gaussian mixture model . Each expert simply predicts a Gaussian distribution, and totally ignores the input. Specifically, the i {\displaystyle i} -th expert predicts that the output is y ~ N (μ i , I) {\displaystyle y\sim N(\mu _{i},I)} , where μ i {\displaystyle \mu _{i}} is a learnable parameter. The weighting function is a linear-softmax function: w (x) i = e k i T x + b i ∑ j e k j T x + b j {\displaystyle 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 log-probability density function: ln ■ f θ (y | x) = ln ■ [∑ i e k i T x + b i ∑ j e k j T x + b j N (y | μ i , I)] = ln ■ [(2 π) – d / 2 ∑ i e k i T x + b i ∑ j e k j T x + b j e – 1 2 ■ y – μ i ■ 2] {\displaystyle \ln 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) {\displaystyle f(y|x)} . The gradient for the i {\displaystyle i} -th expert is

∇ μ i ln ■ f θ (y | x) = w (x) i N (y | μ i , I) ∑ j w (x) j N (y | μ j , I) (y – μ i) {\displaystyle \nabla _{\mu _{i}}\ln 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 ∇ [k i , b i] ln ■ f θ (y | x) = [x 1] w (x) i ∑ j w (x) j N (y | μ j , I) (f i (x) – f θ (y | x)) {\displaystyle \nabla _{[k_{i},b_{i}]}\ln 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) {\displaystyle (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 {\displaystyle i} -th expert is changed to make its prediction closer to y {\displaystyle y} , but the amount of change is proportional to w (x) i N (y | μ i , I) {\displaystyle w(x)_{i}N(y|\mu _{i},I)} . This has a Bayesian interpretation. Given input x {\displaystyle x} , the prior probability that expert i {\displaystyle i} is the right one is w (x) i {\displaystyle w(x)_{i}} , and N (y | μ i , I) {\displaystyle N(y|\mu _{i},I)} is the likelihood of evidence y {\displaystyle y} . So, w (x) i N (y | μ i , I) ∑ j w (x) j N (y | μ j , I) {\displaystyle {\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 {\displaystyle i} , and so the rate of change for the i {\displaystyle 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 [8] [9] 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 {\displaystyle w_{i}} , and second order gating functions w j | i {\displaystyle w_{j|i}} and experts f j | i {\displaystyle f_{j|i}} . The total prediction is then ∑ i w i (x) ∑ j w j | i (x) f j | i (x) {\displaystyle \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. [9] [10]

The choice of gating function is often softmax. Other than that, gating may use gaussian distributions [11] and exponential families . [10]

Instead of performing a weighted sum of all the experts, in hard MoE, [12] only the highest ranked expert is chosen. That is, f (x) = f arg ■ max i w i (x) (x) {\displaystyle f(x)=f_{\arg \max _{i}w_{i}(x)}(x)} . This can accelerate training and inference time. [13]

The experts can use more general forms of multivariant gaussian distributions. For example, [8] proposed f i (y | x) = N (y | A i x + b i , Σ i) {\displaystyle f_{i}(y|x)=N(y|A_{i}x+b_{i},\Sigma _{i})} , where A i , b i , Σ i {\displaystyle A_{i},b_{i},\Sigma _{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 , [14] or Student's t-distribution . [15] For binary classification, it also proposed logistic regression experts, with f i (y | x) = { 1 1 + e β i T x + β i , 0 , y = 0 1 − 1 1 + e β i T x + β i , 0 , y = 1 {\displaystyle 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 β i , β i , 0 {\displaystyle \beta _{i},\beta _{i,0}} are learnable parameters. This is later generalized for multi-class classification, with multinomial logistic regression experts. [16]

One paper proposed mixture of softmaxes for autoregressive language modelling. [17] Specifically, consider a language model that given a previous text c {\displaystyle c} , predicts the next word x {\displaystyle x} . The network encodes the text into a vector v c {\displaystyle v_{c}} ,

and predicts the probability distribution of the next word as S o f t m a x (v c W) {\displaystyle \mathrm {Softmax} (v_{c}W)} for an embedding matrix W {\displaystyle W} . In mixture of softmaxes, the model outputs multiple vectors v c , 1 , … , v c , n {\displaystyle v_{c,1},\dots ,v_{c,n}} , and predict the next word as ∑ i = 1 n p i S o f t m a x (v c , i W i) {\displaystyle \sum _{i=1}^{n}p_{i}\;\mathrm {Softmax} (v_{c,i}W_{i})} , where p i {\displaystyle p_{i}} is a probability distribution by a linear-softmax operation on the activations of the hidden neurons within the model. The original paper demonstrated its effectiveness for recurrent neural networks . This was later found to work for Transformers as well. [18]

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. [19]

The earliest paper that applies MoE to deep learning dates back to 2013, [20] which proposed 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. Since the output from the gating is not sparse , all expert outputs are needed, and no conditional computation is performed.

The key goal when using 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 , [21] 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 , . . . , f n {\displaystyle f_{1},...,f_{n}} , and a gating network w {\displaystyle w} . The gating network is defined by w (x) = s o f t m a x (t o p k (W x + noise)) {\displaystyle w(x)=\mathrm {softmax} (\mathrm {top} _{k}(Wx+{\text{noise}}))} , where t o p k {\displaystyle \mathrm {top} _{k}} is a function that keeps the top-k entries of a vector the same, but sets all other entries to − ∞ {\displaystyle -\infty } . The addition of noise helps with load balancing.

The choice of k {\displaystyle k} is a hyperparameter that is chosen according to application. Typical values are k = 1 , 2 {\displaystyle k=1,2} . The k = 1 {\displaystyle k=1} version is also called the Switch Transformer. The original Switch Transformer was applied to a T5 language model . [22]

As demonstration, they trained a series of models for machine translation with alternating layers of MoE and LSTM , and compared with deep LSTM models. [23] Table 3 shows that the MoE models used less inference time compute, despite having 30x more parameters.

Load balancing

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 Switch Transformer [22] into a single auxiliary loss function. Specifically, let n {\displaystyle n} be the number of experts, then for a given batch of queries { x 1 , x 2 , . . . , x T } {\displaystyle \{x_{1},x_{2},...,x_{T}\}} , the auxiliary loss for the batch is n ∑ i = 1 n f i P i {\displaystyle n\sum _{i=1}^{n}f_{i}P_{i}} Here, f i = 1 T # (queries sent to expert i) {\displaystyle f_{i}={\frac {1}{T}}\#({\text{queries sent to expert }}i)} is the fraction of tokens that chose expert i {\displaystyle i} , and P i = 1 T ∑ j = 1 T w i (x j) ∑ i ′ ∈ experts w i ′ (x j) {\displaystyle P_{i}={\frac {1}{T}}\sum _{j=1}^{T}{\frac {w_{i}(x_{j})}{\sum _{i'\in {\text{experts}}}w_{i'}(x_{j})}}} is the fraction of weight on expert i {\displaystyle i} . This loss is minimized at 1 {\displaystyle 1} , precisely when

every expert has equal weight 1 / n {\displaystyle 1/n} in all situations.

Researchers at DeepSeek designed a variant of MoE, with "shared experts" that are always queried, and "routed experts" that might not be. They found that standard load balancing encourages the experts to be equally consulted, but this then causes experts to replicate the same core capacity, such as English grammar. They proposed the shared experts to learn core capacities that are often used, and let the routed experts to learn the peripheral capacities that are rarely used. [25]

They also proposed "auxiliary-loss-free load balancing strategy", which does not use auxiliary loss. Instead, each expert i {\displaystyle i} has an extra "expert bias" b i {\displaystyle b_{i}} . If an expert is being neglected, then their bias increases, and vice versa. During token assignment, each token picks the top-k experts, but with the bias added in. That is: [26] f (x) = ∑ i is in the top-k of { w (x) j + b j } j w (x) i f i (x) {\displaystyle f(x)=\sum _{i{\text{ is in the top-k of }}\{w(x)_{j}+b_{j}\}_{j}}w(x)_{i}f_{i}(x)} Note that the expert bias matters for picking the experts, but not in adding up the responses from the experts.

Capacity factor

Suppose there are n {\displaystyle n} experts in a layer. For a given batch of queries { x 1 , x 2 , . . . , x T } {\displaystyle \{x_{1},x_{2},...,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 {\displaystyle 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. The capacity factor is sometimes used to enforce a hard constraint on load balancing. Each expert is only allowed to process up to c ⋅ T / n {\displaystyle c\cdot T/n} queries in a batch. The ST-MoE report found c ∈ [1.25 , 2] {\displaystyle c\in [1.25,2]} to work well in practice. [27]

Routing

In the original sparsely-gated MoE, only the top-k experts are queried, and their outputs are weighted-summed. There are other methods. [27] Generally speaking, routing is an assignment problem : How to assign tokens to experts, such that a variety of constraints are followed (such as throughput, load balancing, etc.)? There are typically three classes of routing algorithm: the experts choose the tokens (" expert choice "), [28] the tokens choose the experts (the original sparsely-gated MoE), and a global assigner matching experts and tokens. [29]

During inference, the MoE works over a large batch of tokens at any time. If the tokens were to choose the experts, then some experts might few tokens, while a few experts get so many tokens that it exceeds their maximum batch size, so they would have to ignore some of the tokens. Similarly, if the experts were to choose the tokens, then some tokens might not be picked by any expert. This is the " token drop " problem. Dropping a token is not necessarily a serious problem, since in Transformers, due to residual connections , if a token is "dropped", it does not disappear. Instead, its vector representation simply passes through the feedforward layer without change. [29]

Other approaches include solving it as a constrained linear programming problem, [30] using reinforcement learning to train the routing algorithm (since picking an expert is a discrete action, like in RL). [31] The token-expert match may involve no learning (" static routing "): It can be done by a deterministic hash function [32] or a random number generator. [33]

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. [34]

A trained Transformer can be converted to a MoE by duplicating its feedforward layers, with randomly initialized gating, then trained further. This is a technique called "sparse upcycling". [35]

There are a large number of design choices involved in Transformer MoE that affect the training stability and final performance. The OLMoE report describes these in some detail. [36]

As of 2023 [update] , 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 [37] is a Transformer model with MoE layers. They demonstrated it by training a model with 15 billion parameters. MoE Transformer has also been applied for diffusion models . [38]

A series of large language models from Google used MoE. GShard [39] 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 [40] demonstrated a language model with 1.2 trillion parameters, each MoE layer using top-2 out of 64 experts. Switch Transformers [22] use top-1 in all MoE layers.

The NLLB-200 by Meta AI is a machine translation model for 200 languages. [41] 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. [42]

MoE large language models can be adapted for downstream tasks by instruction tuning . [43]

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. [44] [45]

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. [46] [47]

See also

Product of experts

Mixture models

Mixture of gaussians

Ensemble learning

References

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". IEEE Transactions on Neural Networks and Learning Systems . 23 (8): 1177– 1193. Bibcode : 2012ITNNL..23.1177Y . doi : 10.1109/TNNLS.2012.2200299 . ISSN 2162-237X . PMID 24807516 . S2CID 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 . S2CID 3185688 . Nguyen, Hien D.; Chamroukhi, Faicel (July 2018). "Practical and theoretical aspects of mixture-of-experts modeling: An overview" . WIREs Data Mining and Knowledge Discovery . 8 (4) e1246. doi : 10.1002/widm.1246 . ISSN 1942-4787 . S2CID 49301452 .

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". IEEE Transactions on Neural Networks and Learning Systems . 23 (8): 1177– 1193. Bibcode : 2012ITNNL..23.1177Y . doi : 10.1109/TNNLS.2012.2200299 . ISSN 2162-237X . PMID 24807516 .

S2CID 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 . S2CID 3185688 .

Nguyen, Hien D.; Chamroukhi, Faicel (July 2018). "Practical and theoretical aspects of mixture-of-experts modeling: An overview" . WIREs Data Mining and Knowledge Discovery . 8 (4) e1246. doi : 10.1002/widm.1246 . ISSN 1942-4787 . S2CID 49301452 .

Practical techniques for training MoE Transformer models 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 [cs.CL]. Muennighoff, Niklas; Soldaini, Luca; Groeneveld, Dirk; Lo, Kyle; Morrison, Jacob; Min, Sewon; Shi, Weijia; Walsh, Pete; Tafjord, Oyvind; Lambert, Nathan; Gu, Yuling; Arora, Shane; Bhagia, Akshita; Schwenk, Dustin; Wadden, David; Wettig, Alexander; Hui, Binyuan; Dettmers, Tim; Kiela, Douwe; Farhadi, Ali; Smith, Noah A.; Pang Wei Koh; Singh, Amanpreet; Hajishirzi, Hannaneh (2024). "OLMoE: Open Mixture-of-Experts Language Models". arXiv : 2409.02060 [cs.CL]. , with associated data release at "allenai/OLMoE" . Ai2. 2024-10-17 . Retrieved 2024-10-18 . Rajbhandari, Samyam; Li, Conglong; Yao, Zhewei; Zhang, Minjia; Reza Yazdani Aminabadi; Ammar Ahmad Awan; Rasley, Jeff; He, Yuxiong (2022). "DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale". arXiv : 2201.05596 [cs.LG]. DeepSeek-AI; et al. (2024). "DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model". arXiv : 2405.04434 [cs.CL]. DeepSeek-AI; et al. (2024). "DeepSeek-V3 Technical Report". arXiv : 2412.19437 [cs.CL]. Jin, Chao; Jiang, Ziheng; Bai, Zhihao; Zhong, Zheng; Liu, Juncai; Li, Xiang; Zheng, Ningxin; Wang, Xi; Xie, Cong; Huang, Qi; Heng, Wen; Ma, Yiyuan; Bao, Wenlei; Zheng, Size; Peng, Yanghua; Lin, Haibin; Liu, Xuanzhe; Jin, Xin; Liu, Xin (2025). "MegaScale-MoE: Large-Scale Communication-Efficient Training of Mixture-of-Experts Models in Production". arXiv : 2505.11432 [cs.LG].

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 [cs.CL].

Muennighoff, Niklas; Soldaini, Luca; Groeneveld, Dirk; Lo, Kyle; Morrison, Jacob; Min, Sewon; Shi, Weijia; Walsh, Pete; Tafjord, Oyvind; Lambert, Nathan; Gu, Yuling; Arora, Shane; Bhagia, Akshita; Schwenk, Dustin; Wadden, David; Wettig, Alexander; Hui, Binyuan; Dettmers, Tim; Kiela, Douwe; Farhadi, Ali; Smith, Noah A.; Pang Wei Koh; Singh, Amanpreet; Hajishirzi, Hannaneh (2024). "OLMoE: Open Mixture-of-Experts Language Models". arXiv : 2409.02060 [cs.CL]. , with associated data release at "allenai/OLMoE" . Ai2. 2024-10-17 . Retrieved 2024-10-18 .

Rajbhandari, Samyam; Li, Conglong; Yao, Zhewei; Zhang, Minjia; Reza Yazdani Aminabadi; Ammar Ahmad Awan; Rasley, Jeff; He, Yuxiong (2022). "DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale". arXiv : 2201.05596 [cs.LG].

DeepSeek-AI; et al. (2024). "DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model". arXiv : 2405.04434 [cs.CL].

DeepSeek-AI; et al. (2024). "DeepSeek-V3 Technical Report". arXiv : 2412.19437 [cs.CL].

Jin, Chao; Jiang, Ziheng; Bai, Zhihao; Zhong, Zheng; Liu, Juncai; Li, Xiang; Zheng, Ningxin; Wang, Xi; Xie, Cong; Huang, Qi; Heng, Wen; Ma, Yiyuan; Bao, Wenlei; Zheng, Size; Peng, Yanghua; Lin, Haibin; Liu, Xuanzhe; Jin, Xin; Liu, Xin (2025). "MegaScale-MoE: Large-Scale Communication-Efficient Training of Mixture-of-Experts Models in Production". arXiv : 2505.11432 [cs.LG].

Literature review for deep learning era Fedus, William; Dean, Jeff; Zoph, Barret (2022). "A Review of Sparse Expert Models in Deep Learning". arXiv : 2209.01667 [cs.LG]. Fuzhao, Xue (2024-07-21). "XueFuzhao/awesome-mixture-of-experts" . GitHub . Retrieved 2024-07-21 . Vats, Arpita (2024-09-02). "arpita8/Awesome-Mixture-of-Experts-Papers" . GitHub . Retrieved 2024-09-06 . Cai, Weilin; Jiang, Juyong; Wang, Fan; Tang, Jing; Kim, Sunghun; Huang, Jiayi

(2025). "A Survey on Mixture of Experts in Large Language Models". IEEE Transactions on Knowledge and Data Engineering . 37 (7): 3896. arXiv : 2407.06204 . Bibcode : 2025IDSO...37.3896C . doi : 10.1109/TKDE.2025.3554028 .

Fedus, William; Dean, Jeff; Zoph, Barret (2022). "A Review of Sparse Expert Models in Deep Learning". arXiv : 2209.01667 [cs.LG].

Fuzhao, Xue (2024-07-21). "XueFuzhao/awesome-mixture-of-experts" . GitHub . Retrieved 2024-07-21 .

Vats, Arpita (2024-09-02). "arpita8/Awesome-Mixture-of-Experts-Papers" . GitHub . Retrieved 2024-09-06 .

Cai, Weilin; Jiang, Juyong; Wang, Fan; Tang, Jing; Kim, Sunghun; Huang, Jiayi (2025). "A Survey on Mixture of Experts in Large Language Models". IEEE Transactions on Knowledge and Data Engineering . 37 (7): 3896. arXiv : 2407.06204 . Bibcode : 2025IDSO...37.3896C . doi : 10.1109/TKDE.2025.3554028 .