

CLAIMS

What is claimed and desired to be secured by Letters Patent of the United States is:

1. A method for reducing the computation cost of deep neural network (DNN) inferencing, comprising:
 - (a) determining a loss function based on a specific task, wherein the loss function is capable of measuring the predictive performance of a deep neural network;
 - (b) retrieving an initial deep neural network having a deep neural network computation graph and associated parameter values;
 - (c) iteratively sparsifying the number of active neurons of the deep neural network until the convergence of the sparsity ratio; and
 - (d) generating and outputting an activation-sparsified deep neural network.
2. The method of claim 1, wherein the sparsifying step comprises augmenting the loss function by replacing the original activation function with a new set of sparsity-inducing activation functions to form an objective function.
3. The method of claim 1, wherein the sparsifying step comprises augmenting the loss function with one or more activation regularizations to form an objective function.
4. The method of claim 3, wherein the sparsifying step comprises evaluating a gradient of the objective function by backpropagation with the augmented activation regularizations.
5. The method of claim 3, wherein the sparsifying step comprises evaluating a gradient of the objective function by backpropagation with sparsity-inducing activation functions.
6. The method of claim 4 or 5, wherein the sparsifying step comprises iteratively tuning one or more parameters associated with the deep neural network based on the gradient.
7. The method of claim 6, wherein the sparsifying step comprises determining whether the deep neural network converges by checking whether the change in the sparsity ratio becomes smaller than a predetermined threshold.

8. The method of claim 1, wherein the specific task comprises a training dataset.
9. The method of claim 1, wherein the number of active neurons of the deep neural network at the last layer is reduced to obtain a compressed representation of the vector produced by the deep neural network.
10. An activation compressor, comprising:
 - a network model configured to compute the activation value of a plurality of neurons and to compute a prediction, the network module including a collection of variables that parameterize a Deep Neural Network (DNN), the Deep Neural Network being modularized into a plurality of layers, each layer taking the output of one or more previous layers as input data and computing a new set of outputs;
 - a loss function module, communicatively coupled to the network model, configured to compute a first penalty value that measures the discrepancy between the prediction and a ground-truth answer;
 - an activation regularizer, communicatively coupled to the network model, configured to compute a second penalty value that measures the level of activation of all neurons of the network model by minimizing which neurons can sparsify the number of active neurons in the network model;
 - a data reader module, communicatively coupled to the network model, including a pipeline for obtaining input data for the Deep Neural Network from a plurality of data sources;
 - and
 - an optimizer, communicatively coupled to the loss function module and to the activation regularizer, configured to adjust the values of the collection of variables in the network model for decreasing an objective function in the loss function module and in the activation regularizer.
11. The activation compressor of claim 10, wherein the neurons comprise the one or more outputs of each layer.
12. The activation compressor of claim 10, wherein the prediction comprises the one or more outputs of the last layer.

13. The activation compressor of claim 10, wherein the loss function module computes the derivative of the penalty value with respect to the prediction.

14. The activation compressor of claim 10, wherein the loss function module comprises the following task-dependent loss function representation:

$$L(X^{(J)}) := \frac{1}{N} \sum_{i=1}^N \ell(X_i^{(J)}, Y_i)$$

where $X^{(J)} : N \times K$ are the outputs of the last layer of the deep neural network, which provides K scores of each sample that relate to the labels of target task $Y : N \times K$, the task-dependent loss function measuring discrepancy between outputs predicted from the DNN on a given training data set $\{X_i^{(J)}\}_{i=1}^N$ and on the given correct outputs on the training set $\{Y_i\}_{i=1}^N$.

15. The activation compressor of claim 10, wherein the activation regularizer computes the derivative that measures the level of activation of all neurons of the network by minimizing which neurons can sparsify the number of active neurons in the network.

16. The activation compressor of claim 10, wherein the activation regularizer directly regularizes the value of activation.

17. The activation compressor of claim 16, wherein the activation regularizer directly regularizes the value of activation, as represented by the following equation:

$$\sum_{l=1}^{J-1} \rho_l \|X^{(j)}\|_1$$

where the term $\{\rho_l\}_{l=1}^{J-1}$ denotes the tuning hyperparameters, and the term $\|X\|_1$ (summation of all absolute values of X) denotes a convex surrogate function of X that approximately measures how many non-zero components are in X .

18. The activation compressor of claim 10, wherein the activation regularizer regularizes a plurality of parameters that control the sparsity level of activation of each layer.

19. The activation compressor of claim 18, wherein the activation regularizer is used together with activation functions having parameters that control the sparsity level of activation, the activation function being represented as follows:

$$[\alpha_r(X)]_k := \begin{cases} \text{vec}(X)_k, & k \in \text{top}_r(\text{vec}(X)) \\ 0, & \text{o. w.} \end{cases}$$

which preserves only values of the top- r elements with largest values and suppresses the remaining to be zeros.

20. The activation compressor of claim 18, wherein the data reader module comprises a pipeline for obtaining input data for the deep neural network from a plurality of data sources, the pipeline including at least one cropping, subsampling, upsampling, batching, and whitening and normalization.

21. The activation compressor of claim 18, wherein the optimizer module is configured to adjust the values of variables in the network model for decreasing an objective function that includes the loss function and the activation regularizer.

22. The activation compressor of claim 18, wherein the input data comprises at least one of image, audio, video and text.

23. A method for efficient computation of a deep neural network (DNN), comprising:
 providing a network model for computing the activation value of a plurality of neurons and to compute a prediction, the network model module including a collection of variables that parameterize a deep neural network, the deep neural network being modularized into a plurality of layers, each layer taking the output of one or more previous layers as input data and computes a new set of outputs;

(b) evaluating gradient of an objective function by backpropagation through a network model and a data reader;

(c) updating DNN parameters based on the gradient by an optimizer for adjusting the values of the collection of variables in the network model for decreasing the objective function;

(d) determining by an optimizer whether the change in objective or the magnitude of gradient is sufficiently small based on a predetermined threshold, to indicate that the objective has converged,

when the objective has not converged, iteratively repeating steps (b) and (c), and

when the objective has converged, generating an output with the sparsified deep neural network.

24. The method of claim 23, wherein the evaluating gradient step comprises requesting a gradient of an activation regularizer with respect to the parameters W (i.e. $\nabla_W R$) from a regularizer gradient.

25. The method of claim 24, wherein the evaluating gradient step comprises requesting the network model module to compute the activation values of all layers $\{X^{(l)}\}_{l=1...L}$ (462), with input data provided by an preprocessing pipeline.

26. The method of claim 23, wherein the updating DNN parameters step comprises requesting gradient of the loss function with respect to the parameters W (i.e. $\nabla_W L$) from a loss gradient by a loss function module.

27. The method of claim 26, wherein the update DNN parameters step comprises requesting the network model to compute the values of the prediction (output) layers $X^{(l)}$, with input data provided from an input preprocessing pipeline to the network model and to correct labels to the loss function module.

28. The method of claim 23, wherein the input data comprises at least one of image, audio, video and text.