

# Multimodal Tucker Decomposition for Gated RBM Inference

Mauricio Maldonado-Chan <sup>1</sup>, Andres Mendez-Vazquez <sup>1,\*</sup> and Ramon Osvaldo Guardado-Medina <sup>2</sup>

<sup>1</sup> Department of Computer Science, Cinvestav, Unidad Guadalajara, Zapopan 45019, Jalisco, Mexico; maldonado80@gmail.com

<sup>2</sup> Department of Research, Escuela Militar de Mantenimiento y Abastecimiento, Universidad del Ejercito y Fuerza Aerea, Guadalajara 45138, Jalisco, Mexico; ramon.o.guardado-medina@ieee.org

\* Correspondence: andres.mendez@cinvestav.mx

**Abstract:** Gated networks are networks that contain gating connections in which the output of at least two neurons are multiplied. The basic idea of a gated restricted Boltzmann machine (RBM) model is to use the binary hidden units to learn the conditional distribution of one image (the output) given another image (the input). This allows the hidden units of a gated RBM to model the transformations between two successive images. Inference in the model consists in extracting the transformations given a pair of images. However, a fully connected multiplicative network creates cubically many parameters, forming a three-dimensional interaction tensor that requires a lot of memory and computations for inference and training. In this paper, we parameterize the bilinear interactions in the gated RBM through a multimodal tensor-based Tucker decomposition. Tucker decomposition decomposes a tensor into a set of matrices and one (usually smaller) core tensor. The parameterization through Tucker decomposition helps reduce the number of model parameters, reduces the computational costs of the learning process and effectively strengthens the structured feature learning. When trained on affine transformations of still images, we show how a completely unsupervised network learns explicit encodings of image transformations.

**Keywords:** unsupervised learning; gated restricted Boltzmann machine; tucker decomposition; tensors



**Citation:** Maldonado-Chan, M.; Mendez-Vazquez, A.; Guardado-Medina, R.O. Multimodal Tucker Decomposition for Gated RBM Inference. *Appl. Sci.* **2021**, *11*, 7397. <https://doi.org/10.3390/app11167397>

Academic Editor: Marcos Gestal and Alexandre Carvalho

Received: 25 May 2021

Accepted: 3 August 2021

Published: 11 August 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

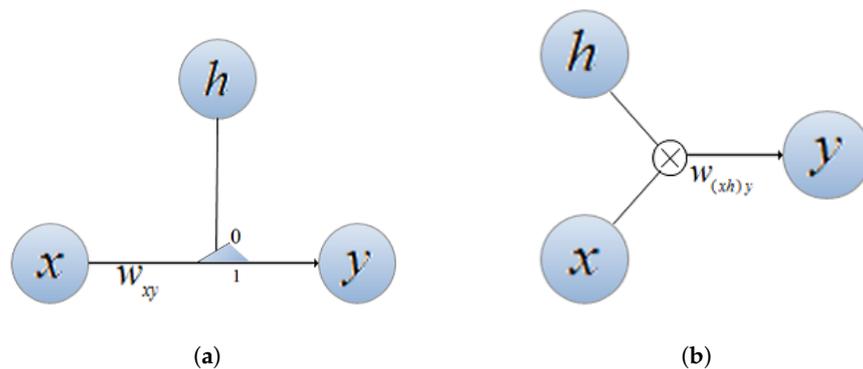
## 1. Introduction

Feature engineering is the process of transforming raw data into a suitable representation or feature vector that can be used to train a machine learning model for a prediction problem. For decades, the traditional approach to feature engineering was to manually build a feature extractor that required careful engineering and considerable domain expertise. Manual feature engineering required writing code that was problem-specific and that had to be adjusted for each new dataset. Deep learning allows computational models that are composed of multiple processing layers to learn representation of data with multiple levels of abstraction [1] (p. 436). Deep learning improves the process of feature engineering by automatically extracting useful and interpretable features. It also eliminates the need for domain expertise and hard core feature extraction by learning high-level features from the data in a hierarchical manner. The main building blocks in the deep learning literature are restricted Boltzmann machines [2,3], autoencoders [4,5], convolutional neural networks [6,7], and recurrent neural networks [8].

Most of these architectures are used to learn a relationship between a single input source and the corresponding output. However, there are many domains where the representation to be learned is the correspondence between more than one source and one output [9]. For instance, many tasks in vision carry the relevant information in the encoding of the relationship between observations, not the content of a single observation.

The above deep learning building blocks can be extended to contain gated connections and allow them to learn relationships between at least two sources of input and at least

one output. A defining feature of the gated networks is that they contain gating connections. Unlike other networks, whose layer-to-layer connections are linear, gated networks introduce higher order interactions. The connection between two neurons  $x$  and  $y$  is in fact modulated by the activity of a third neuron  $h$ . Figure 1 illustrates two different approaches for the connection relationship between three neurons: to control the flow of information in the network or to model multiplicative interactions between several inputs. In the first type of connection, the neuron  $h$  is used as a switch or a gate that stops or does not stop the flow of information between  $x$  and  $y$ . In the second type of connection, the connection implements a multiplicative relationship between  $x$  and  $h$ , whose values are multiplied before being projected to the output  $y$  by the synaptic connection. In the multiplicative interaction, we can say that the neuron  $h$  modulates the signal between  $x$  and  $y$ .



**Figure 1.** Two types of gating connections. (a): gated connection used to control the flow of information in a network; (b): gated connection implementing a multiplicative relationship between two inputs  $x$  and  $h$  to provide the output  $y$ .

Despite the growing interest, the literature about gated networks is still sparse [9] (p. 2). The focus of this paper is the specific family of neural networks implementing a multiplicative gating relationship that are built on an RBM architecture. This concept of a gated restricted Boltzmann machine was first introduced in [10]. The basic idea of the gated model is to use the binary hidden units to learn the conditional distribution of one image (the input) given another image (the output). In [11], the authors revisit the problem and present a factorization alternative to the gated RBM. A gated RBM can be also considered a higher-order Boltzmann machine. As cited in [10] (p. 1474), Boltzmann machines that contain multiplicative interactions between more than two units are known in general as higher-order Boltzmann machines [12].

The remainder of this paper is organized as follows: in the following section, the RBM model and its popular training method Contrastive Divergence (CD) are presented. In Section 3, the standard gated RBM model is presented and a mechanism to reduce the number of its weights by projecting onto factor layers is discussed. Next, we present in Section 4 a small overview on the subject of tensors and the factorization known as Tucker decomposition. In Section 5, we introduce a multimodal tensor-based Tucker decomposition for the three-way parameter tensor in the gated RBM. In this section, we also show that by using Tucker Decomposition, we can use less than the cubically many parameters implied by the three-way weight tensor and introduce a Contrastive Divergence-based training procedure for the gated RBM, which reduces the number of model parameters and efficiently parameterizes its bilinear interactions. Experimental results and their corresponding discussion are reviewed in Section 6. Finally, conclusions are presented in Section 7.

## 2. Restricted Boltzmann Machines

A restricted Boltzmann machine is a type of graphical model in which the nodes  $x = \{\mathbf{v}, \mathbf{h}\}$  form a symmetrical bipartite graph with binary observed variables  $\mathbf{v} \in \{0, 1\}^n$

(visible nodes) and binary latent variables  $\mathbf{h} \in \{0, 1\}^m$  (hidden nodes). Each visible unit (node) is connected to each hidden unit, but there are no visible-to-visible or hidden-to-hidden connections. Importantly, RBMs are able to model the probabilistic density of the joint distribution of visible and hidden units, enabling them to generate samples similar to those of the training data onto the visible layer. This type of model is called *generative* models. A classic RBM model is illustrated in Figure 2.

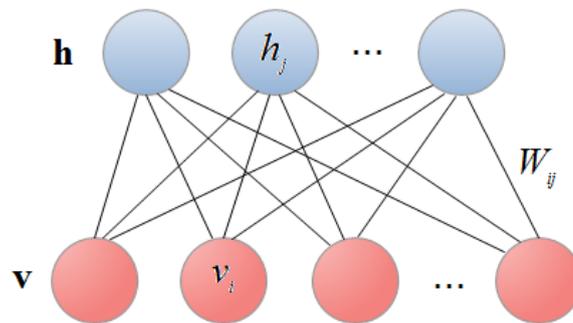


Figure 2. Restricted Boltzmann machine.

An RBM is governed by an energy function, the energy of a joint configuration  $(\mathbf{v}, \mathbf{h})$  between the visible layer, and the hidden layer is given by

$$E(\mathbf{v}, \mathbf{h}) = - \sum_{i \in \text{visible}} b_i v_i - \sum_{j \in \text{hidden}} c_j h_j - \sum_{ij} W_{ij} v_i h_j, \tag{1}$$

where  $\theta = \{W, \mathbf{b}, \mathbf{c}\}$  are the parameters of the model.  $W_{ij}$  is the connection weight matrix between the visible layer and the hidden layer, and  $b_i$  and  $c_j$  are biases of the visible layer and the hidden layer, respectively.

$E(\mathbf{v}, \mathbf{h})$  is called the energy of the state  $(\mathbf{v}, \mathbf{h})$ . The joint probability distribution under the model is given by the Boltzmann distribution:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\frac{-1}{\tau} E(\mathbf{v}, \mathbf{h})\right)$$

where  $\mathcal{Z}(\theta)$  is a normalizing constant and  $\tau$  is the thermodynamic temperature (often considered as 1).  $\mathcal{Z}(\theta)$  is called a *partition function* and is defined by summing over all possible visible and hidden configurations. Therefore, it is extremely hard to compute when the number of units is large. The partition function is represented as follows:

$$\mathcal{Z}(\theta) = \sum_{\mathbf{v}, \mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) \text{ for } \tau = 1.$$

Since there are no connections between two variables of the same layer, it is possible to derive an expression for  $p(v_k = 1 | \mathbf{h})$ ; this is the probability of a particular visible unit being *on* given a hidden configuration:

$$p(h_j = 1 | \mathbf{v}) = \sigma\left(\sum_i W_{ij} v_i + c_j\right) \tag{2}$$

This is also true for a particular hidden unit given a visible configuration:

$$p(v_i = 1 | \mathbf{h}) = \sigma\left(\sum_j W_{ij} h_j + b_i\right) \tag{3}$$

where  $\sigma(z) = 1/(1 + \exp(-z))$ . This leads to a block Gibbs sampling dynamics, used universally for sampling from RBMs.

*RBM Training*

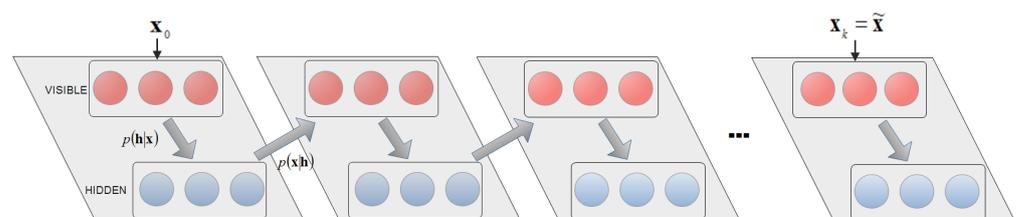
Carreira-Perpinan and Hinton [13] showed that the derivative of the log-likelihood of the data under the RBM with respect to its parameters is:

$$\frac{\partial \log p(\mathbf{v}, \mathbf{h})}{\partial \theta} = - \left\langle \frac{\partial \log E(\mathbf{v}, \mathbf{h})}{\partial \theta} \right\rangle_{data} + \left\langle \frac{\partial \log E(\mathbf{v}, \mathbf{h})}{\partial \theta} \right\rangle_{model} \tag{4}$$

where  $\langle \cdot \rangle_{data}$  denotes the expectation over the data, or in other words the distribution  $p(\mathbf{h} | \mathbf{v}^{(t)}, \theta)$ . In the same way,  $\langle \cdot \rangle_{model}$  denotes the expectation over the model distribution  $p(\mathbf{v}, \mathbf{h} | \theta)$ . However, directly calculating the sums that run over all values of  $\mathbf{v}$  and  $\mathbf{h}$  in the second term in (4) leads to a computational complexity, which is in general exponential in the number of variables.

The  $\langle \cdot \rangle_{model}$  expectation can be approximated with samples from the model distribution. These samples can be obtained via Gibbs sampling, iteratively sampling all units in one layer at once given the other layer using (2) and (3) alternately. However, this requires running the Markov chain for an infinite time to ensure convergence to a stationary state, making it an unfeasible solution.

Obtaining an unbiased sample of  $\langle \cdot \rangle_{model}$  is extremely difficult. Hinton [3] approximates the second term in the true gradient in (4) by using an approximation of the derivative called Contrastive Divergence: the goal of CD is to replace the average  $\langle \cdot \rangle_{model}$  with samples  $\langle \cdot \rangle_k$  obtained after running  $k$  steps of Gibbs sampling starting from each data sample. This is illustrated in Figure 3. A typical value used in the literature is  $k = 1$ . Moreover, this way of updating the parameters has become a standard way of training RBMs. Although it has proven to work well in practice, CD does not yield the best approximation of the log-likelihood gradient [13,14]. There has been much research dedicated to better understanding this approach and the reasoning behind its success [13–15], leading to many variations being proposed from the perspective of improving the Markov chain Monte Carlo approximation to the gradient, namely, Persistent CD [16], Fast Persistent CD [17], and Parallel Tempering [18].



**Figure 3.** The CD- $k$  learning procedure. To estimate  $\langle v_i h_j \rangle_{model}$ , we initialize the visible units of the data and alternately sample the hidden and then the visible units. The observed value of  $v_i h_j$  at the  $(N + 1)$ st sample is used as the estimate.

**3. Gated RBM**

Gated RBMs are a natural extension of RBMs, in which the gating idea is applied via the units of a hidden layer connecting/gating the neurons of two other layers (input and output). As in the RBM model, the gated RBM is governed by an energy function. Memisevic and Hinton [10] propose using the following three-way energy function that captures all possible correlations among the components of the  $\mathbf{x}$  (input),  $\mathbf{y}$  (output), and  $\mathbf{h}$  (hidden) layers:

$$- E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = \sum_{ijk} W_{ijk} x_i y_j h_k \tag{5}$$

where  $i, j$  and  $k$  index the units in the input, output and hidden layers, respectively;  $x_i$  is the binary state of input pixel  $i$ ;  $y_j$  is the binary state of output pixel  $j$ ; and  $h_k$  is the binary state of hidden unit  $k$ .

Figure 4 shows a fully connected gated RBM. The components  $W_{ijk}$  of its three-way interaction tensor connect units  $x_i, y_j$  and  $h_k$  and learn to weight the importance of the possible correlations given some training data. This type of multiplicative interaction among the input, output, and hidden units leads to a type of higher-order Boltzmann machine that retains the computational benefits of RBMs, such as being amenable to contrastive divergence training and allowing for efficient inference schemes that use alternating Gibbs sampling [11] (p. 1474).

In practice, to be able to model affine and not just linear dependencies, it is useful to add biases to the output and hidden units, which makes (5):

$$-E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = \sum_{ijk} W_{ijk} x_i y_j h_k + \sum_k W_k^h h_k + \sum_j W_j^y y_j \tag{6}$$

where the terms  $\sum_k W_k^h h_k$  and  $\sum_j W_j^y y_j$  are bias terms used to model the base rates of activity of the hidden and output units, respectively. In general, a higher-order Boltzmann machine can also contain bias terms for the input units, but following [11], we do not use these.

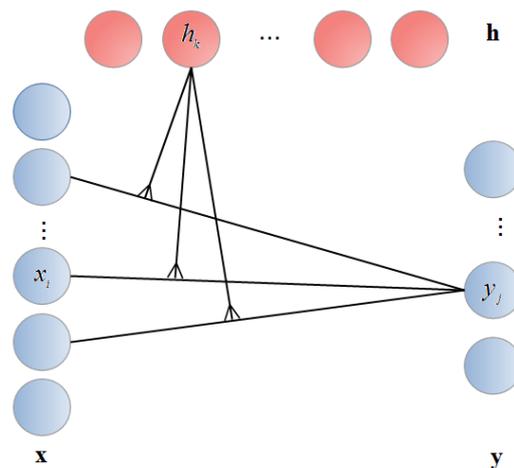


Figure 4. A fully connected multiplicative gated RBM.

The negative energy  $-E(\mathbf{y}, \mathbf{h}; \mathbf{x})$  captures the compatibility between the input, output and hidden units. As in the RBM model, we can use this energy function to define the joint distribution  $p(\mathbf{y}, \mathbf{h} | \mathbf{x})$  over output and hidden variables by exponentiating and normalizing:

$$p(\mathbf{y}, \mathbf{h} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{y}, \mathbf{h}; \mathbf{x})) \tag{7}$$

where

$$Z(\mathbf{x}) = \sum_{\mathbf{h}, \mathbf{y}} \exp(-E(\mathbf{y}, \mathbf{h}; \mathbf{x})) \tag{8}$$

is a normalizing constant, which depends on the input image  $\mathbf{x}$ . To obtain the distribution over output images, given the input, we marginalize and get:

$$p(\mathbf{y} | \mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{y}, \mathbf{h} | \mathbf{x}) \tag{9}$$

This marginalization over the hidden units is known in the literature as *free energy*. Note that  $p(\mathbf{y}, \mathbf{h} | \mathbf{x})$  or  $Z(\mathbf{x})$  cannot be computed exactly, since both contain sums over the exponentially large number of all possible instances of the hidden units and output units

for  $Z(\mathbf{x})$ . However, we do not actually need to compute any of these quantities to perform either inference or learning, as we shall see in the next section.

It is important to note that the normalization step in (8) is performed over  $\mathbf{h}$  and  $\mathbf{y}$ ; thus, it defines the conditional distribution  $p(\mathbf{y}, \mathbf{h} | \mathbf{x})$  rather than the joint  $p(\mathbf{y}, \mathbf{h}, \mathbf{x})$ . This is done deliberately to free the model from many of the independence assumptions that a fully generative model would need to make, hence simplifying inference and learning [10].

Inference then consists of guessing the transformation, or equivalently its encoding  $\mathbf{h}$ , from a given pair of observed images  $\mathbf{x}$  and  $\mathbf{y}$ . Since the energy function does not contain interactions between any pairs of output units or pairs of hidden units, it is possible to derive a closed-form expression for  $p(h_k = 1 | \mathbf{x}; \mathbf{y})$ ; this is the probability of a particular hidden unit when an input–output image pair is given:

$$p(h_k = 1 | \mathbf{x}; \mathbf{y}) = \sigma\left(\sum_i \sum_j W_{ijk} x_i y_j + w_k^h\right) \quad (10)$$

This is also true for the output units when input and hidden units are given:

$$p(y_j = 1 | \mathbf{h}; \mathbf{x}) = \sigma\left(\sum_i \sum_k W_{ijk} x_i h_k + w_j^y\right) \quad (11)$$

where  $\sigma(z) = 1/(1 + \exp(-z))$ .

Note that in practice, function  $\sigma(\cdot)$  can be changed for some other non-linear activation function. Regardless of the activation function, these models are called *bilinear* because, if one input is held fixed, the output is linear in the other input.

Consider now the task of predicting the hidden layer  $\tilde{\mathbf{h}}$  given the input  $\mathbf{x}$  and output  $\mathbf{y}$  units, in such multiplicative network, this consists in computing all the values  $\tilde{h}_k$  of  $\tilde{\mathbf{h}}$  using (10):

$$\forall k, \tilde{h}_k = \sigma_h\left(\sum_i \sum_j W_{ijk} x_i y_j + w_k^h\right) \quad (12)$$

Alternatively, one may compute  $\tilde{\mathbf{y}}$  given the input  $\mathbf{x}$  and hidden  $\mathbf{h}$  units using (11):

$$\forall j, \tilde{y}_j = \sigma_y\left(\sum_i \sum_k W_{ijk} x_i h_k + w_j^y\right) \quad (13)$$

Memisevic and Hinton [10] point out that this type of three-way model can be interpreted as a mixture of experts. Note from (5) that in the way the energy is defined, the importance that each hidden unit  $h_k$  attributes to the correlatedness (or anti-correlatedness) of a particular pair  $x_i, y_j$  is determined by  $W_{ijk}$ .

To train the probabilistic model we can use the same principle from Contrastive Divergence in the RBM model: we maximize the average conditional log-likelihood  $L = \frac{1}{N} \sum_{\alpha} \log p(\mathbf{y}^{\alpha} | \mathbf{x}^{\alpha})$  for a set of training pairs  $\{(\mathbf{x}^{\alpha}, \mathbf{y}^{\alpha})\}$ . The derivative of the (negative) log probability with respect to the weight parameter  $W_{ijk}$  is given by the difference of two expectations:

$$-\frac{\partial L}{\partial W_{ijk}} = \left\langle \frac{\partial E(\mathbf{y}^{\alpha}, \mathbf{h}; \mathbf{x}^{\alpha})}{\partial W_{ijk}} \right\rangle_{\mathbf{h}} - \left\langle \frac{\partial E(\mathbf{y}^{\alpha}, \mathbf{h}; \mathbf{x}^{\alpha})}{\partial W_{ijk}} \right\rangle_{\mathbf{h}, \mathbf{y}} \quad (14)$$

where  $\langle \cdot \rangle_z$  denotes the expectation with regard to variable  $z$ . Note that the expectation in the first term in (14) is over the posterior distribution over hidden units, and it can be computed efficiently using (12). The expectation in the second term in (14) is over all possible output/hidden instantiations and is intractable. However, because of the conditional independences of  $\mathbf{h}$  given  $\mathbf{x}$  and  $\mathbf{y}$  and  $\mathbf{y}$  given  $\mathbf{x}$  and  $\mathbf{h}$ , we can easily sample from the conditional distributions  $p(\mathbf{h} | \mathbf{x}, \mathbf{y})$  and  $p(\mathbf{y} | \mathbf{x}, \mathbf{h})$ . Using Gibbs sampling with Equations (12) and (13), respectively, for the hidden and output layer, we can approximate the intractable term.

### Factorized Gated RBM

Memisevic and Hinton [11] propose a way of reducing the number of weights that consists of projecting the  $\mathbf{x}$ ,  $\mathbf{y}$ , and  $\mathbf{h}$  layers onto smaller layers, noted, respectively, as  $\mathbf{f}^x$ ,  $\mathbf{f}^y$ , and  $\mathbf{f}^h$  before performing the product between these smaller layers. Given their multiplicative role, these layers are called *factor* layers.

The three-way tensor  $W_{ijk}$  is constrained to use these projections; three factor layers  $\mathbf{f}^x$ ,  $\mathbf{f}^y$ , and  $\mathbf{f}^h$  of the same size  $n_f$  as is illustrated in Figure 5. Moreover, the weights  $W_{ijk}$  are restricted to follow a specific form:

$$W_{ijk} = \sum_{f=1}^F W_{if}^x W_{jf}^y W_{kf}^h. \tag{15}$$

With this constraint, the matrices  $W^x$ ,  $W^y$  and  $W^h$  are of respective size  $n_x \times n_f$ ,  $n_y \times n_f$  and  $n_h \times n_f$ ; thus the total number of weights is just  $n_f \times (n_x + n_y + n_h)$ , which is quadratic instead of cubic in the size of input or factors.

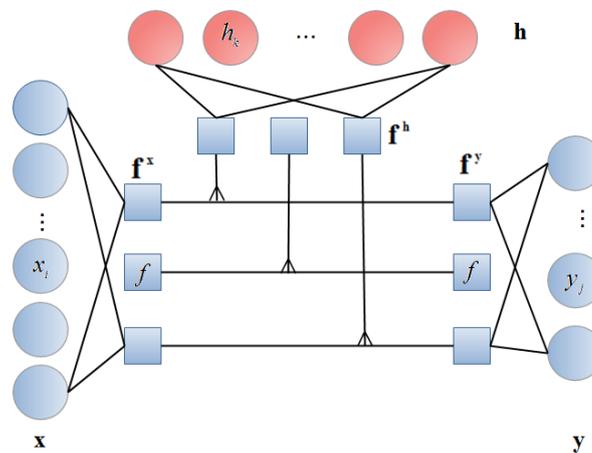


Figure 5. A simplified gated RBM introducing factor layers.

### 4. Tucker Decomposition

Since tensors, specifically a three-way tensor, and its corresponding Tucker decomposition are at the core of this study, a brief overview of the subject of tensors is presented. First introduced by Tucker [19] and refined in subsequent articles by Levin [20] and Tucker et al. [21], the Tucker decomposition is a form of higher-order Principal Component Analysis. Tucker decomposition factorizes a tensor into a (usually smaller) core tensor and a set of factor matrices. One factor matrix along each mode. Then, in the three-way case where  $\mathcal{W}_{ijk} \in \mathbb{R}^{I \times J \times K}$ , we have

$$\mathcal{W}_{ijk} \approx \mathcal{G}_{pqr} \times_I \mathbf{A}_{ip} \times_J \mathbf{B}_{jq} \times_K \mathbf{C}_{kr} \tag{16}$$

where the operator  $\times_n$  denotes the mode- $n$  multiplication of a tensor by a matrix in mode  $n$ .  $\mathbf{A} \in \mathbb{R}^{I \times P}$ ,  $\mathbf{B} \in \mathbb{R}^{J \times Q}$ , and  $\mathbf{C} \in \mathbb{R}^{K \times R}$  are known as the factor matrices and can be thought of as the principal components in each mode. The tensor  $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$  is called the core tensor, and its entries show the level of interaction between the different components. The Tucker decomposition of  $\mathcal{W}_{ijk}$  is usually summarized as:

$$\mathcal{W} = \llbracket \mathcal{G}; \mathbf{A}; \mathbf{B}; \mathbf{C} \rrbracket \tag{17}$$

A comprehensive discussion on Tucker decomposition and tensor analysis is available in Kolda [22]. If  $\mathcal{G}$  is the same size as  $\mathcal{W}$ , the Tucker decomposition is simply a change of

basis. More often, we are interested in using a change of basis to compress  $\mathcal{W}$ . If  $P, Q, R$  are smaller than  $I, J, K$ , the core tensor  $\mathcal{G}$  can be thought of as compressed version of  $\mathcal{W}$ .

For some computations presented in this document, it is important to be able to transform the indices of a tensor so that it can be represented as a matrix and vice versa. *Matricization*, also known as *unfolding* or *flattening*, is the process of reordering the elements of a tensor ( $N$ -way array) into a matrix [23]. For instance, a  $3 \times 4 \times 5$  tensor can be rearranged as a  $12 \times 5$  matrix or a  $3 \times 20$  matrix, and so on.

The matricized forms (one per mode) of (16) are:

$$\mathbf{W}_i \approx \mathbf{A}\mathbf{G}_p(\mathbf{C} \otimes \mathbf{B})^T \tag{18}$$

$$\mathbf{W}_j \approx \mathbf{B}\mathbf{G}_q(\mathbf{C} \otimes \mathbf{A})^T \tag{19}$$

$$\mathbf{W}_k \approx \mathbf{C}\mathbf{G}_r(\mathbf{B} \otimes \mathbf{A})^T \tag{20}$$

### 5. Materials and Methods

In this section, we propose a strategy for reducing the number of parameters in a gated RBM. First, we refactor the gated RBM model by applying a multimodal tensor-based Tucker decomposition to its three-way weight tensor. Then, we show that by using Tucker Decomposition, we can use fewer than the cubically many parameters implied in the model. Finally, we introduce a Contrastive Divergence-based training procedure for the tucker decomposed gated RBM, which efficiently parameterizes its bilinear interactions.

#### 5.1. Decomposing the Three-Way Tensor in a Gated RBM

The central idea of this research is to represent the required three-way interaction tensor in the gated RBM model using far fewer parameters through its Tucker Decomposition. The energy function in a gated RBM (Equation (5)) captures all possible correlations among the components of the  $\mathbf{x}$  (input),  $\mathbf{y}$  (output), and  $\mathbf{h}$  (hidden) layers. In this function, parameter  $\mathcal{W}$  defines a three-way interaction tensor that learns the importance of correlations between layers  $\mathbf{x}$  and  $\mathbf{y}$ . However, despite its appealing modeling power, a fully parametrized gated RBM suffers from an explosion in the number of parameters, quickly becoming intractable because the size of the full tensor  $\mathcal{W}$  is prohibitive using common dimensions for textual, visual, or output spaces.

As we will see, it is possible to use much fewer parameters by factorizing the multi-way interaction tensor via Tucker decomposition. We can plug the Tucker decomposition Equation (16) into the energy function of the gated RBM Equation (5). Then, the energy of a joint configuration of the visible (input/output) and hidden units is defined as:

$$-E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = \underbrace{[\mathcal{G}_{pqr} \times_P \mathbf{A}_{ip} \times_Q \mathbf{B}_{jq} \times_R \mathbf{C}_{kr}]}_{\mathcal{W}_{ijk}} \mathbf{x}_i \mathbf{y}_j \mathbf{h}_k$$

Using the distributive law, this can be rewritten as:

$$-E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = \mathcal{G}_{pqr} \times_P \mathbf{x}_i^T \mathbf{A}_{ip} \times_Q \mathbf{y}_j^T \mathbf{B}_{jq} \times_R \mathbf{h}_k^T \mathbf{C}_{kr}$$

We can drop subindices for clarity and get:

$$-E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = \mathcal{G} \times_P (\mathbf{x}^T \mathbf{A}) \times_Q (\mathbf{y}^T \mathbf{B}) \times_R (\mathbf{h}^T \mathbf{C}) \tag{21}$$

It is possible to simplify the notation in (21) if we define:

$$\hat{\mathbf{x}} = \mathbf{x}^T \mathbf{A},$$

$$\hat{\mathbf{y}} = \mathbf{y}^T \mathbf{B}, \text{ and}$$

$$\hat{\mathbf{h}} = \mathbf{h}^T \mathbf{C}.$$

Then the energy function in Equation (21) is given by:

$$-E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = \mathcal{G} \times_P \hat{\mathbf{x}} \times_Q \hat{\mathbf{y}} \times_R \hat{\mathbf{h}} \tag{22}$$

5.2. Interpretation of the Refactored Model: Dimensionality Reduction

Let us consider the three-way tensor with shape  $(n_i, n_j, n_k)$  and its corresponding Tucker decomposition presented in Figure 6. As we parametrize the weights of the three-way tensor  $\mathcal{W}$  with its Tucker decomposition, we are now able to separate  $\mathcal{W}$  into four components, each having a specific role in the gated RBM model. Matrices  $\mathbf{A}$  and  $\mathbf{B}$  project the input ( $\mathbf{x}$ ) and output ( $\mathbf{y}$ ) images into spaces of respective dimension  $n_p$  and  $n_q$ . The core tensor  $\mathcal{G}$ , whose shape is  $(n_p, n_q, n_r)$ , is used to model the interactions between the input and output image projections. Finally, the matrix  $\mathbf{C}$  projects the scores of the pair embedding  $\mathbf{h}$  into a space of dimension  $n_r$ .

Moreover, if  $\mathcal{G}$  has the same shape as  $\mathcal{W}$ , the Tucker decomposition is simply a change of basis. However, in our case, we are interested in using a change of basis to compress  $\mathcal{W}$ . If  $n_p, n_q, n_r$  are smaller than  $n_i, n_j, n_k$ , the core tensor  $\mathcal{G}$  can be thought of as compressed version of  $\mathcal{W}$ . Note that the dimensions for the factor matrices  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{C}$  are a result of the  $n$ -mode product between the original tensor  $\mathcal{W}$  and the core tensor  $\mathcal{G}$ . The factor matrix  $\mathbf{A}$  has dimensions  $n_i \times n_p$  as a result of the  $i$ -mode product between  $\mathcal{W}$  and  $\mathcal{G}$ . Respectively, factor matrix  $\mathbf{B}$  has dimensions  $n_j \times n_q$  (from  $j$ -mode product) and factor matrix  $\mathbf{C}$  has dimensions  $n_k \times n_r$  (from  $k$ -mode product). By constraining  $n_p, n_q, n_r$  to be smaller than  $n_i, n_j, n_k$ , we use a lower number of components for each of the three modes while at the same time linking these components to each other by means of the three-way core tensor.

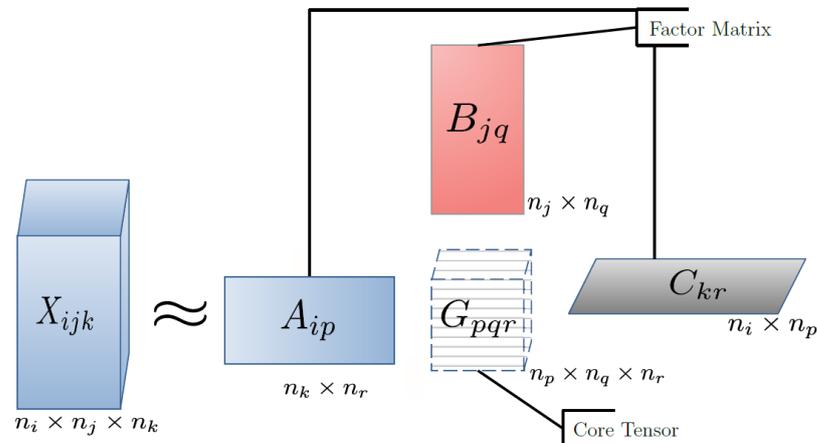


Figure 6. Dimensionality reduction in the factorized tensor.

Again, consider the three-way tensor in Figure 6, whose respective cardinality of each layer is given by  $n_i, n_j$  and  $n_k$ . If we consider  $n_i \approx n_j \approx 2048$  and  $n_k \approx 2000$ , then the number of free parameters in the tensor  $\mathcal{W}$  is  $\sim 8.39 \times 10^9$ . It is easy to see that having such a number of free parameters is a problem both for memory and computing costs. In contrast, if we apply Tucker decomposition to this three-way tensor using a core tensor with shape  $1024 \times 1024 \times 1000$ , then the number of free parameters would be  $\sim 1.05 \times 10^9$ , which is given by the sum of parameters from the core tensor and the three factor matrices. By applying Tucker decomposition we reduce the dimensionality of the model. Note that the compression in the data is determined by the ranks of the core tensor.

Dimensionality reduction has long been an important technique for data representation. It reduces the space complexity of the underlying model so that it has higher stability when fitting, require fewer parameters and consequently becomes easier to interpret.

### 5.3. Training the Refactored Gated RBM

To train the refactored gated RBM, we can maximize its average conditional log-likelihood

$$L = \sum_{\alpha} \log p(\mathbf{y}^{\alpha} | \mathbf{x}^{\alpha})$$

for a set of training pairs  $\{(\mathbf{x}^{\alpha}, \mathbf{y}^{\alpha})\}$ . By substituting Equations (7) and (9), the derivative of the negative log probability with regard to any element  $\theta$  of the parameter tensor is given by

$$-\frac{\partial L}{\partial \theta} = \left\langle \frac{\partial E(\mathbf{y}^{\alpha}, \mathbf{h}; \mathbf{x}^{\alpha})}{\partial \theta} \right\rangle_{\mathbf{h}} - \left\langle \frac{\partial E(\mathbf{y}^{\alpha}, \mathbf{h}; \mathbf{x}^{\alpha})}{\partial \theta} \right\rangle_{\mathbf{h}, \mathbf{y}} \tag{23}$$

where  $\langle \cdot \rangle_{\mathbf{z}}$  denotes the average with regard to variable  $\mathbf{z}$ . By substituting the reparametrized energy function from (21) into (23), we get

$$-\frac{\partial L}{\partial \theta} = \left\langle \frac{\partial \mathcal{G} \times_P \mathbf{x}^T \mathbf{A} \times_Q \mathbf{y}^T \mathbf{B} \times_R \mathbf{h}^T \mathbf{C}}{\partial \theta} \right\rangle_{\mathbf{h}} - \left\langle \frac{\partial \mathcal{G} \times_P \mathbf{x}^T \mathbf{A} \times_Q \mathbf{y}^T \mathbf{B} \times_R \mathbf{h}^T \mathbf{C}}{\partial \theta} \right\rangle_{\mathbf{h}, \mathbf{y}} \tag{24}$$

Equation (24) calculates the derivative of the (negative) log probability with respect to any parameter in the refactored weights tensor: core tensor  $\mathcal{G}$  and the three factor matrices  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{C}$ .

Similar to the unrefactored gated RBM model presented in Section 3, note that the derivative of the (negative) log probability with respect to any parameter of the Tucker refactored gated RBM is given by the difference of two expectations. The first expectation in (24) is over the posterior distribution over the hidden units. On the other hand, the second expectation in (24) is over all possible output/hidden instantiations and is intractable.

Note that the first term in (24) amounts to inferring the transformation (encoding)  $\mathbf{h}$  from a given pair of observed inputs  $\mathbf{x}$  and  $\mathbf{y}$  as considered in Equation (12). It is possible to plug the Tucker refactored energy function from (22) into (12) (bias term dropped for clarity), which becomes:

$$\hat{\mathbf{h}} = \sigma_h \left( \mathcal{G} \times_I \hat{\mathbf{x}} \times_J \hat{\mathbf{y}} \times_K \mathbf{C} \right). \tag{25}$$

In an analogous way, we may consider the task to compute  $p(\hat{\mathbf{y}} | \mathbf{h}, \mathbf{x})$  from an input image  $\mathbf{x}$  and a given fixed transformation  $\mathbf{h}$  considered in Equation (13). By plugging the Tucker decomposed energy function from (22) into (13) (bias term dropped for clarity), when input and hidden units are given, we get

$$\hat{\mathbf{y}} = \sigma_y \left( \mathcal{G} \times_I \hat{\mathbf{x}} \times_K \hat{\mathbf{h}} \times_J \mathbf{B} \right). \tag{26}$$

Let us now focus again on Equation (24). Note that the second term, also known as the *model expectation*, is an expectation over all possible instances of the output/mapping units and is intractable. However, similar to the bipartite structure in an RBM, the tripartite structure of a gated RBM facilitates Gibbs-sampling. With this in mind, we also consider the task to compute  $p(\hat{\mathbf{x}} | \mathbf{y}, \mathbf{h})$ .

$$\hat{\mathbf{x}} = \sigma_x \left( \mathcal{G} \times_J \hat{\mathbf{y}} \times_K \hat{\mathbf{h}} \times_I \mathbf{A} \right). \tag{27}$$

Then, Gibbs sampling suggests itself as a way to approximate the intractable term in (24). Because of the conditional independences of  $\mathbf{h}$  given  $\mathbf{y}$  and  $\mathbf{h}$ ,  $\mathbf{y}$  given  $\mathbf{x}$  and  $\mathbf{h}$ , and  $\mathbf{x}$  given  $\mathbf{y}$  and  $\mathbf{h}$ , we can easily sample from the conditional distributions  $p(\hat{\mathbf{h}} | \mathbf{x}, \mathbf{y})$ ,  $p(\hat{\mathbf{y}} | \mathbf{h}, \mathbf{x})$ , and  $p(\hat{\mathbf{x}} | \mathbf{y}, \mathbf{h})$  using (25)–(27) respectively.

Given the tripartite structure of the gated RBM, it is possible to perform three-way alternating Gibbs sampling. This scheme of optimizing an undirected graphical model is known as Contrastive Divergence. In this research, we perform a single Gibbs iteration when approximating the negative phase.

Using this Contrastive Divergence approach with Equations (25)–(27), we can make use of a machine learning library that supports reverse-mode automatic differentiation such as PyTorch.

PyTorch provides two high-level features: tensor computing with strong acceleration via GPU and automatic differentiation for all operations on tensors. Conceptually, this method for automatic differentiation records a graph recording all of the operations that created the data as the operations are executed. The leaves in the graph are the input tensors, and the roots are the output tensors. PyTorch traces this graph from roots to leaves, automatically computing the gradients using the chain rule. From a computational point of view, training a model consists of two phases: a forward pass to compute the value of the loss function and a backward pass to compute the gradients of the learnable parameters. With this in mind, we use the Contrastive Divergence approach for building the forward pass and generating the graph. This process is summarized in Algorithm 1.

---

**Algorithm 1:** Forward pass

---

**Input:**  $(x_i, y_i)$ : Training pair;  $k$  number of steps for CD learning

**Output:**  $\mathbf{x}^k$ : input vector once CD- $k$  is applied;  $\mathbf{y}^k$ : output vector once CD- $k$  is applied

1. Calculate positive phase performing three-way Gibbs sampling
    - $\mathbf{x}^0 \leftarrow x_1$
    - $\mathbf{y}^0 \leftarrow y_1$
    - Calculate  $p(\tilde{\mathbf{h}}^0 | \mathbf{x}^0, \mathbf{y}^0)$  using Equation (25).
    - $\mathbf{h}^0 \sim p(\tilde{\mathbf{h}}^0 | \mathbf{x}^0, \mathbf{y}^0)$
    - $\mathbf{h}^k \leftarrow \mathbf{h}^0$
  2. Calculate negative phase
    - For each step in  $k$ :
      - (a) Calculate  $p(\tilde{\mathbf{y}}^k | \mathbf{x}^0, \mathbf{h}^k)$  using Equation (26). Sample the  $\mathbf{y}^k$  states  
 $\mathbf{y}^k \sim p(\tilde{\mathbf{y}}^k | \mathbf{x}^0, \mathbf{h}^k)$
      - (b) Calculate  $p(\tilde{\mathbf{x}}^k | \mathbf{y}^0, \mathbf{h}^k)$  using Equation (27). Sample the  $\mathbf{x}^k$  states  
 $\mathbf{x}^k \sim p(\tilde{\mathbf{x}}^k | \mathbf{y}^0, \mathbf{h}^k)$
      - (c) Calculate  $p(\tilde{\mathbf{h}}^k | \mathbf{x}^0, \mathbf{y}^k)$  using Equation (25). Sample the  $\mathbf{h}^k$  states  
 $\mathbf{h}^k \sim p(\tilde{\mathbf{h}}^k | \mathbf{x}^0, \mathbf{y}^k)$
- 

When computing the forward pass, PyTorch simultaneously performs the requested computations and builds up a graph representing the function that computes the gradient. Once the forward pass is completed, this graph is evaluated in the backward pass to compute the gradients. To build the backward pass, we used the concept of *free energy* as presented in (9). Under the gated RBM, the probability of marginalizing a configuration of output units  $\mathbf{y}$  given the input units  $\mathbf{x}$  can be obtained by marginalizing out the hidden units. This computation is called free energy and is given by:

$$\begin{aligned}
 p(\mathbf{y} | \mathbf{x}) &= \sum_{\mathbf{h}} p(\mathbf{y}, \mathbf{h} | \mathbf{x}) \\
 &= \frac{1}{Z(\mathbf{x})} \sum_{\mathbf{h}} \exp(-E(\mathbf{y}, \mathbf{h}; \mathbf{x})) \\
 &= \frac{1}{Z(\mathbf{x})} \sum_{\mathbf{h}} \exp\left(\sum_{ijk} W_{ijk} x_i y_j h_k\right)
 \end{aligned} \tag{28}$$

Generally speaking, RBMs and gated RBMs are a class of models that belongs to the more general class of energy-based models (EBMs) [24]. EBMs have probabilistic equations of the following form:

$$p(\mathbf{x} | \theta) = \frac{1}{Z} \exp(-G(\mathbf{x} | \theta)) \tag{29}$$

where  $\mathbf{x}$  is the observed variables,  $Z$  is the normalization term, and  $G$  is the energy function. Since (28) yields the same form for the equation describing an EBM (29), we can then minimize the free energy function for maximum (log-)likelihood.

Unfortunately,  $p(\mathbf{y} | \mathbf{x})$  is intractable to compute since  $Z(\mathbf{x})$  involves an integration/sum over all possible settings of the input and hidden units:

$$Z(\mathbf{x}) = \sum_{\mathbf{h}'} \int_{\mathbf{y}'} \exp(-E(\mathbf{y}', \mathbf{h}'; \mathbf{x})) d\mathbf{y}'$$

However,  $\log p(\mathbf{y} | \mathbf{x})$  can be computed up to a constant, which is useful for scoring observation under a fixed model. First, we notice that Equation (28) involves a sum over all possible configurations of the hidden units  $\sum_{\mathbf{h}}$ , but we observe that the hidden units are binary. This means that we only need to consider two possible states for each unit. This observation leads to a non intractable form of the energy function:

$$\begin{aligned} p(\mathbf{y} | \mathbf{x}) &= \sum_{\mathbf{h} \in \{0,1\}^H} \exp(\sum_i \mathcal{W}_{i,j} \mathbf{x}_i \times_j \mathbf{y} \times_k \mathbf{h}) / Z \\ &= \sum_{h_1 \in \{0,1\}} \cdots \sum_{h_H \in \{0,1\}} \exp\left(\sum_k h_k \mathbf{W}_{::k} \mathbf{x} \mathbf{y}\right) / Z \\ &= \left(\sum_{h_1 \in \{0,1\}} \exp(h_1 \mathbf{W}_{::1} \mathbf{x} \mathbf{y})\right) \cdots \left(\sum_{h_H \in \{0,1\}} \exp(h_H \mathbf{W}_{::H} \mathbf{x} \mathbf{y})\right) \\ &= (1 + \exp(\mathbf{W}_{::1} \mathbf{x} \mathbf{y})) \cdots (1 + \exp(\mathbf{W}_{::H} \mathbf{x} \mathbf{y})) / Z \\ &= \exp(\log(1 + \exp(\mathbf{W}_{::1} \mathbf{x} \mathbf{y}))) \cdots \exp(\log(1 + \exp(\mathbf{W}_{::H} \mathbf{x} \mathbf{y}))) / Z \\ &= \exp\left(\sum_{k=1}^H \log(1 + \exp(\mathbf{W}_{::k} \mathbf{x} \mathbf{y}))\right) / Z \end{aligned}$$

Then the  $\log p(\mathbf{y} | \mathbf{x})$  is:

$$\begin{aligned} \log p(\mathbf{y} | \mathbf{x}) &= \log(\exp\left(\sum_1^H \log(1 + \exp(\mathbf{W}_{::k} \mathbf{x} \mathbf{y}))\right) / Z) \\ &= \left(\sum_1^H \log(1 + \exp(\mathbf{W}_{::k} \mathbf{x} \mathbf{y}))\right) / Z \end{aligned} \tag{30}$$

For scoring observations under a model, we can ignore the partition function  $Z$ . Finally, the backward pass can be computed using the free energy function in its scoring version (no partition function  $Z$ ) as it is shown in Algorithm 2.

---

**Algorithm 2:** Backward pass

---

**Input:**

$\mathbf{x}^0$ : input vector at step 0

$\mathbf{y}^0$ : output vector at step 0

$\mathbf{x}^k$ : input vector once CD- $k$  is applied

$\mathbf{y}^k$ : output vector once CD- $k$  is applied.

1. Calculate the free energy  $F(\mathbf{x}^0, \mathbf{y}^0)$  using Equation (30)
  2. Calculate the free energy with  $F(\mathbf{x}^k, \mathbf{y}^k)$  using Equation (30)
  3. Calculate the difference between the free energy  $F(\mathbf{x}^0, \mathbf{y}^0)$  and  $F(\mathbf{x}^k, \mathbf{y}^k)$
-

## 6. Results and Discussion

To illustrate the performance and viability of the model, we conducted experiments on pairs of shifted random binary images using the dataset provided on the accompanying website of Memisevic and Hinton [11].

We trained the model on pairs of transformed image patches, forcing the hidden variables to encode the transformations. The goal of this experiment is to investigate what forms the model weights take on when trained on affine transformations. The dataset consists of 10,000 binary image patch pairs of size  $13 \times 13$  pixels each, where the output image in each pair is a transformed version of the input image. The input images are shifted by one pixel in a random direction in each sample. As stated in Memisevic and Hinton [11] (p. 1482), there is no structure at all in the images themselves, which are composed of random pixels. The only source of structure in the data comes from the way the images transform.

Figure 7 shows three different samples of the binary images in the upper row. The lower row shows the shifted binary samples. This dataset was generated by a set of initial images where each pixel in the image is turned on randomly with probability 0.1. These initial images are used as input for the input layer  $x$  in the gated RBM model. Then, a random direction is chosen from the set {up, down, left, right, up-left, up-right, down-left, down-right, no shift} and each initial image is shifted by one pixel to create the output images. The newly appearing edges are filled randomly and independently as before with probability 0.1. The shifted images are used as input for the output layer  $y$  in the gated RBM model.

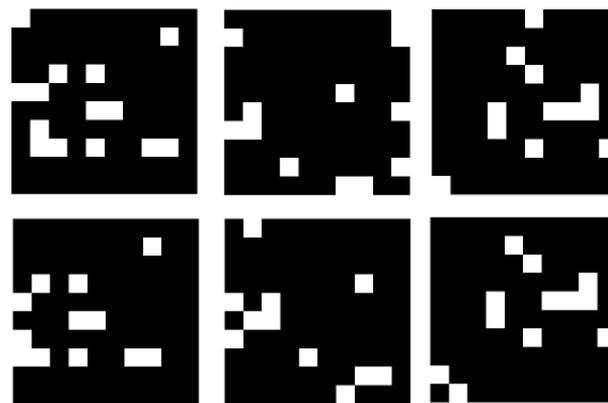
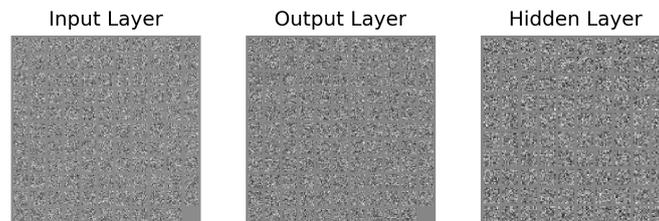


Figure 7. Samples of shifted binary images.

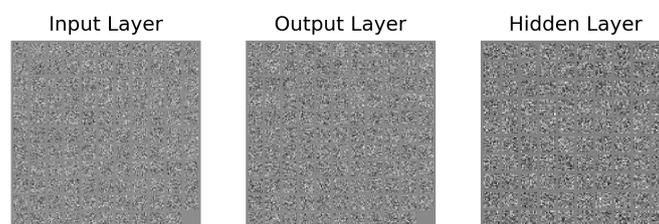
For this task, we trained a gated RBM with the proposed training algorithm from Section 5.3. We performed parameter exploration via grid search considering the following parameters: core tensor ranks, number of units in the hidden layer, and learning rate. We ranged the value of hidden units from 64 to 144 units. We did not identify significant changes in the filters learned given the number of units in the hidden layer or the core tensor ranks. The reason is that much of the interactions between the input and output layers are captured in the core tensor. The learning rate ranged from  $1 \times 10^{-2}$  to  $1 \times 10^{-4}$ , and the core tensor ranks evaluated were  $80 \times 80 \times 80$ ,  $120 \times 120 \times 120$  and  $169 \times 169 \times 169$ .

In Figure 8, we display the filters learned at different stages by the input, output, and hidden layers as a qualitative assessment of the trained gated RBM model. The filters displayed in Figure 8 correspond to a model with a core tensor with ranks  $120 \times 120 \times 120$ , 144 units in the hidden layer, and a learning rate of  $1 \times 10^{-3}$ . In the figure, each column in the factor matrices is rearranged to be displayed as a square. For example, if factor matrix  $A$  is of shape  $120 \times 169$ , then 120 squares of shape  $13 \times 13$  are displayed. From iteration 0 in Figure 8, we observe that the weights for each layer are randomly initialized. On each iteration, the weights become more structured with no supervision in the model. Note that the filters presented in iteration 211 in Figure 8 do not totally resemble the filters found

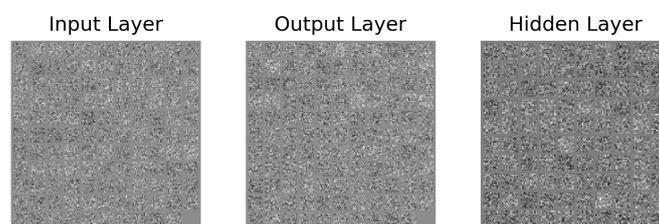
in [11] although the same dataset of random transformations is used. The reason is that the factorization of the three-way tensor proposed in this research is different to the one presented in [11]. While Memisevic and Hinton [11] project each mode of the three-way tensor in the gated RBM onto smaller layers, the current factorization involves a core tensor that models the interactions between the input and output image projections modulated via the gated connections in the hidden layer.



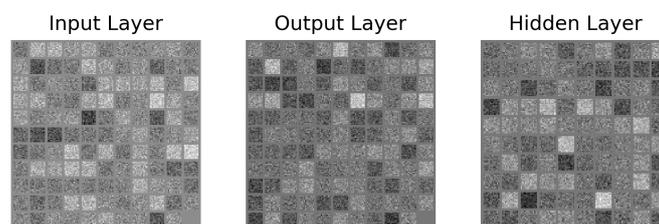
(a) Iteration 0



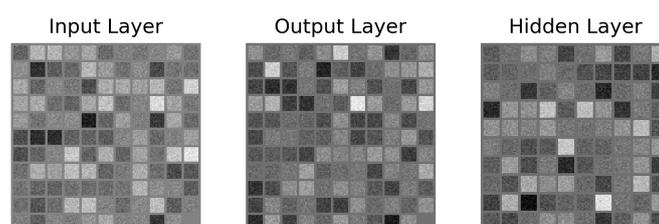
(b) Iteration 100



(c) Iteration 150



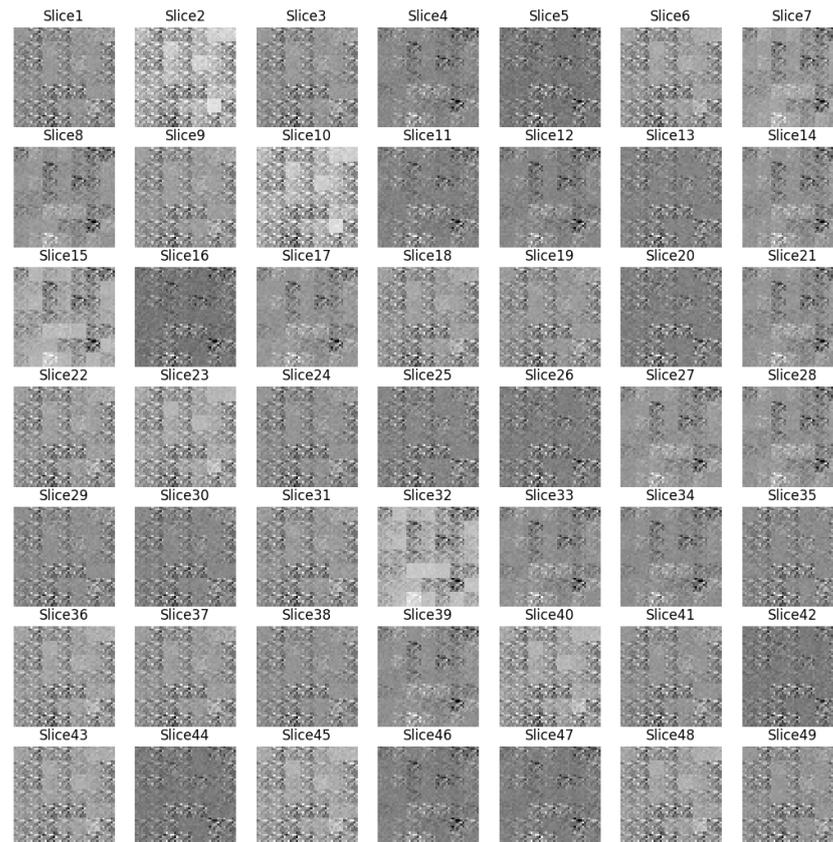
(d) Iteration 200



(e) Iteration 211

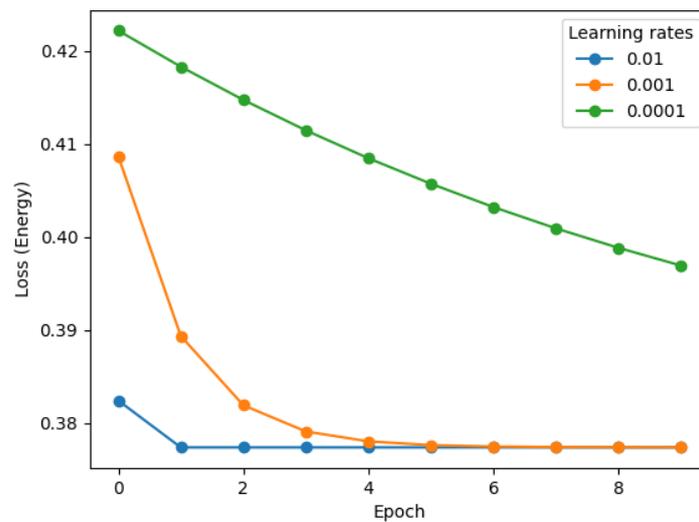
**Figure 8.** Image filters learned from shifted random images at different iterations.

In fact, the core tensor also learns filters, as presented in Figure 9. Moreover, the filters learned by each layer of the model show a correspondence with the filters in the three-way core tensor. In other words, each unit in the model is connected to the core tensor with variable strengths, and this allows them to detect specific changes in frequency, orientation, and phase shift in the data. In Figure 9, we show the filters learned by a model with a core tensor of shape  $49 \times 49 \times 49$ . In the image, we display each frontal slice of the core tensor. For simplicity, we present the frontal slices for this model with a smaller tensor, but the same behavior was observed regardless of the tensor shape.



**Figure 9.** Core tensor filters learned from shifted random images.

We also confirmed that the learning rate would affect convergence. The learning rate controls how much the parameters in the model are adjusted with respect to the energy loss gradient. Figure 10 shows energy loss during training for different learning rate configurations:  $1 \times 10^{-2}$ ,  $1 \times 10^{-3}$ , and  $1 \times 10^{-4}$ . We limit the number of epochs to 10. The number of units in the hidden layer is 144, and the ranks selected for the core tensor is  $120 \times 120 \times 120$ . As can be observed in the figure, the largest learning rate provides the fastest convergence in less than one epoch. On the other hand, the smallest learning rate decreases steadily on each epoch but does not arrive to a point of convergence. In each case, we observed that the corresponding model learns filters at different stages. In general, a smaller learning rate requires more training epochs given the smaller changes made to the model parameters in each update. However if the learning rate is too small, it can cause the learning process to get stuck and not converge. On the other hand, a larger learning rate results in rapid changes and requires fewer training epochs. However, it is possible that too large a learning rate causes the model to converge to a sub-optimal solution. The rate at which the learning takes place is an important hyper-parameter and will vary depending on the application of the model.



**Figure 10.** Gated RBM energy loss comparison with different learning rate configurations.

To gain insight into the way the core tensor affects the learning speed, we also trained the gated RBM model on the random-shifts dataset under various configurations of the core tensor while holding the learning rate and the number of hidden units constant. The gated RBM evaluated has 169 units both in its input and output layer and 169 units in its hidden layer. With this configuration, the original weight tensor has a shape of  $169 \times 169 \times 169$ . On the other hand, the core tensor was tested with ranks  $80 \times 80 \times 80$ ,  $120 \times 120 \times 120$ , and  $169 \times 169 \times 169$ . Note that in the last configuration, the core tensor provides no compression since it has the same shape as the original tensor. For simplicity, we only considered cubical tensors. The models were trained using the same learning rate of  $1 \times 10^{-3}$  for 10 epochs and were not stopped via early stopping. The idea is to isolate the effect of the core tensor configurations.

Figure 11 shows the energy loss when training under the three core tensor configurations. Note that in each case, the energy loss decreases steadily although with different gradients. We should remember that in a gated RBM as well as in an RBM, the concept of the energy function is analogous to the cost function. In fact, the configuration with the smallest core tensor has the highest energy decrease, while the core tensor configuration that is the exact shape of the original tensor has the lowest energy decrease. The reason is because the core tensor does not provide any compression but only functions as a change of basis. In addition, note that different configurations of the core tensor explore different configuration of the energy loss, which is explained by the fact that the core tensor modulates the level of interaction between the different components of each layer, resulting in different configurations of energy.

In Figure 12, we show the training time (in seconds) for the same core tensor configurations. As was expected, this figure confirms that the training speed is a function of the core tensor dimensionality. Although it could be inferred from Figures 11 and 12 that the best model configuration is the one with the smallest core tensor, it should be noted that the core tensor ranks should be calibrated according to the specifics of the problem to solve and evaluated according to the desired performance of the model.

As a last evaluation, we compared the Tucker refactored gated RBM proposed in this research against the unrefactored model presented in [10], which uses a full three-way tensor. The filters learned by the input, output, and hidden layers in combination with the filters learned by the core tensor are highly structural and in fact are very similar to the ones that are applied to the output images in [10], as shown in Figure 13b. The latter produces a canvas-like effect on the image. In both cases, the filters are learned in an unsupervised fashion. We should underline that there is no available code accompanying the research

in [10] and that we implemented their unfactored model using the update rules provided in the paper using PyTorch.

Although the filters produced both for the Tucker refactored and unfactored models might be similar, the training for both approaches has different performance metrics. To better understand this, we trained both models on the same random shifts dataset while holding all the variables constant. The input and output images have a size of  $13 \times 13$  pixels each, which determines the number of neurons in the input and output layers to be 169 in both models. The number of hidden units and learning rate are kept the same in both implementations and have values of 121 and  $1 \times 10^{-3}$ , respectively. Figure 14 shows the energy loss for the Tucker refactored, while Figure 15 shows the energy loss for the unfactored model.

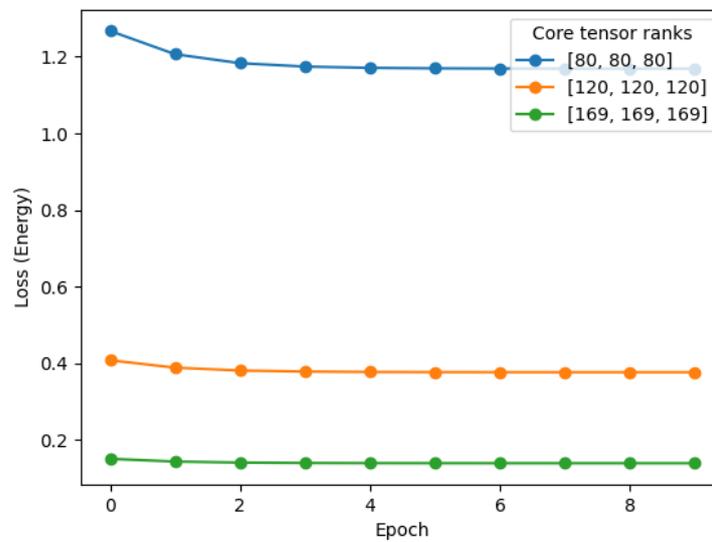


Figure 11. Gated RBM energy loss comparison with different core tensor ranks.

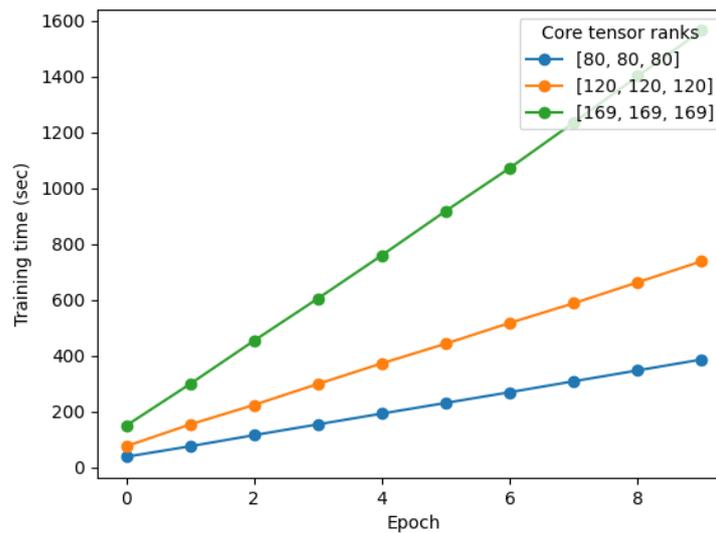
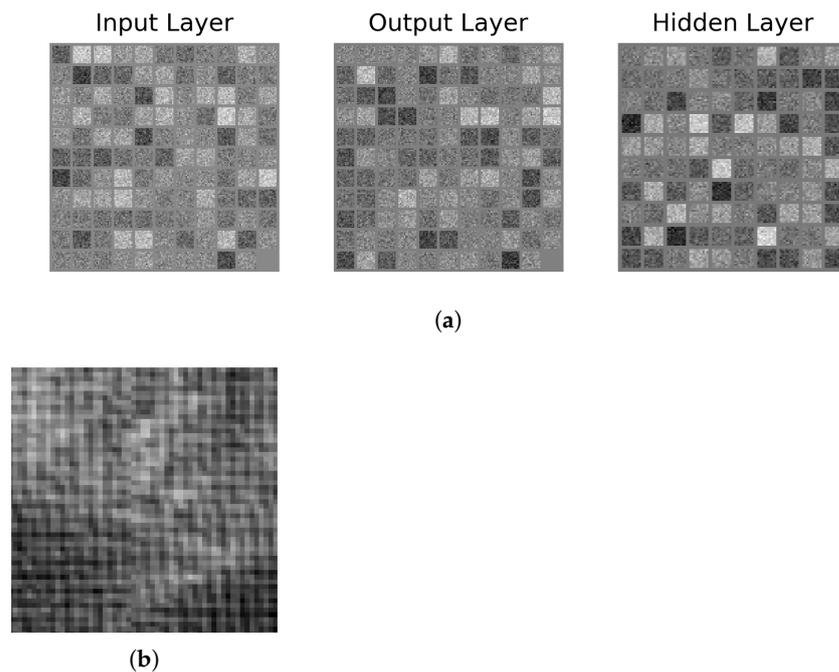
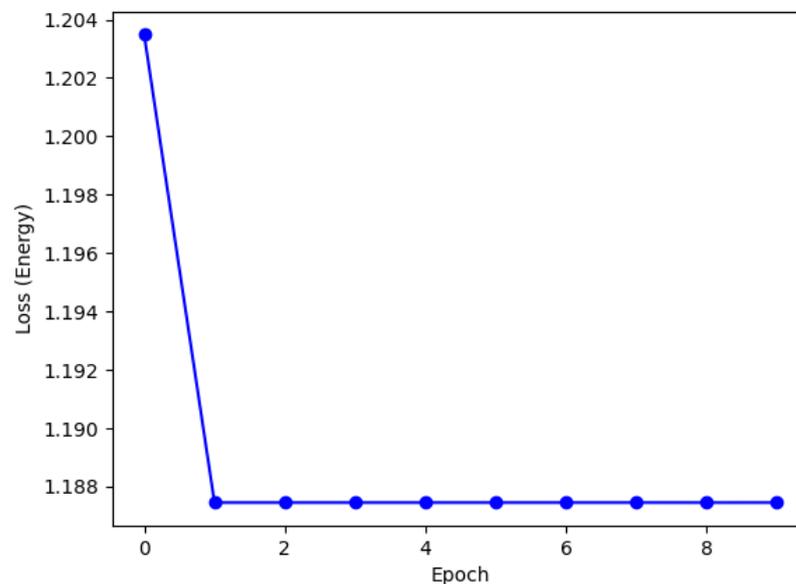


Figure 12. Gated RBM training time comparison with different core tensor ranks.

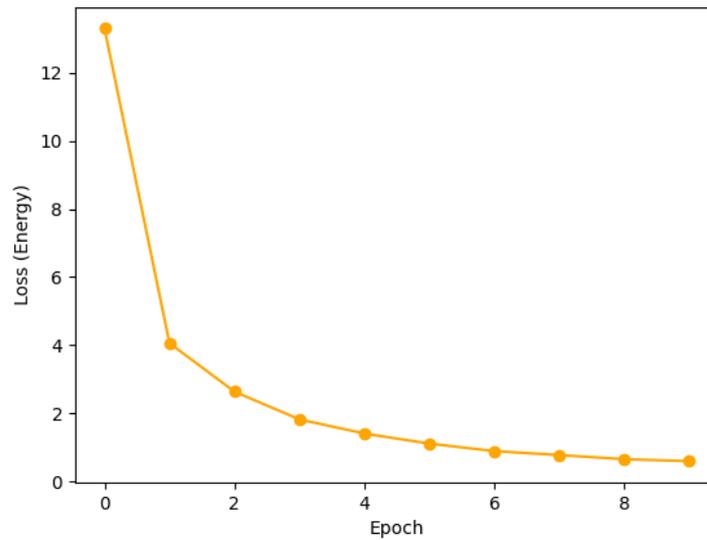


**Figure 13.** (a): Filters learned in by the input, output and hidden layers. (b): Filters applied from Memisevic and Hinton [10]. Reproduced with permission from R. Memisevic and G. Hinton, Unsupervised Learning of Image Transformations; published by IEEE, 2007.



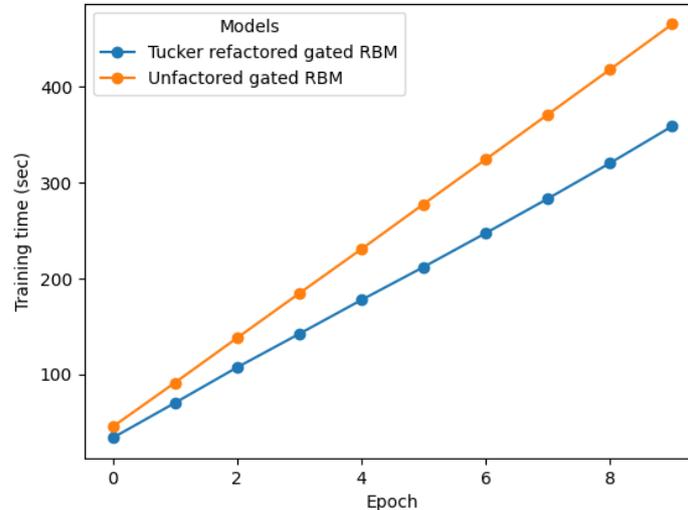
**Figure 14.** Energy loss when training the Tucker refactored model.

The only difference between the two models is that the unrefactored gated RBM uses a full tensor  $169 \times 121 \times 169$ , and the Tucker refactored model factorizes the full tensor into a core tensor  $80 \times 80 \times 80$ , input and output factor matrices  $169 \times 80$  and a hidden factor matrix  $121 \times 80$ . Note that the unrefactored gated RBM starts with a much higher energy configuration. On the other hand, the Tucker refactored model starts at a much smaller energy configuration as a result of the core tensor modulating the interactions of the three layers. The relationship between the core tensor configuration and the resulting energy configuration explored is also observable in Figure 14. From Figures 14 and 15, we see that the Tucker refactored gated RBM reaches convergence in fewer iterations than the unrefactored gated RBM while having the same learning rate and hidden units.



**Figure 15.** Energy loss when training the unfactored gated RBM from Memisevic and Hinton [10].

Finally, in Figure 16, we show a comparison of the training time (in seconds) for both the Tucker refactored and unfactored gated RBMs. It is easy to see that the unfactored model using a full tensor takes much longer than the Tucker decomposed factored model proposed in this research. This is because the core tensor reduces the number of free parameters in the model while maintaining its learning capacity.



**Figure 16.** Training time in Tucker refactored vs unfactored gated RBMs.

## 7. Conclusions

The multimodal tensor-based Tucker decomposition presented in this research has the useful property that it keeps the independent structure in the gated RBM model intact. We take advantage of this independent structure by developing a contrastive-divergence-based training procedure used for inference and learning.

In this paper, we combine Tucker Decomposition for a gated RBM and showed how the model allows us to obtain image filter pairs that are highly structural when trained on transformations of images. There is some resemblance between our approach and the bilinear model for Visual Question Answering (VQA) task proposed in [25]; however, the problems solved and the learning methods presented are quite different. Despite its appealing modeling power, the literature on gated RBMs is still scarce. The literature

review on gated RBMs revealed that almost all the publications on the subject present only different applications for the model, namely texture modeling [26], classification [27], and rotation representations [28]. In that sense, this research contributes an alternative method for training the gated RBM.

As we have seen, Tucker decomposition is a great tool for multidimensional data dimensionality reduction. Implementing it in the gated RBM means adding an additional hyperparameter to the model: the multimodal shape of the core tensor. In fact, the resulting model allows us to explicitly control the model complexity and to choose an accurate and interpretable factorization of the learnable parameters. One important property of tensor decomposition is that the number of parameters from the core tensor and factor matrices is usually much smaller than the number of parameters in the original tensor. Using the compression ratio (number of elements before and after tensor decomposition) and the approximation error caused by tensor decomposition, we can evaluate the proper decomposition to be performed.

Time complexity for inference or one-step inference in the Tucker decomposed gated RBM is  $O(n_p n_q n_r)$ , where  $n_p$ ,  $n_q$ , and  $n_r$  correspond to the dimensionality of each mode in the core tensor and directly impact the modeling complexity that will be allowed for each modality. Note that the core tensor can be fixed to a small rank regardless of the dimensionality of each layer in the weight tensor. This is in contrast to the time complexity for inference in a fully parameterized gated RBM, which is  $O(n_i n_j n_k)$ . The latter is in terms of the dimensionality of the input, output, and hidden layers in the fully parametrized gated RBM and is not efficient in terms of memory with large inputs.

Strictly speaking, we could also select the dimension of each modality  $n_p$ ,  $n_q$ ,  $n_r$  to be equal to or greater than  $n_i$ ,  $n_j$ ,  $n_k$ . This would lead to a change of basis or an explosion in the number of free parameters. There are several interesting directions for future work. In this research, we used a fixed shape in the core tensor; however, fine tuning this new hyperparameter needs to be addressed in future research. A possible idea is to measure the approximating error and select the smallest multimode dimensionality that meets a selected threshold. Another idea is to use the tripartite structure of the gated RBM to model discriminative tasks in which an image on layer  $x$  and its corresponding target in layer  $y$  are provided.

**Author Contributions:** Conceptualization, M.M.-C. and A.M.-V.; methodology, M.M.-C. and A.M.-V.; software, M.M.-C.; validation, M.M.-C., A.M.-V. and R.O.G.-M.; formal analysis, M.M.-C., A.M.-V. and R.O.G.-M.; investigation, M.M.-C.; resources, M.M.-C. and A.M.-V.; data curation, M.M.-C.; writing—original draft preparation, M.M.-C.; writing—review and editing, M.M.-C., A.M.-V. and R.O.G.-M.; visualization, M.M.-C.; supervision, M.M.-C., A.M.-V. and R.O.G.-M.; project administration, M.M.-C., A.M.-V. and R.O.G.-M.; funding acquisition, A.M.-V. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Consejo Nacional de Ciencia y Tecnologia (Mexico).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The implemented code that supports the findings of this research are openly available at [https://github.com/macma80/tucker\\_gatedrbm](https://github.com/macma80/tucker_gatedrbm) (accessed on 6 August 2021).

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

RBM	Restricted Boltzmann machine
CD	Contrastive Divergence
GPU	Graphics processing unit
EBM	Energy-based model
VQA	Visual Question Answering

## References

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [[CrossRef](#)] [[PubMed](#)]
2. Smolensky, P. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*; Technical report; Colorado University at Boulder Department of Computer Science: Boulder, CO, USA, 1986.
3. Hinton, G.E. Training products of experts by minimizing contrastive divergence. *Neural Comput.* **2002**, *14*, 1771–1800. [[CrossRef](#)] [[PubMed](#)]
4. Bourlard, H.; Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biol. Cybern.* **1988**, *59*, 291–294. [[CrossRef](#)] [[PubMed](#)]
5. Hinton, G.E.; Zemel, R.S. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 1994; pp. 3–10.
6. LeCun, Y. Generalization and network design strategies. *Connect. Perspect.* **1989**, *19*, 143–155.
7. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 1995; Volume 3361, p. 1995.
8. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533. [[CrossRef](#)]
9. Sigaud, O.; Masson, C.; Filliat, D.; Stulp, F. Gated networks: An inventory. *arXiv* **2015**, arXiv:1512.03201.
10. Memisevic, R.; Hinton, G. Unsupervised learning of image transformations. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
11. Memisevic, R.; Hinton, G.E. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Comput.* **2010**, *22*, 1473–1492. [[CrossRef](#)] [[PubMed](#)]
12. Sejnowski, T.J. Higher-order Boltzmann machines. In *AIP Conference Proceedings*; AIP: New York, NY, USA, 1986; Volume 151, pp. 398–403.
13. Carreira-Perpinan, M.A.; Hinton, G.E. On contrastive divergence learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*; PMLR: New York, NY, USA, 2015; Volume 10, pp. 33–40.
14. Bengio, Y.; Delalleau, O. Justifying and generalizing contrastive divergence. *Neural Comput.* **2009**, *21*, 1601–1621. [[CrossRef](#)] [[PubMed](#)]
15. Sutskever, I.; Tieleman, T. On the convergence properties of contrastive divergence. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 789–795.
16. Tieleman, T. Training restricted Boltzmann machines using approximations to the likelihood gradient. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; ACM: New York, NY, USA, 2008; pp. 1064–1071.
17. Tieleman, T.; Hinton, G. Using fast weights to improve persistent contrastive divergence. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; ACM: New York, NY, USA, 2009; pp. 1033–1040.
18. Desjardins, G.; Courville, A.; Bengio, Y.; Vincent, P.; Delalleau, O. Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines. In Proceedings of the Thirteenth International Conference on artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 145–152.
19. Tucker, L.R. Implications of factor analysis of three-way matrices for measurement of change. *Probl. Meas. Chang.* **1963**, *15*, 122–137.
20. Levin, J. Three-mode factor analysis. *Psychol. Bull.* **1965**, *64*, 442. [[CrossRef](#)] [[PubMed](#)]
21. Tucker, L.R. The extension of factor analysis to three-dimensional matrices. In *Contributions to Mathematical Psychology*; Holt, Rinehart and Winston: New York, NY, USA, 1964; pp. 110–127.
22. Kolda, T.G. *Multilinear Operators for Higher-Order Decompositions*; Technical Report; Sandia National Laboratories: Albuquerque, NM, USA, 2006.
23. Kolda, T.G.; Bader, B.W. Tensor decompositions and applications. *SIAM Rev.* **2009**, *51*, 455–500. [[CrossRef](#)]
24. LeCun, Y.; Chopra, S.; Hadsell, R.; Ranzato, M.; Huang, F. A tutorial on energy-based learning. In *Predicting Structured Data*; Bakir, G., Hofman, T., Scholkopf, B., Smola, A., Taskar, B., Eds.; MIT Press: New York, NY, USA, 2006; Volume 1.
25. Ben-Younes, H.; Cadene, R.; Cord, M.; Thome, N. Mutan: Multimodal tucker fusion for visual question answering. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2612–2620.
26. Hao, T.; Raiko, T.; Ilin, A.; Karhunen, J. Gated Boltzmann machine in texture modeling. In *International Conference on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 124–131.
27. Sorokin, I. Classification factored gated restricted Boltzmann machine. In *Proceedings of the 1st International Workshop on Advanced Analytics and Learning on Temporal Data (AALTD)*; CEUR-WS.ORG: Porto, Portugal, 2015; pp. 131–136.
28. Giuffrida, M.V.; Tsafaris, S.A. Theta-rbm: Unfactored gated restricted boltzmann machine for rotation-invariant representations. *arXiv* **2016**, arXiv:1606.08805.