

# Semi-automatic CNN Architectural Pruning using the Bayesian Case Model and Dimensionality Reduction Visualization

Wilson E. Marcílio-Jr.<sup>1</sup> <sup>a</sup>, Danilo M. Eler<sup>1</sup> <sup>b</sup> and Ivan R. Guilherme<sup>2</sup> <sup>c</sup>

<sup>1</sup>Department of Mathematics and Computer Science, São Paulo State University - UNESP, Presidente Prudente, SP, Brazil

<sup>2</sup>Department of Statistics, Applied Math. and Computing, São Paulo State University - UNESP, Rio Claro, SP, Brazil

Keywords: CNN Pruning, Case-based Reasoning, Visualization.

Abstract: Visualization techniques have been applied to reasoning about complex machine learning models. These visual approaches aim to enhance the understanding of black-box models' decisions or guide in hyperparameters configuration, such as the number of layers and neurons/filters in deep neural networks. While several works address the architectural tuning of convolutional neural networks (CNNs), only a few works face the problem from a semi-automatic perspective. This work presents a novel application of the Bayesian Case Model that uses visualization strategies to convey the most important filters of convolutional layers for image classification. A heatmap coordinated with a scatterplot visualization emphasizes the filters with the most contribution to the CNN prediction. Our methodology is evaluated on a case study using the MNIST dataset.

## 1 INTRODUCTION


Visualization techniques play a major role in understanding the learning patterns of deep neural networks due to their low interpretation abilities. For example, visual approaches are created to enhance understanding of how convolutional neural networks apply series of non-linear equations to images (Liu et al., 2017a; Zeiler and Fergus, 2014), how attention mechanisms can uncover dependencies among words in recurrent neural networks (Strobel et al., 2018), and how to progressively inspect the training process (Rauber et al., 2017; Pezzotti et al., 2018; Marcilio-Jr et al., 2020).


The design of neural networks is usually a time-consuming task where various decisions, such as the number of layers, number of neurons in each layer, and other aspects must be taken into account (Goodfellow et al., 2016). The decision of these parameters is usually taken based on the practitioners' intuition or various *trial-and-error* iterations. Usually, the first step consists of choosing a complex architecture to perform the learning task. Then, such architecture is refined so that simpler models could maintain performance while requiring a much lower number of pa-


rameters. In particular, the refinement of the architecture is performed by inspecting the model after training. For example, one could remove all the close to zero parameters in a feed-forward neural network. To this end, visualization techniques also play a major role in understanding the processes involving neural networks. One beneficial approach is to use Visual Analytics tools to help define model architecture, as shown by Garcia et al. (Garcia et al., 2019), that provides visual metaphors based on heatmaps to uncover and emphasize filters that could be removed from convolutional neural networks.

However, the literature lacks techniques that indicate which filters are prone to be removed from inspected layers, i.e., using semi-automatic strategies. In this scenario, users would benefit from the cues given by the automatic techniques while assessing the performance of such cues using visual representations.

In this work, we propose a novel application of the Bayesian Case Model (BCM) (Kim et al., 2014) to uncover the most important convolutional filters of CNNs applied to image classification. The result of BCM, which consists of data samples that most describe datasets, is visualized through a scatterplot-based visualization after dimensionality reduction, which has been widely applied to investigate the learning process of deep learning models (Rauber et al., 2017; Marcílio-Jr et al., 2021b; Marcílio-Jr

<sup>a</sup>  <https://orcid.org/0000-0002-8580-2779>

<sup>b</sup>  <https://orcid.org/0000-0002-9493-145X>

<sup>c</sup>  <https://orcid.org/0000-0002-3610-3779>

et al., 2021a). The scatterplot visualization is coordinated with a heatmap summarizing an activation map that emphasizes which filters could be candidates for removal. Thus, instead of automatically pruning the network with the result of BCM, our approach enhances the trust in the final refinement by letting the users decide which filters—among those indicated by BCM—will be removed from the network. Finally, our approach is evaluated through a case study on the classification of the MNIST dataset. Summarily, the contributions of this paper are:

- Application of the Bayesian Case Model for pruning convolutional neural network;
- A visualization approach for supporting in the network pruning.

This work is organized as follows: in Section 2 we present the related works; Section 3 focus on the explanation of the Bayesian Case Model; our methodology is delineated in Section 4; a use case is provided in Section 5; discussions about the work are provided in Section 6; finally, the work is concluded in Section 7.

## 2 RELATED WORKS

Using visualization strategies to understand the caveats and decisions taken by neural networks has been a trending topic in the visualization community in the past few years, mainly due to the need to explain these complex models.

According to Pezzotti et al. (Pezzotti et al., 2018), the visualization techniques designed to enhance the analysis of deep neural networks (DNNs) can be divided into three groups: weight-centric, dataset-centric, and filter-centric techniques. The *weight-centric* techniques aim at understanding the relationship among the weight learned by the networks so that the learning process can be understood based on a combination of input-weight-output. These approaches use node-link diagrams (Reingold and Tilford, 1981), and visual clutter is reduced by neuron aggregation and edge bundle (Liu et al., 2017a). *Dataset-centric* techniques are usually employed to understand the training process of neural networks from a higher point of view, such as using dimensionality reduction algorithms to understand the training evolution of such techniques (Rauber et al., 2017) or using star-plots to monitor the evolution of the training process as well as comparing different deep learning models (Marcilio-Jr et al., 2020). Finally, *filter-centric* methodologies aim to understand the patterns learned by the filters, for example, by visualizing the

relationships among filters and the labels associated with them (Rauber et al., 2017). Garcia et al. (Garcia et al., 2019), for example, use activation maps to communicate the filter’s redundancy to help in architectural tuning. Hohman et al. (Hohman et al., 2020) use aggregation to visualize which features deep learning models have learned and how these features interact inside the model to produce predictions. Clavien et al. (Clavien et al., 2019) utilize heatmap representations to progressively visualize the neuron’s activation during the training of deep learning models.

More related to our work are techniques aiming to reduce models by pruning filters. The majority of the methods propose automatic approaches that uses thresholds to remove filters from CNNs (Luo et al., 2017; Liu et al., 2017b; He et al., 2017; Yu et al., 2018; Dubey et al., 2018; He et al., 2019). For instance, Li et al. (Li et al., 2021) propose a visual analytics system to provide users with a better view of the convolutional filters of CNNs and support them with pruning plans. Thus, our work lies between these two approaches since we indicate to users which filters may be good for removal but leave the final to them the final decision.

This work presents a semi-automatic approach that uses visualization strategies to help in the architectural pruning of convolutional neural networks. First, we briefly introduce the Bayesian Case Model (BCM) to explain how it searches for features that explain clustering results. Then, we introduce our approach that uses the BCM output to generate hints for filter removal.

## 3 BAYESIAN CASE MODEL

The Bayesian Case Model (BCM) (Kim et al., 2014) is an interpretable model that tries to describe the latent space of a high-dimensional dataset through clusters’ prototypes and their subspaces. These prototypes (that correspond to actual data points) are described by several features that characterize the data points. In other words, the prototypes are described by a series of features that are important to them.

BCM execution starts with a standard discrete mixture model (Hofmann, 1999) to represent the structure of the data points. Since only the mixture model cannot interpret the clustering result, in the sense that which features contribute the most for the cluster formation, BCM augments the mixture model result by adding the prototypes and subspace feature indicators to characterize/explain the clusters. A few parameters are involved in the BCM execution, as shown in Fig. 1.

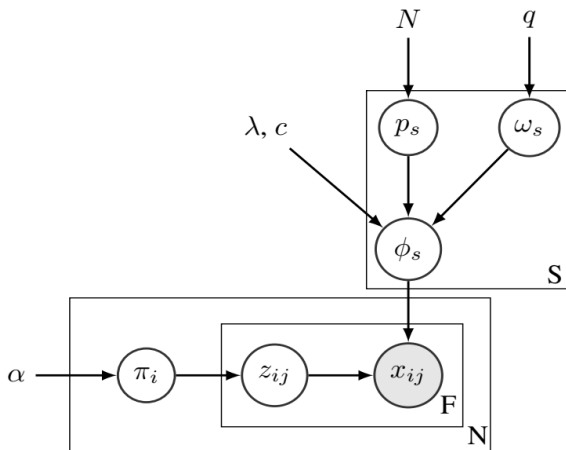


Figure 1: Graphical representation of BCM technique (Kim et al., 2014).

The graphical representation indicates all the components used in BCM. Firstly, the algorithm starts with  $N$  observations ( $X = \{x_1, x_2, \dots, x_N\}$ ), where each observation ( $x_i$ ) represent a random mixture over  $S$  clusters. At the beginning of the algorithm, each data point is described by each cluster with random contributions. These contributions are indicated in Fig. 1 by the mixture weights ( $\pi_i \in \mathbb{R}_+^S$ ) that associate a number to each data point, denoted by  $x_{ij}$ . Each of these features is given from one cluster, meaning that which cluster (or subspace) contributes the most to defining the data point. The indication of which cluster defines feature  $x_{ij}$  is denoted by  $z_{ij}$ , and it assumes a value between 1 and  $S$ . The hyper-parameters  $q$ ,  $\lambda$ ,  $c$ , and  $\alpha$  are assumed to be fixed.

BCM is trained so that it returns the prototypes and their corresponding subspaces. While the prototypes correspond to actual data points, the subspaces are a way to tell which features contribute the most to defining the clusters—represented by the prototypes. A complete description of the BCM training process is out of the scope of this paper, and interested readers can refer to the original article by Kim et al. (Kim et al., 2014).

## 4 METHODOLOGY

Defining the architecture of deep learning models is usually a process carried out based on the practitioners' experience. One of the most employed approaches is to start with a complex model (e.g., too many layers, neurons, or convolutional filters) and then refine the architecture while retraining the model to investigate its ability to generalize well for the data. In this case, approaches that help machine learning

practitioners to spend less time on architectural tuning are of the great value of rapid prototyping.

Our methodology for architectural tuning is focused on the convolutional layers of convolutional neural networks (CNNs). Consider the CNN architecture of Fig. 2, which we also use in the use case (see Section 5). After the input, the two convolutional layers are responsible for extracting images' features. The dense part of the network contains neurons that use these characteristics to discriminate among classes.

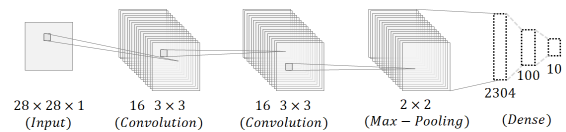


Figure 2: CNN architecture used to train MNIST dataset.

After the training step, we can input an image to the convolutional filters to investigate the importance of that image's structures. In other words, the convolutional process of an input image to the trained layers will result in values that show how each filter assigns importance to that input image and its structures. Given an input dataset  $X$  that consists of  $|X|$  images, the set of activation images for an input image  $X_i$  results from applying a convolutional layer  $l_k$  and is denoted as  $l_k(X_i)$ . The application of  $l_k$  to an input image  $X_i$  returns  $m$  activation images, resulted from convolution of  $m$  filters by image  $X_i$ . For example, in Fig. 2, the application of layer  $l_2$  to an input image  $X_i$  ( $l_2(X_i)$ ) results in 16 activation images since it contains 16 convolutional filters.

For a single input image, our task is to provide the users with the filters that would be the candidates for the removal. Thus, we use BCM to return such information based on the activation images of that image. Since BCM accepts a matrix of observations by dimensions, we transpose the filter activations of each input image into dimensions. So, we transpose the matrix of flattened convolutional filters, as shown in Fig. 3. Before feeding such a matrix to BCM, we first linearly scale each pixel of filter activation to the range  $\in [0, 10]$  to decrease the computing time of the BCM algorithm. From the output of BCM, we are only interested in the subspaces, i.e., the features of the matrix and, consequently, the filters that are most important for the input images.

Notice that by transposing the matrix of flattened convolutional filters (see Fig. 3), we construct a matrix in which each filter corresponds to a column. We choose to represent the filters as columns since BCM returns prototypes (rows of the matrix) and the associated dimensions (columns of the matrix) used to de-

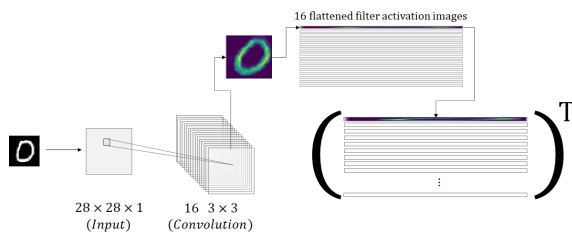


Figure 3: Generating input matrix for BCM technique. For each input image,  $m$  activation images are generated, where  $m$  is the number of filters in the analyzed layer. These activation images are flattened and then transposed to be fed to the BCM technique.

scribe the prototype. Thus, choosing only one prototype for the matrix created using the activation images of an input image, BCM returns the dimensions used to describe all of the matrices, in other words, the most important filters. We discard the prototype in our application.

Finally, since we have to repeat the process described for each input image, we select from the test set only a representative set to apply the BCM. Such a representative set is constructed using the SADIRE (Marcilio-Jr, 2020) sampling selection technique, which builds the representative set after dimensionality reduction (DR) and can preserve most of the structures imposed by the DR technique.

#### 4.1 Visualization Design

To visualize the result of BCM after selecting the most important filters, we used coordination between scatterplot and heatmap visualizations, as presented in Fig. 4. In this context, the coordination between these two visualizations means that the interaction in one view (e.g., scatterplot) results in a change in the other (e.g., heatmap). The main view, the scatterplot is shown in Fig. 4(a) allows users to investigate the separation among classes imposed by the learning features from the CNN. When interested in a particular data sample, the interaction with the circles (through mouse clicks) makes the visualization update the *Activation Map* and *BCM Heatmap*, as shown in Fig. 4(b) for samples of classes 0, 1, and 9. While the *Activation Map* shows the filter activations for that image, and the *BCM Heatmap* shows which filters BCM considers important for subspace definition.

As commented in the previous section, we used a sampling approach to feed BCM with fewer data samples. Fig. 4(a) shows a subset (the sampled data points) used for computing the most important filters using BCM, where circle area encodes the number of points represented by each sampled point—notice that the selected data points are evenly

distributed throughout the projected space since we used SADIRE (Marcilio-Jr, 2020) sampling technique. When users click on data points, the filter activations for the sample are visualized in the activation map (see Fig. 4(b)). The results of the BCM algorithm are shown in the BCM heatmap, where each cell of the heatmap corresponds to the respective filter activation in the *Activation Map*. The dark-blue cells indicate the filters in which BCM is judged important, while light-blue cells indicate filters not as important for defining the subspace.

Fig. 4 shows the state of the visualization after clicking on ten samples of different classes in the scatterplot representation (a)—the visualization is updated with the activation maps and BCM output for the ten points. The example highlighted in red further explain the process for and point from class five.

It is worth emphasizing that an automatic approach to extracting the most important filters after BCM and then refining the network would be possible. However, our visualization strategy allows users to trust in the final refinement due to their expertise in tuning network hyperparameters. Finally, the visualization also decreases the chances in which BCM may fail to capture the contribution of each filter to define the subspace of a particular image.

## 5 USE CASE

In this use case, we employ BCM, and the visualization design explained in Section 4 to prune filters from convolutional layers of a CNN trained to classify the MNIST (LeCun and Cortes, 2010) dataset, which consists of hand-written digits. The dataset comprises 50k training images, 10k validation images, and 10k test images. The CNN architecture is described as follows (also, see Fig. 2):

1. A  $28 \times 28 \times 1$  input image;
2. A Convolutional Layer I:  $26 \times 3 \times 3$  filters;
3. A Convolutional Layer II:  $16 \times 3 \times 3$  filters;
4. A  $2 \times 2$  max-pooling layer (dropout 0.25);
5. A fully connected layer;
6. A dense layer with 100 neurons (dropout 0.25);
7. A soft-max output layer with 10 neurons.

The CNN was trained during 500 epochs, achieving 0.9243 of accuracy and 0.25704 of loss. From the projection of the test set in Fig. 4, we can see that imposing a separation to this dataset is a relatively easy task. The primary source of error comes from the similarity of digits' traits, such as for the digits six and



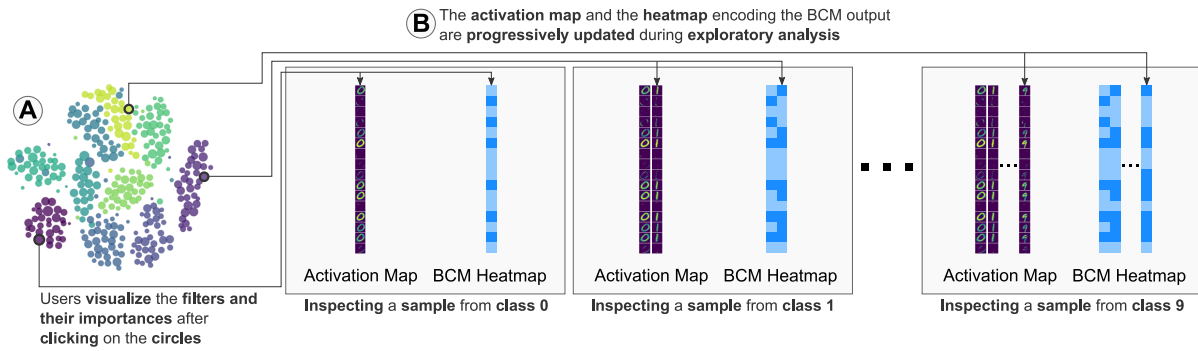


Figure 4: Assessing feature importance using Bayesian Case Model (BCM) and visualization techniques. The projection of the test set can be seen by the scatter plot visualization in (a) to help users make sense of the similarity among data points. After clicking on a circle of the projection, all of the activation filters are shown in another view, as seen in (b). The important filters highlighted by BCM are encoded as a heatmap, where each cell encodes a filter, and darker blue encodes the filters defined as important.

five (6 and 5). With such an idea in mind, it could be interesting to verify if all the filters of a given convolutional layer contribute to the classification of the hand-written digits. In this case, users would know if their architecture needs to be more complex—when all the filters contribute to the prediction, but the loss is high—or if its architecture could be pruned—when the model is already showing good performance.

We already know that our architecture is performing quite well on the MNIST dataset. Our methodology could be applied to find candidates to reduce the number of parameters while maintaining as much of the model’s performance as possible. After computing the BCM for each  $l_2(X_i^{test})$  of the test set—that is, for each set of filter activations for all data observations in the test set—we selected the filters highlighted in orange of Fig. 5 as the important ones, that is, the remaining of the filters were removed from the Convolutional Layer II. Then, predicting the test set with the pruned architecture resulted in 0.9278 of accuracy and 0.247421 of loss. Here, removing the filters that were not doing a valuable job to help the model discriminate hand-written digits increased the accuracy and reduced the loss. Such a result could be explained by the fact that the filters with no useful contribution may introduce artifacts that can confuse the model when predicting the class of the digits with a complex trait so that removing these filters from the network could reduce the probability of such errors.

One may notice in Fig. 5 that a few filters seem to present very similar activation patterns. When designing a CNN, each filter of a convolutional layer is meant to capture a different characteristic of the input data, such as information about borders, textures, and shape. So, if two or more filters present similar activations, it means they are redundant. The redundancy has the same problem with filters that do not activate

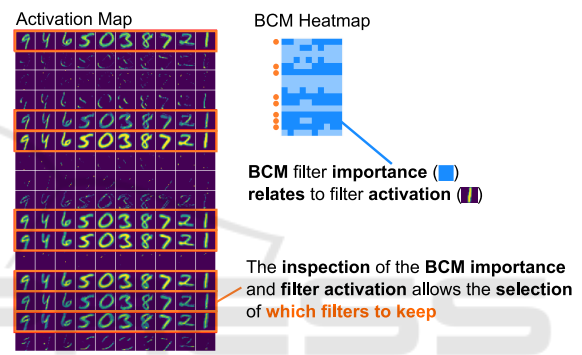


Figure 5: Highlighting important filters based on the importance returned by BCM (*BCM heatmap*) and the visual inspection of the activations in the *Activation Map*.

at all, and they do not add information that the following layers could use to discriminate among digits. As a result, redundant filters can also be removed from the convolutional layer.

Fig. 6 shows the features kept in the final configuration of Convolutional Layer II with redundancy removal in mind. In this case, by visualizing the feature activations classified as important by the BCM technique, consecutive filters that express similar activation were removed together with the features expressing no contribution. After predicting the test set with the filter configuration of Fig. 6 the model achieved 0.9281 of accuracy and 0.24984 of loss.

Although we had only a slight gain in the performance after filter pruning, augmenting the performance of deep learning models is a difficult task when the model is already presenting good results (Raubert et al., 2017). Moreover, removing filters means removing computational operations, which leads to a decrease in time execution to train and prediction—a big challenge for deep learning models. Lastly, by successively using our methodology, practitioners

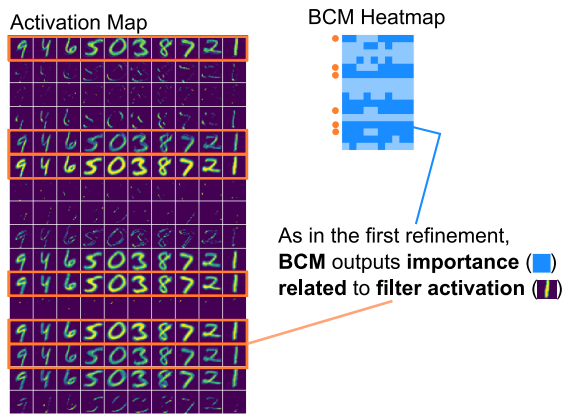


Figure 6: Using visual inspection to remove redundancy among convolutional filters. Besides identifying useful filters, our approach allows users to discover redundancy among filters visually—redundant filters contribute similarly and can be removed from the convolutional layer.

could build an intuition on designing more efficient deep learning models. The performance of CNN before and after the two refinements is shown in Table 1.

Table 1: Performance of the CNN before and after two refinements.

Model	Loss	Accuracy
Initial	0.25794	0.9243
After refinement #1	0.24742 ↓	0.9278 ↑
After refinement #2	0.24984 ↑	0.9281 ↑

## 6 DISCUSSION

We demonstrated the usefulness of our approach during the experimentation section to aid in the architectural tuning of a convolutional neural network. By employing scatterplot and heatmap visual metaphors to emphasize the similarity among data points and the importance of the filters, users can get an overview of how filters react with inputs presenting higher or lower similarity. Such a task is further improved by the coordination mechanism that draws columns of filter activations and filter importance as users click on data points of interest. In this case, given two or more data points, users can quickly inspect if filters contribute in a contrastive fashion, contribute equally, or even if filters do not contribute at all to the model’s prediction.

Another interesting application of our approach consists of allowing users to investigate the filters for a particular cluster of images, supported by the coordination between the scatterplot and the activation map. Focusing on a group of interest, users may un-

derstand the activation patterns and how the convolutional neural network made the decisions to the classification.

One limitation of our methodology is related to the computational complexity of the Bayesian Case Model and by the fact that a matrix must be generated for each input image, as explained in Section 4. Although this problem can be reduced using representative data points of the dataset used in this work, we plan to investigate alternative solutions further or develop simpler versions.

Another limitation of our approach is related to an well-known problem of representing classes using color. While humans can differentiate well ten classes represented by colors (Ware, 2012), real-world datasets commonly have more classes. Thus, in order to prune deep learning models trained on more than ten classes, the interaction mechanism to visualize the filters’ patterns using the heatmap would need a different approach, such as selection boxes on the interface.

## 7 CONCLUSIONS

The design of deep learning architectures can be a tedious task. The most common approach is to define models that are way too complex for the problem in consideration and then fine-tune the architectures by removing filters or layers. Then, the models are re-trained with tuned architecture to achieve similar performance.

In this work, we propose a semi-automatic approach that uses the Bayesian Case Model (BCM) to identify the most important filters of convolutional layers based on the activation of the filters. Users can explore the dataset through scatterplot visualization while investigating the filters’ activations and their corresponding importance to the model’s prediction using coordination mechanisms. A preliminary use case shows that BCM can select the filters that truly contribute to the model’s performance. At the same time, the visualization emphasizes other aspects that contribute to the classification performance of datasets, such as redundancy among filters. After removing non-important filters, the prediction with finer CNN architectures yielded better results.

In future works, we plan to analyze more complex datasets and well-known CNN architectures to fully understand how much of these architectures could be removed while maintaining performance. Besides that, we plan to implement an entire pipeline where all the techniques involved in our methodology would be integrated.

## ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) and by Fundação de Amparo à Pesquisa (FAPESP) [grant numbers #2018/17881-3, #2018/25755-8].

## REFERENCES

- Clavien, G., Alberti, M., Pondenkandath, V., Ingold, R., and Liwicki, M. (2019). Dnnviz: Training evolution visualization for deep neural network. In *2019 6th Swiss Conference on Data Science (SDS)*, pages 19–24.
- Dubey, A., Chatterjee, M., and Ahuja, N. (2018). Coreset-based neural network compression.
- Garcia, R., Falcão, A. X., Telea, A. C., da Silva, B. C., Tørresen, J., and Dihl Comba, J. L. (2019). A methodology for neural network architectural tuning using activation occurrence maps. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.
- He, Y., Liu, P., Wang, Z., Hu, Z., and Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration.
- He, Y., Zhang, X., and Sun, J. (2017). Channel pruning for accelerating very deep neural networks.
- Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99*, pages 50–57, New York, NY, USA. ACM.
- Hohman, F., Park, H., Robinson, C., and Polo Chau, D. H. (2020). Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1096–1106.
- Kim, B., Rudin, C., and Shah, J. (2014). The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS' 14*, pages 1952–1960, Cambridge, MA, USA. MIT Press.
- LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.
- Li, G., Wang, J., Shen, H.-W., Chen, K., Shan, G., and Lu, Z. (2021). Cnnpruner: Pruning convolutional neural networks with visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1364–1373.
- Liu, S., Maljovec, D., Wang, B., Bremer, P., and Pascucci, V. (2017a). Visualizing high-dimensional data: Advances in the past decade. *IEEE Trans. Vis. Comput. Graph.*, 23(3):1249–1268.
- Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. (2017b). Learning efficient convolutional networks through network slimming.
- Luo, J.-H., Wu, J., and Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression.
- Marcilio-Jr, W. E., E. D. M. (2020). Sadire: a context-preserving sampling technique for dimensionality reduction visualizations. *J Vis*, 23:999–1013.
- Marcilio-Jr, W. E., Eler, D. M., Garcia, R. E., Correia, R. C. M., and Silva, L. F. (2020). A hybrid visualization approach to perform analysis of feature spaces. In Latifi, S., editor, *17th International Conference on Information Technology–New Generations (ITNG 2020)*, pages 241–247, Cham. Springer International Publishing.
- Marcílio-Jr, W. E., Eler, D. M., Paulovich, F. V., and Martins, R. M. (2021a). Humap: Hierarchical uniform manifold approximation and projection.
- Marcílio-Jr, W. E., Eler, D. M., Paulovich, F. V., Rodrigues-Jr, J. F., and Artero, A. O. (2021b). Explorer-tree: A focus+context exploration approach for 2d embeddings. *Big Data Research*, 25:100239.
- Pezzotti, N., Höllt, T., Van Gemert, J., Lelieveldt, B. P. F., Eisemann, E., and Vilanova, A. (2018). Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):98–108.
- Rauber, P. E., Fadel, S. G., Falcão, A. X., and Telea, A. C. (2017). Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):101–110.
- Reingold, E. M. and Tilford, J. S. (1981). Tidier drawings of trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228.
- Strobel, H., Gehrmann, S., Pfister, H., and Rush, A. M. (2018). Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676.
- Ware, C. (2012). *Information Visualization: Perception for Design*. Morgan Kaufmann Series in Interactive Technologies. Morgan Kaufmann, Amsterdam, 3 edition.
- Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., and Davis, L. S. (2018). Nisp: Pruning networks using neuron importance score propagation.
- Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.