The Thirty-Second International Florida
Artificial Intelligence Research Society Conference (FLAIRS-32)

On the Tree Structure of Deep Convolutional Sum-Product Networks

Cory J. Butz
butz@cs.uregina.ca
University of Regina
Canada

André L. Teixeira
teixeira@cs.uregina.ca
University of Regina
Canada

Jhonatan S. Oliveira
oliveira@cs.uregina.ca
University of Regina
Canada

André E. dos Santos
dossantos@cs.uregina.ca
University of Regina
Canada

Abstract

Deep convolutional sum-product networks (DCSPNs) have very recently been introduced and shown to yield state-of-the-art results in image completion tasks. A DCSPN consists of a tree structure (a directed acyclic graph) coupled with parameters of the structure. Given that DCSPNs are in their infancy, many open questions remain regarding the properties and topology of their tree structure.

In this paper, we undertake three investigations pertaining to the DCSPN structure. The first two studies revolve around the original structure put forth in the seminal paper. These studies increase the number of pooling layers and vary the hyperparameters in attempts to improve accuracy. The third inquiry suggests a new DCSPN tree structure that significantly lowers the training time at a modest expense of accuracy.

Introduction

Deep convolutional sum-product networks (DCSPNs) (Butz et al. 2019) are a subclass of *convolutional neural networks* (CNNs) (Goodfellow, Bengio, and Courville 2016) that define valid sum-product networks. This subclass is characterized by convolutional layer filters of certain sizes and non-overlapping windows in sum-pooling layers. The payoff is that DCSPNs are a rigorous probabilistic model allowing for exploitation of various forms of probabilistic reasoning, including *marginal* inference and *most probable explanation* (MPE) inference.

The DCSPN structure reported in (Butz et al. 2019) performed exceptionally well in image completion. Open questions then include whether variations of this structure are more effective, what the optimal hyperparameter values are, and whether a completely new structure can learn faster than the structure in (Butz et al. 2019).

We undertake two investigations pertaining to variations of the DCSPN structure in (Butz et al. 2019) and put forth a new DCSPN structure aimed at dramatically reducing the training time. One variation is to increase the number of pooling layers and also their window size in order to better capture local structure in image data. Whereas (Butz et al. 2019) used 2 pooling window sizes exclusively, here we study 3 up to 6 sizes. A second variation is conducting a grid search on the hyperparameters, but keeping the inaugural DCSPN structure. In these two investigations, we can achieve competitive results for left-completion in a benchmark dataset. Our third study proposes a new DCSPN structure. Our structure yields comparable *mean squared error* (MSE) scores compared to all competing methods except for (Butz et al. 2019). In addition, the training time for our proposed structure is significantly reduced. Thus, this new structure serves as a trade-off between accuracy and training time.

Varying the DCSPN Structure

We refer the reader to (Butz et al. 2019) for an introduction to DCSPNs. In the next two subsections, we respectively increase the number of pooling layers and vary some hyperparameters in attempts to improve accuracy.

Varying the Number of Pooling Layers

The horizontal and vertical windows in the pooling layers capture local structure in the image data. Here, we investigate the effect of varying the pooling layers.

In (Butz et al. 2019), two pooling layers follow each convolutional layer: one with a window size of 1x2 and the other 2x1. Alternate the window sizes of 1x2 and 2x1 with 2x2 and 2x2 every n layers. We call the former *fine* pooling layers and the latter *coarse* pooling layers.

Consider the fine pooling layers. We increase the number of fine pooling windows from 2 up to 3-6. In addition, the size of the additional windows ranges from 4 up to 8. Since increasing the number of fine pooling layers can give an unwieldy network, the overall size of the structure is compensated by using even coarser pooling layers, which serve as a regularization technique. We use either 2x2 and 4x4 or 4x4 and 4x4. Table 1 shows the exact combinations tried and the corresponding *mean squared error* (MSE) score.

Given that the number and size of pooling layers were increased, one may wonder whether the alternating hyperparameter should be adjusted accordingly. To this end, Table 1 reports the value of the alternating hyperparameter chosen in order to avoid a large number of layers.

The MSE scores in Table 1 beat all competing methods reported in (Butz et al. 2019), except for DCSPNs which scored 455. These encouraging results, exemplified by the

---

Table 1: Increasing the amount of fine pooling layers and using a larger coarse pooling to avoid a massive network.

MSE	Fine pooling	Coarse pooling	# Layers	# Layers between alternating
514	1x2,2x1,1x4	2x2,4x4	2139	100
600	1x2,2x1,1x4	4x4,4x4	1383	100
646	1x2,2x1,1x4,4x1	4x4,4x4	2941	100
604	1x2,2x1,1x4,4x1,1x8	4x4,4x4	1741	89
536	1x2,2x1,1x4,4x1,1x8,8x1	4x4,4x4	2306	101

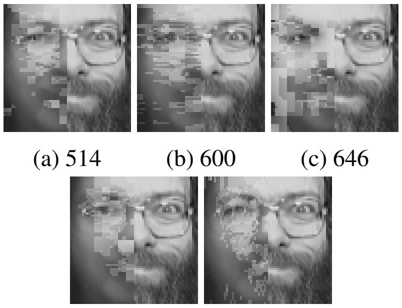

(a) 514 (b) 600 (c) 646
(d) 604 (e) 536

Figure 1: Left-completions with respect to Table 1.

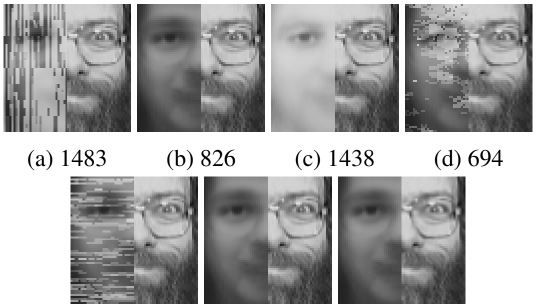

(a) 1483 (b) 826 (c) 1438 (d) 694
(e) 1062 (f) 1329 (g) 1689

Figure 2: Left-completions using the grid search in Table 2.

left-completions illustrated in Figure 1, suggest further research in this direction may be fruitful.

Varying Non-Structural Hyperparameters

Here, we conduct an in-depth investigation into the effect of three non-structural hyperparameters on the MSE scores.

We investigate the roles played by three non-structural hyperparameters. *Initializer*, *Inference Type*, and *Filter* are the focus of this section. These hyperparameters are non-structural in the sense that they are independent of the topological structure of the DCSPN. For instance, *Initializer* and *Filter* hyperparameters are related to the SPN weights, which are parameters of the SPN model and not the SPN structure.

Initializer can assume either *uniform* or *Glorot* and represents how network weights are initialized. A *uniform* initializer generates values between a defined lower bound and upper bound using a uniform distribution. The values chosen were 0 and 1.

On the other hand, a *Glorot* initializer (Glorot and Bengio 2010) samples from a uniform distribution using the follow bound:

$$\left[-\sqrt{\frac{6}{t_{in}+t_{out}}},+\sqrt{\frac{6}{t_{in}+t_{out}}}\right], \qquad (1)$$

where t_{in} and t_{out} denote the number of tensor inputs and output units, respectively.

Observe that the Glorot initializer may involve negative weights, which are not allowed by the definition of SPN sum node weights. Also, it is worth mentioning that during learning, weights can assume negative values. However, (Hsu, Kalra, and Poupart 2017) project negative weights to zero during inference in their implementation of sum nodes. We adopted this approach in our implementation. The weights projection approach works as a regularization technique.

Next, the hyperparameter *Inference Type* takes domain values *Hard* or *Soft* and stipulates which function is applied during the forward phase of MPE inference (Poon and Domingos 2011). *Hard* means a max function is utilized, while *Soft* indicates that a typical sum function is invoked.

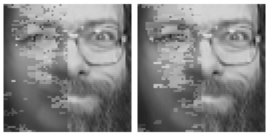
Finally, the hyperparameter *Filter* can assume domain values 1-by-1 or height-by-width (H-by-W). The Filter hyperparameter indicates the size of the filter for each convolutional layer.

Table 2 reports the MSE score of a grid search over these three hyperparameters. Figure 2 illustrates left-completions on an image taken from the Olivetti dataset.

Table 2: A grid search over 3 hyperparameters.

Initializer	Filter	Inference Type	MSE
uniform	H-by-W	Hard	1483
uniform	1-by-1	Soft	826
uniform	1-by-1	Hard	1438
Glorot	H-by-W	Soft	694
Glorot	H-by-W	Hard	1062
Glorot	1-by-1	Soft	1329
Glorot	1-by-1	Hard	1689

Analysis of the results in Table 2 and Figure 2 reveals a few trends. Clearly, *Inference Type* is an important hyperparameter. Instantiating to *Hard* yields very poor MSE scores. In contrast, when set to *Soft*, the MSE scores are competitive with other methods. Similarly, *Filter* tends to be another polarizing hyperparameter. When set to 1-by-1, the majority of MSE scores are poor. On the other hand, H-by-W can



(a) 694 (b) 570

Figure 3: Fine tuning hyperparameter values.

yield both poor scores as well as achieve state-of-the-art. The same can be said of the *Initializer* hyperparameter.

Subsequent fine-tuning could lead to further improvement of MSE scores. For instance, consider the score 694 in Table 2. Increasing the number of epochs from 300 to 500 during learning and increasing the number of convolutional layer channels from 12 to 16, reduced the MSE score from 694 down to 570 as shown in Figure 3.

The Filter hyperparameter significantly impacts the size of the network. The number of weights increases considerably when using filter size of H-by-W compared to 1-by-1. It appears that a larger number of parameters yields better MSE scores.

A New DCSPN Structure

In order to reduce the training time, we propose in this section a new DCSPN structure.

While Theorem 2 in (Butz et al. 2019) imposes constraints that must necessarily be satisfied by the DCSPN structure, there are a wide variety of such structures in theory. Here, we investigate 11 possible structures defined by the following pooling layer windows:

2x1,1x2
2x1,1x2,4x1
2x1,1x2,4x1,1x4
2x1,1x2,4x1,1x4,8x1
2x1,1x2,4x1,1x4,8x1,1x8
2x1,1x2,4x1,1x4,8x1,1x8,16x1
2x1,1x2,4x1,1x4,8x1,1x8,16x1,1x16
2x1,1x2,4x1,1x4,8x1,1x8,16x1,1x16,32x1
2x1,1x2,4x1,1x4,8x1,1x8,16x1,1x16,32x1,1x32
2x1,1x2,4x1,1x4,8x1,1x8,16x1,1x16,32x1,1x32,64x1
2x1,1x2,4x1,1x4,8x1,1x8,16x1,1x16,32x1,1x32,64x1,1x64.

Let $\mathbf{W}$ denote one of the above 11 sets of pooling windows choices. Let $\mathbf{S}$ be a set of convolutional layers to be processed. That is, convolutional layers on which pooling windows in $\mathbf{W}$ will be applied. Let $\mathbf{G}$ be a set of pooling layers sets.

Initialization is first conducted. A convolutional layer follows the representational layer and is inserted in the set $\mathbf{S}$. Each time a convolutional layer is removed from $\mathbf{S}$, all possible pooling windows in $\mathbf{W}$ are applied on it. Thus, every convolutional layer will be followed by pooling layers using $\mathbf{W}$. Group together all pooling layers having the same height and width. For each group, perform a channel (depth) augmentation by applying a convolutional layer and then add the convolutional layer to $\mathbf{S}$.

Repeat the above process until a convolutional layer of unitary height and width is reached. This convolutional layer is the root of the DCSPN structure.

Algorithm 1 formalizes the structure construction process.

Algorithm 1 Building the new DCSPN Structure

```
Input: a DCSPN representational layer $R$ and a set of pooling windows $\mathbf{W}$
Output: a DCSPN structure $\mathcal{D}$
Main:
  ▷ Initialization
  $\mathbf{S} = \emptyset$        ▷ Set of convolutional layers to be processed
  $\mathbf{G} = \emptyset$        ▷ Set of pooling layers groups
  New convolutional layer $S$
  ▷ A convolutional layer follows the representation layer
  Add edge $(S, R)$ in $\mathcal{D}$
  $\mathbf{S} = \mathbf{S} \cup \{S\}$        ▷ and is added to S
  while $|\mathbf{G}|$ is not 1 do
      $\mathbf{P} = \emptyset$        ▷ Set of pooling layers to be grouped
      while S is not empty do
          $S$ = choose and remove $S$ from $\mathbf{S}$
          ▷ Applying pooling windows
          for $w$ in W do
              New pooling layer $P$ using $w$
              ▷ Convolutional layer is followed by generated pooling layers
              Add edge $(P, S)$ in $\mathcal{D}$        ▷ add edge
              $\mathbf{P} = \mathbf{P} \cup \{P\}$        ▷ and insert into P
      ▷ Group same size pooling layers
      $\mathbf{G} = groupSameSize(\mathbf{P})$
      for group in G do
          New layer $S$
          ▷ Channel augmentation
          for $P$ in group do
              Add edge $(S, P)$ in $\mathcal{D}$        ▷ add edge
          ▷ Add convolutional layer to be processed
          $\mathbf{S} = \mathbf{S} \cup \{S\}$
  return $\mathcal{D}$        ▷ DCSPN structure
```

Table 3 gives structural information of the DCSPN constructed by Algorithm 1 for the 11 choices of pooling windows. Figure 4 gives sample left-completion in Olivetti when using the structure built by Algorithm 1.

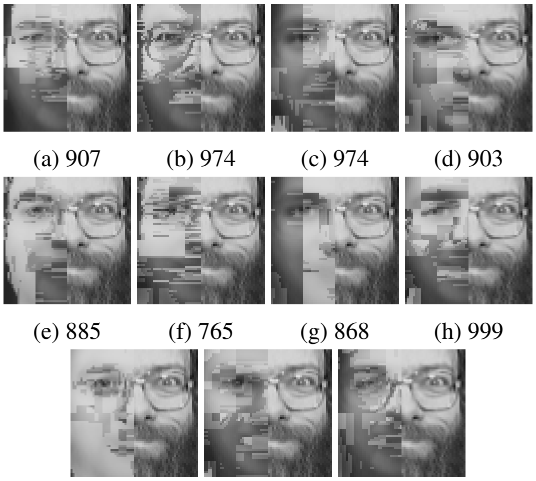
Figure 5 shows that the learning time ranged from 26-34 minutes when using the new proposed structure built by Algorithm 1, which is significantly lower than the roughly 2 hours required for the structure in (Butz et al. 2019). Figure 6 gives the TensorFlow model size of the structure built by Algorithm 1 versus the structure in (Butz et al. 2019).

Conclusion

DCSPNs (Butz et al. 2019) formally establish when CNNs define valid SPNs. A DCSPN consists of convolutional and sum-pooling layers together with network parameters. Here, we have conducted a comprehensive investigation into the DCSPN structure. Our first main result is that increasing the

Table 3: Structural information with respect to the above 11 pooling windows.

MSE	# Layers	ModelSize(MB)	Learning Time(min)
907	344	159	32
974	337	158	32
974	330	158	32
903	316	156	34
885	302	154	34
765	281	150	32
868	260	146	31
999	232	137	31
796	204	128	30
843	169	110	26
784	134	92	26


(a) 907 (b) 974 (c) 974 (d) 903

(e) 885 (f) 765 (g) 868 (h) 999

(i) 796 (j) 843 (k) 784

Figure 4: Completion examples from the 11 reported results of the compact structure.

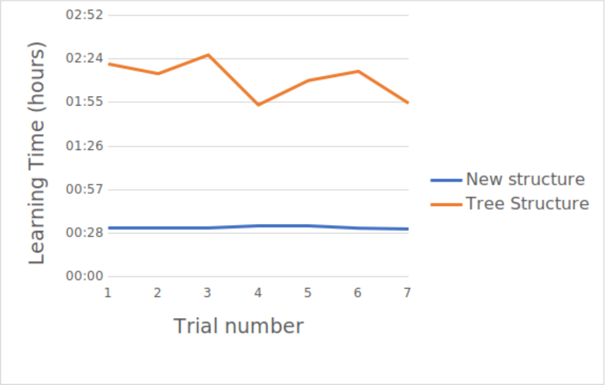


Figure 5: Algorithm 1 dramatically lowers DCSPN training time.

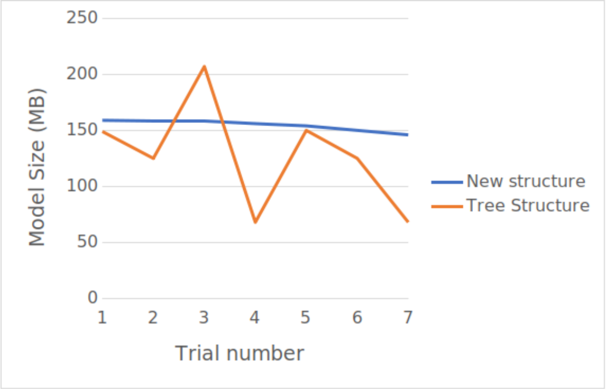


Figure 6: Algorithm 1 has a stable DCSPN size.

number of pooling layers does not capture more local structure in the image data. This suggests that the small windows used in (Butz et al. 2019) of 1-by-2 and 2-by-1 sufficiently capture local structure, and that increasing the number of pooling windows does not improve accuracy.

Our second main result is that comparable results can be obtained using the DCSPN structure detailed in (Butz et al. 2019), but with different hyperparameter values. Our third main contribution is a novel structure to replace the one suggested in (Butz et al. 2019). Here, the focus in not on a lower MSE score, but instead a structure that allows learning significantly faster while maintaining competitive MSE scores. The structure can indeed be trained significantly faster at a small increase in MSE scores, as shown in Table 3.

References

Butz, C.; Oliveira, J.; dos Santos, A.; and Teixeira, A. L. 2019. Deep convolutional sum-product networks. In *Thirty-third Conference on Artificial Intelligence (AAAI 2019)*.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, 249–256.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press.

Hsu, W.; Kalra, A.; and Poupart, P. 2017. Online structure learning for sum-product networks with gaussian leaves. *arXiv preprint arXiv:1701.05265*.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, 337–346.