GSPNN: Graph Shortest Path Neural Network

Atharv Navale
Bengaluru, India

September 16, 2025

Abstract

A neural architecture termed *GSPNN* (Graph Shortest Path Neural Network) is introduced in which both inference and learning are carried out without backpropagation. Classification is posed as a shortest-path problem over a layered directed acyclic graph (DAG). Each layer defines a local softmax distribution over outgoing edges; per-edge cost is the (temperature-scaled) negative log-probability. Inference reduces to a Viterbi (min-sum) dynamic program over the graph. Learning proceeds by local, forward-only updates: for each training example, the shortest path to the true class is compared with the best competing class. If a margin is violated, normalized per-node updates are applied that increase the probability of chosen (true) edges and decrease it for competing (wrong) edges. The updates require only forward signals (local probabilities and keys) and avoid gradients and backpropagation. On MNIST with PCA features, a compact configuration attains 94–96% test accuracy with sub-second epoch times on a single Colab GPU due to fully vectorized updates and optional precomputation of per-layer keys. Algorithmic details, computational complexity, ablations (temperature and margin schedules, top-k negatives, EMA), limitations, and connections to Viterbi decoding, structured prediction, and local learning rules are discussed.

1 Introduction

Backpropagation is the de facto standard for training deep networks; however, numerous alternatives with improved locality or biological plausibility have been explored [1, 2, 3, 4, 5, 6, 7]. In this work, a forward-only alternative is investigated in which classification is cast as finding a minimum-cost path through a layered DAG whose edge costs are derived from local softmax distributions. Inference follows a standard Viterbi min-sum dynamic program [1, 2]. Learning relies on local, normalized adjustments of edge scores along two forward-computed paths (true vs. best competitor), reminiscent of margin-based structured prediction [5, 4] but without backpropagated gradients.

Contributions.

1. A forward-only training rule for layered graph classifiers using local softmax costs and Viterbi inference is formalized.
2. An efficient vectorized implementation with optional precomputed keys is provided, enabling sub-second training epochs on MNIST.
3. Empirical validation is presented along with analyses of temperature, margin, top-k negatives, and EMA smoothing.

2 Model

Let $x \in \mathbb{R}^d$ denote an input (optionally PCA-preprocessed). The graph contains L internal layers and a terminal class layer of size C. Layer ℓ has n_ℓ nodes; the previous layer has $n_{\ell-1}$ nodes.

Each layer ℓ computes a key vector

$$K^{(\ell)}(x) \triangleq \frac{P_\ell x}{\|P_\ell x\|_2 + \varepsilon} \in \mathbb{R}^k, \quad \varepsilon > 0, \quad P_\ell \in \mathbb{R}^{k\times d}. \tag{1}$$

For each edge $(i \to j)$ from node i in layer $\ell - 1$ to node j in layer ℓ, a logit is defined by

$$s_{i,j}^{(\ell)}(x) \triangleq \langle W_{i,j}^{(\ell)}, K^{(\ell)}(x)\rangle + B_{i,j}^{(\ell)}, \quad W_{i,j}^{(\ell)} \in \mathbb{R}^k,\ B_{i,j}^{(\ell)} \in \mathbb{R}. \tag{2}$$

Per-node probabilities and costs are obtained through a temperature softmax:

$$p_{i,\cdot}^{(\ell)}(x) \triangleq \operatorname{softmax}\big(s_{i,\cdot}^{(\ell)}(x)/\tau\big), \qquad c_{i,j}^{(\ell)}(x) \triangleq -\tau \log p_{i,j}^{(\ell)}(x). \tag{3}$$

3 Inference (Viterbi)

Define dynamic-programming tables $D_{\ell,j}$ as the minimum path cost to reach node j in layer ℓ and $bp_{\ell,j}$ as the best predecessor index. Initialization uses $D_{0,1} = 0$ at the unique start node and $D_{0,i>1} = +\infty$. For each layer $\ell = 1, \dots, L$,

$$D_{\ell,j} = \min_{i\in\{1,\dots,n_{\ell-1}\}} \big[D_{\ell-1,i} + c_{i,j}^{(\ell)}(x)\big], \qquad bp_{\ell,j} = \arg\min_i \big[D_{\ell-1,i} + c_{i,j}^{(\ell)}(x)\big]. \tag{4}$$

At the class layer ($\ell = L$), the prediction is

$$\hat{y}(x) = \arg\min_{j\in\{1,\dots,C\}} D_{L,j}. \tag{5}$$

4 Forward-Only Learning

For a labeled sample (x, y), quantities D and the backpointers are computed once (a forward pass). Let $\pi(y)$ denote the backtracked path ending at class y, and $\pi(\bar{y})$ the path for the best competing class $\bar{y} = \arg\min_{j\neq y} D_{L,j}$. Given a margin $m > 0$, no update is made if $D_{L,y} + m \le D_{L,\bar{y}}$. Otherwise, local, normalized updates are applied at each visited node along both paths.

For a node i with outgoing distribution $p_{i,\cdot}$ and chosen edge j on a given path, the logit increments are

$$\Delta s_{i,k} = +\eta\,\mathbf{1}[k = j] - \eta\, p_{i,k} \qquad \text{(true-path visit)}, \tag{6}$$
$$\Delta s_{i,k} = -\eta\,\mathbf{1}[k = j] + \eta\, p_{i,k} \qquad \text{(wrong-path visit)}, \tag{7}$$

with learning rate η. Parameters are then updated via

$$W_{i,k} \leftarrow W_{i,k} + \Delta s_{i,k}\, K(x), \qquad B_{i,k} \leftarrow B_{i,k} + \Delta s_{i,k}, \tag{8}$$

optionally followed by weight decay and parameter clipping. No gradients are backpropagated; only forward-computed probabilities p, keys K, and backpointers are used.

Top-k negatives and schedules. To sharpen class separation, updates can include the top-k competing classes (small k, e.g., 3). A temperature schedule (e.g., $\tau : 0.9 \to 0.5$) and a margin schedule (e.g., $0.6 \to 0.4$) improve stability.

5 Algorithms

Algorithm 1 Viterbi Inference (Min-Sum DP)

```
Require: keys $\{K^{(\ell)}(x)\}_{\ell=1}^{L}$, parameters $\{W^{(\ell)}, B^{(\ell)}\}$, temperature $\tau$
 1: Initialize $D_{0,1} \leftarrow 0$, $D_{0,i>1} \leftarrow +\infty$
 2: for $\ell = 1$ to $L$ do
 3:     $z \leftarrow \langle W^{(\ell)}, K^{(\ell)}(x)\rangle + B^{(\ell)}$          (batched inner products)
 4:     $\log p \leftarrow \log\mathrm{softmax}(z/\tau,\ \text{dim=outgoing})$
 5:     $C \leftarrow -\tau \cdot \log p$
 6:     for each target node $j$ in layer $\ell$ do
 7:         $D_{\ell,j}, bp_{\ell,j} \leftarrow \min_i \left(D_{\ell-1,i} + C_{i,j}\right)$
 8:     end for
 9: end for
10: return $D_{L,\cdot}$ and backpointers $\{bp_{\ell,\cdot}\}$
```

Algorithm 2 Forward-Only Update (per batch, vectorized)

```
Require: batch indices $B$, labels $y$, precomputed keys $\{K^{(\ell)}(x_b)\}$, margin $m$, learning rate $\eta$,
    top-$k$ negatives $K$
 1: Run Viterbi to obtain $D$ and backpointers for all $b \in B$
 2: For each $b$, compute best competitor $\bar{y}_b$ and violation mask $\mathbf{1}[D_{y_b} + m > D_{\bar{y}_b}]$
 3: For violating samples, backtrack true paths $\pi(y_b)$ and wrong paths $\pi(\bar{y}_b^{(t)})$ for $t = 1..K$
 4: for each layer $\ell$ do
 5:     For all visited source nodes $i$ (from both true and wrong paths), build $\Delta s_{i,\cdot}$ rows via
        $(I - p)$ or $(p - I)$
 6:     Apply $\Delta B_{i,\cdot} \leftarrow \eta\,\Delta s_{i,\cdot}$ and $\Delta W_{i,\cdot,:} \leftarrow \eta\,\Delta s_{i,\cdot} \otimes K^{(\ell)}(x)$ using indexed additions
 7: end for
```

6 Complexity and Implementation Notes

Let k be the key dimension and $E_\ell = n_{\ell-1}n_\ell$ the number of edges per layer. The dominant forward cost per batch is the batched inner product $O(|B|\sum_\ell E_\ell k)$, plus a softmax and a minimum over sources per target. Python loops can be eliminated by vectorizing operations and accumulating updates with indexed additions. Precomputing $K^{(\ell)}(x)$ for all samples removes repeated $xP_\ell^\top$ multiplications (feasible at MNIST scale), further reducing wall-clock time.

7 Experiments

Evaluation is conducted on MNIST with PCA features (StandardScaler → PCA with whitening). Unless otherwise noted, the following configuration is used: `pca_dim` = 128, $k = 256$, layer sizes $(128, 128)$, batch size 1024, $\eta = 10^{-3}$ with 0.98 epoch decay, weight decay $5 \cdot 10^{-6}$, temperature annealed from 0.9 to 0.5, margin $0.6 \to 0.4$, top-$k = 3$, and EMA (0.999) for evaluation only. On a single Colab GPU, epochs complete in approximately 1–1.5 seconds. Test accuracy reaches 95–96% within 10–20 epochs. Training and evaluation remain stable under small hyperparameter variations.

8 Related Work

Viterbi and DP. The inference procedure is classical Viterbi min-sum DP [1]; see [2] for an HMM tutorial. **Structured prediction.** Local-normalized costs connect to CRFs [3], and the margin-based competition is akin to large-margin structured methods [5] and the structured perceptron [4]. **Forward/local learning.** Forward-only updates using local activations and probabilities resonate with alternatives to backpropagation (e.g., feedback alignment [6]) and are discussed in standard texts [7].

9 Limitations and Future Directions

GSPNN does not provide end-to-end credit assignment in the backpropagation sense; training relies on local competition and may saturate without richer global feedback. Extensions to deeper/larger graphs, sequence tasks, or combinations with differentiable relaxations (e.g., entropy-regularized shortest paths) appear promising. Theoretical analysis of convergence and generalization under local-normalized updates remains an open problem.

10 Conclusion

A forward-only graph softmax path network trained exclusively from forward signals has been presented. Despite its simplicity, competitive accuracy on MNIST is achieved with excellent training throughput, suggesting that forward-only learning with structured inference is a viable direction for efficient classification.

Reproducibility. Code, default hyperparameters, and a vectorized implementation with optional key precomputation can be provided upon request.

Appendix A: Default Hyperparameters

Parameter	Value
PCA dimension	128
Key dimension k	256
Layer sizes	$(128, 128)$
Batch size	1024
Learning rate	10^{-3} (decay 0.98/epoch)
Weight decay	$5 \cdot 10^{-6}$
Temperature τ	$0.9 \to 0.5$ (anneal)
Margin	$0.6 \to 0.4$
Top-k negatives	3
EMA	0.999 (eval only)

References

[1] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[2] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[3] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, 2001.

[4] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, 2002.

[5] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[6] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, 2016.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.