C. Graber, O. Meshi, A. Schwing. Deep structured prediction with nonlinear output transformations. In *Advances in Neural Information Processing Systems*, 2018, pp. 6320-6331.

UVA CS 6316: Machine Learning : 2019 Fall
Course Project: Deep2Reproduce @
https://github.com/qiyanjun/deep2reproduce/tree/master/2019Fall

Deep Structured Prediction with Nonlinear Output Transformations

Reproduced By:

Guangtao Zheng, Hanjie Chen, Sanxing Chen, Wenbo Pang

(gz5hp, hc9mx, sc3hn, wp6ju)

Dec 5, 2019

Motivation - structured prediction

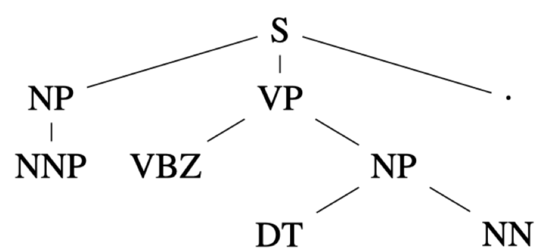
$$f : \mathcal{X} \to \mathcal{Y}.$$

Y is a scalar (Regression) $f : \mathcal{X} \to \mathbb{R}.$

Y is a class (Classification)

Y is ....

John has a dog . $\rightarrow$

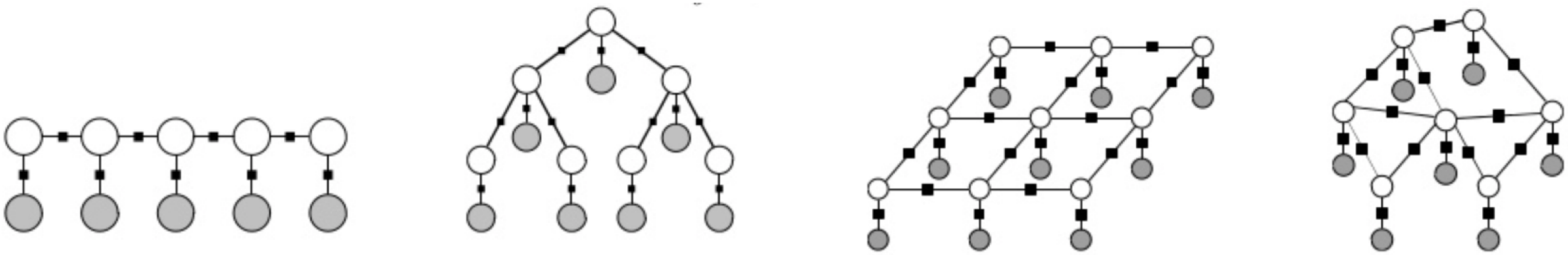


John has a dog . $\rightarrow$ (S (NP NNP $)_{NP}$ (VP VBZ (NP DT NN $)_{NP}$ $)_{VP}$. $)_S$

Vinyals, et.al., 2015

Motivation

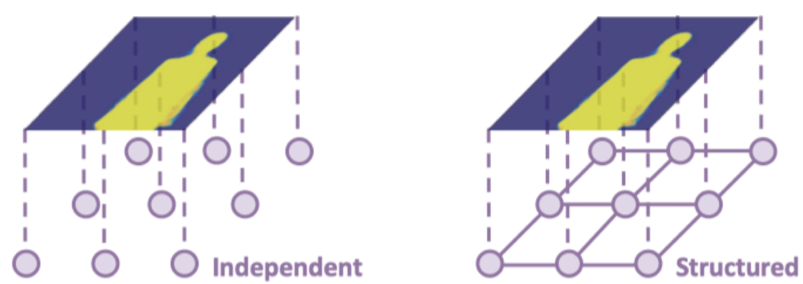
- Introducing structural assumptions into neural network(NN) can make learning more easier. [C. Ciliberto et al., 2018]
- Current methods add structure models ***on top of NN*** can only capture simple type of interactions. [J. Tompson el al., 2014; L.-C. Chen et al., 2015]
- Current methods include structured prediction ***inside NN*** (e.g. SPENs) are hard to optimize. [D. Belanger and A. McCallum 2016]
- Finding a method which is able to incorporate structure assumptions of data while retaining effectiveness.



Credit prof. Lampert's slides

Background

- ❖ A lot of tasks require predicting structured output rather than a simple scalar or classes
  - ➢ Generating natural language or image, sequential tagging, semantic segmentation, etc.
- ❖ A lot of methods have been proposed to address this problem
  - ➢ Structured SVM and Structured Logistic Regression (i.e. CRF)
- ❖ Deep NNs have been favoured in recent year, after achieving great results on various tasks
- ❖ Some efforts have been putting in addressing structure prediction problem for NNs
  - ➢ Show great promises of this direction
  - ➢ But can only model superficial interactions between output variables or hard to optimize



Related Work

- Structured Prediction *in the old days*
  - Augment linear classifiers (e.g. SVM, LR) [T. Finley et al., 2008; J. Lafferty et al., 2001]
- Structured Prediction *in the era of NN*
  - NN has a lot of potential to model structures [A. Krizhevsky et al., 2012]
  - Autoregressive Models
    - Recurrent neural network to model the structure of sequential output
    - Based on the ability of the neural net to model the conditional distribution
  - Structured Prediction Energy Networks (SPENs) [Belanger and McCallum, 2016]
    - Automatically learning the structure of deep nets leads to improved results
    - Optimization of the proposed approach remains challenging due to the non-convexity of NNs.
  - Deep value networks [Gygli et al., 2017]
    - Using training objective inspired by value based reinforcement learning

Claim / Target Task

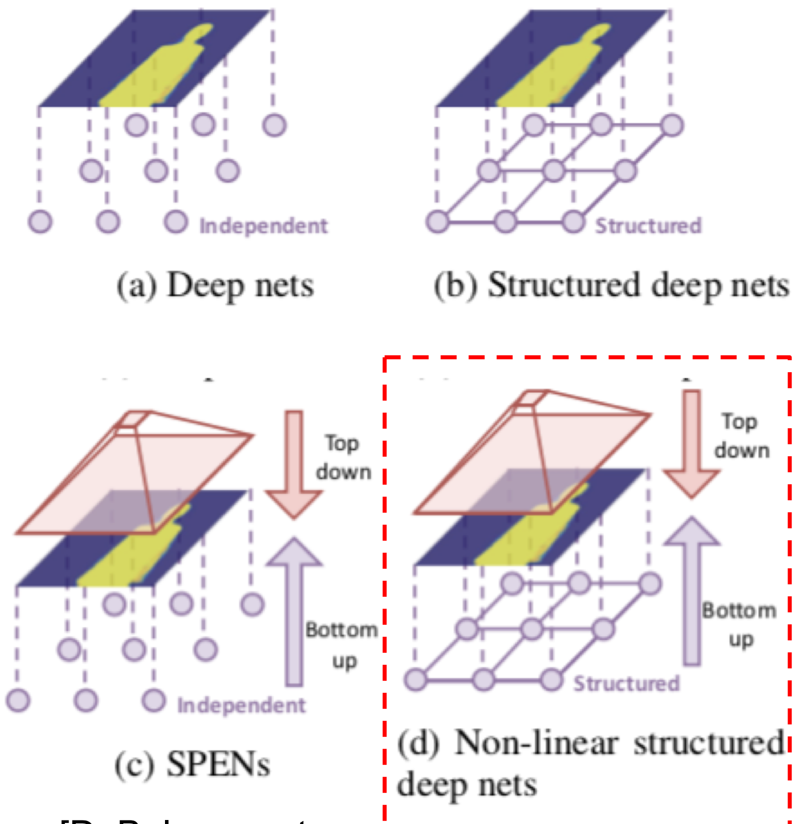
- ❖ We represent output variables as an intermediate structured layer in the middle of the neural architecture.

Capture nonlinear interactions between output variables.

- ❖ We discuss a rigorous formulation for structure inside deep nets using a Lagrangian framework.

Address the optimization problem.

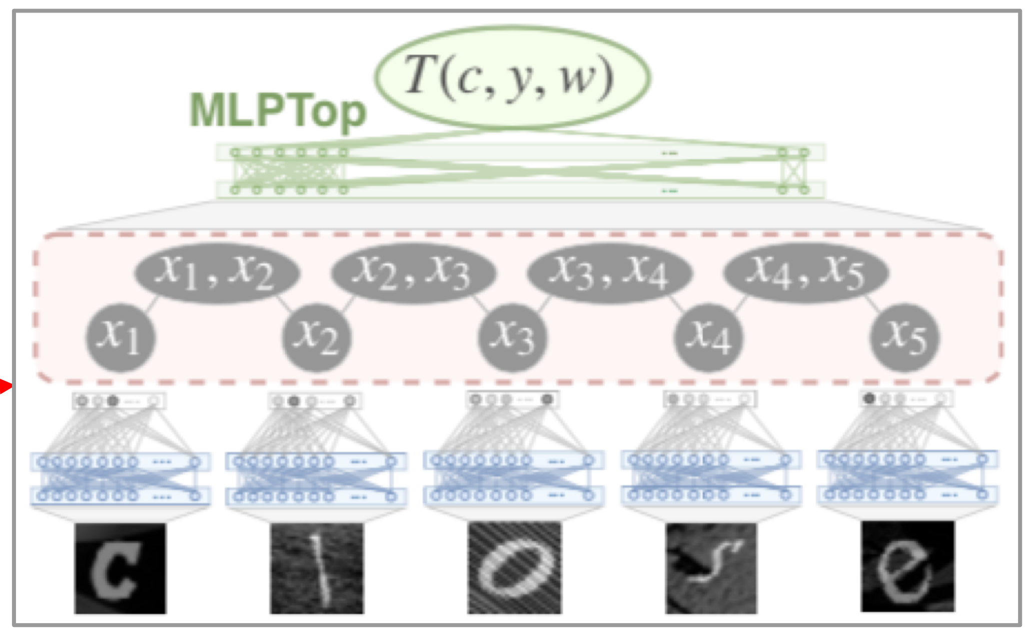
An Intuitive Figure Showing WHY Claim



(a) Deep nets (b) Structured deep nets

(c) SPENs (d) Non-linear structured deep nets

[D. Belanger et al., 2016]



Problem

- $x = (x_1, \dots, x_K)$ denote the multi-variate output space with $x_k \in \mathcal{X}_k, k \in \{1, \dots, K\}$ indicating a single variable.

$$x^* = \arg\max_{x \in \mathcal{X}} F(x, c, w)$$

- Classical deep networks: $F(x, c, w) = \sum_{k=1}^{K} f_k(x_k, c, w)$

Ignore correlations between pair of variables

- Structured deep networks: $F(x, c, w) = \sum_{r \in \mathcal{R}} f_r(x_r, c, w)$

NP-hard, low-order locality [S. E. Shimony, 1994]

Proposed Solution

❖ Non-linear Structured Deep Networks

$$F(x, c, w) = T(c, H(x, c, w), w)$$

H is a vector where each entry represents the score $f_k(x_k, c, w)$

- **Inference**

$$\max_{x \in \mathcal{X}, y} T(c, y, w) \quad s.t. \quad y = H(x, c, w)$$

Introduce Lagrange multipliers λ,

$$\min_{\lambda} \left(\max_{y} \{T(c, y, w) - \lambda^T y\} + \max_{x \in \mathcal{X}} \lambda^T H(x, c, w) \right)$$

Simplify the discrete optimization problem,

$$\min_{\mu} \left(\min_{\lambda} \left(\max_{y} \{T(c, y, w) - \lambda^T y\} + H^D(\mu, c, \lambda, w) \right) \right)$$

$H^D(\mu, c, \lambda, w)$ is the relaxed dual objective.

Proposed Solution

- **Learning**

$$\min_{w} \sum_{(x,c)\in\mathcal{D}} \underbrace{\max_{\hat{x}\in\mathcal{X}}\{F(\hat{x},c,w) + L(x,\hat{x})\}}_{\text{Loss augmented inference}} - F(x,c,w)$$

$$\Rightarrow \min_{w} \frac{C}{2}\|w\|_2^2 + \sum_{(x,c)\in\mathcal{D}} \left(\underbrace{\max_{\hat{x}\in\mathcal{X}}\{T(c,H(\hat{x},c,w),w) + L(x,\hat{x})\}}_{\text{Loss augmented inference}} - T(c,H(x,c,w),w)\right)$$

Implementation

❖ Inference Procedure

Algorithm 1 Inference Procedure

Input: Learning rates α_y, α_λ; y_0; λ_0; number of iterations n
$\mu^* \Leftarrow \operatorname{argmin}_{\hat{\mu}} H^D(\hat{\mu}, c, \lambda, w)$
$\bar{\lambda} \Leftarrow \lambda_0$
$y_1 \Leftarrow y_0$
for $i = 1$ **to** n **do**
  repeat
    $y_i \Leftarrow \frac{1}{\alpha_y}\left(y_i - y_{i-1} + \alpha_y \bar{\lambda}\right) - \nabla_y T(c, y, w)$
  until convergence
  $\lambda_i \Leftarrow \lambda_{i-1} - \alpha_\lambda \left(\nabla_\lambda H^D(\mu^*, c, \lambda, w) - y_i\right)$
  $\bar{\lambda} = 2\lambda_i - \lambda_{i-1}$; $y_{i+1} \Leftarrow y_i$
end for
$\boxed{\lambda \Leftarrow \frac{2}{n} \sum_{i=n/2}^{n} \lambda_i \; ; \; y \Leftarrow \frac{2}{n} \sum_{i=n/2}^{n} y_i}$
$\mu \Leftarrow \operatorname{argmin}_{\hat{\mu}} H^D(\hat{\mu}, c, \bar{\lambda}, w)$
Return: μ, $\bar{\lambda}$, y

better convergence in practice by averaging over the last n/2 iterates of y and $\boldsymbol{\lambda}$

Implementation

❖ Learning Procedure

Algorithm 2 Weight Update Procedure

Input: Learning rate α, $\hat{y}$, $\hat{\lambda}$, and $\mathcal{D}$
for $i = 1$ **to** n **do**
  $g = 0$
  for every datapoint in a minibatch **do**
    $\hat{x} \Leftarrow$ Inference in Algorithm 1 (adding $L(x, \hat{x})$)
    $g \Leftarrow g + \nabla_w \left(T(c, H(\hat{x}, c, w), w) - T(c, H(x, c, w), w)\right)$
  end for
  $w \Leftarrow w - \alpha\,(Cw + g)$
end for

- A minibatch of data at every iteration
- Every round of inference is followed by an update of the weights of the model, which is accomplished via gradient descent

Data Summary

Exp1: Word Recognition

- A synthetic word recognition dataset
- Constructed by taking a list of 50 common five-letter English words and rendering each letter as a 28x28 pixel image.
- Select a random image of each letter from the Chars74K dataset [Campos et al., 2009], randomly rotate, shift, and scale them, and then insert them into random background patches with high intensity variance
- **Task**: identify each word from the five letter images
- The training, validation, and test sets for these experiments consist of 1,000, 200, and 200 words

close

other

world

(a) Word recognition datapoints

Data Summary

Exp2: Multilabel classification

- Binary feature vectors (#1836)
- 159 possible labels
- 500 pairs chosen for structured models

dataset	Bibtex
binary feature vectors # possible labels	159
# pairs of most frequent label pairs	500

Exp3: Image tagging

- MIRFLICKR25k dataset [Huiskes et al., 2008]
- 25,000 images taken from Flickr
- Each assigned some subset of a possible 24 tags
- train/development/test sets: 10,000/5,000/10,000 images

Data Summary

Exp4: Semantic segmentation

- Weizmann Horses database [Borenstein et al., 2002]
- 328 images of horses paired with segmentation masks
- train/validation/test : 196/66/66 images
- scale the input images: 224x224 pixels for image, 64x64 pixels for masks

(b) Segmentation datapoints

Experimental Results

- Tasks
  - Word recognition
  - Multilabel classification
  - Image tagging
  - Semantic segmentation
- Models
  - **Unary**: a deep network model containing containing *unary* potentials
  - **DeepStruct**: a deep structured model containing *pairwise* potentials
  - **LinearTop**: a structured deep model with *linear* output transformations
  - **NLTop**: a structured deep model with *nonlinear* output transformations

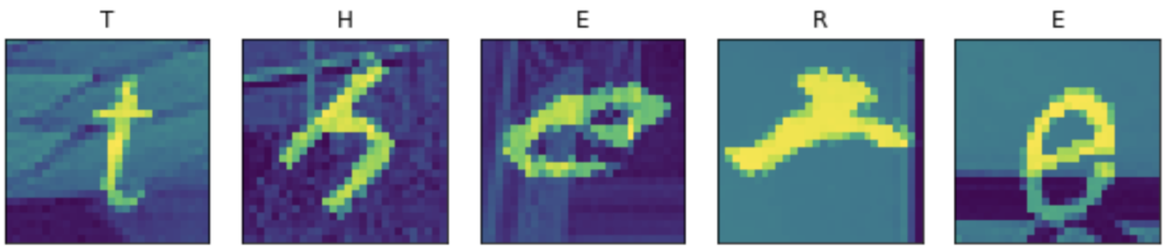
Experimental Results

- ❖ Word Recognition
  - ➢ Identify English words from images of letters
  - ➢ Two different graphs in structured models
    - ■ Chain: adjacent letters are connected
    - ■ Second-order: connecting letters two positions away
  - ➢ Measures
    - ■ Character accuracy
    - ■ Word accuracy: count the accuracy every five words

Table 1: Results for word recognition experiments. The two numbers per entry represent the word and character accuracies, respectively.

	Chain				Second-order			
	Train		Test		Train		Test	
Unary	0.003	0.2946	0.000	0.2350				
DeepStruct	0.077	0.4548	0.040	0.3460	0.084	0.4528	0.030	0.3220
LinearTop	0.137	0.5308	**0.085**	0.4030	0.164	0.5386	0.090	0.4090
NLTop	**0.156**	**0.5464**	0.075	**0.4150**	**0.242**	**0.5828**	**0.140**	**0.4420**

Reproduced Results




```
Unary:['S', 'S', 'E', 'R', 'E']
DeepStruct (Chain):['S', 'S', 'E', 'R', 'E']
DeepStruct (Second-Order):['S', 'S', 'E', 'E', 'E']
LTOP (Chain):['T', 'H', 'E', 'R', 'E']
LTOP (Second-Order):['T', 'H', 'R', 'E', 'E']
NLTOP (Chain):['T', 'H', 'E', 'R', 'E']
```

Chain: (T,H), (H,E), (E,R), (R,E)
Second-Order: (T,E), (H,R), (E,E)

word accuracy

Character accuracy

	Chain				Second-Order			
	Train		Test		Train		Test	
Unary	0	0.4626	0	0.2970	N/A	N/A	N/A	N/A
DeepStruct	0.2710	0.6760	0.0700	0.3990	0.2580	0.6628	0.0650	0.3750
LinearTop	0.2810	0.6980	0.1100	0.4770	0.3000	0.6940	0.0750	0.4340
NLTop	0.2370	0.6342	0.0900	0.4370	0.2820	0.6432	0.115	0.4300

Experimental Results

- Multilabel Classification
  - Dataset: Bibtex
  - Most frequent label pairs are chosen for the structured models
    - 500 pairs for Bibtex
  - Use macro-averaged F1 scores as the measure

Models	Bibtex
Unary	44.0
DeepStruct & NLStruct	Comparably
SPEN	42.4

Reproduced Results

- Multilabel Classification
  - Dataset: Bibtex, #Train: 4836   #Test: 2515
  - Loss: Cross Entropy
  - Unary model:

    two-layer perceptrons with ReLU nonlinearities and 318 units
  - Deepstruct:

    constrain pairwise potentials so that $W_{0,0} = W_{1,1}$ and $W_{0,1} = W_{1,0}$
  - Experiment results

	Precision	Recall	F1
Unary	0.4066	0.4800	0.4403
DeepStruct	0.4208	0.5184	0.4645

Experimental Results

❖ Image Tagging

➢ Fully connected pairwise graph is used as the structure

■ binary node -> a label

■ edge -> connecting labels

➢ SPENInf: SPEN-like inference procedure

➢ DeepStruct++

■ Add 2-layer perceptrons to DeepStruct

■ 1.8 times more parameters than NLTop

$$\mathrm{HammingLoss} = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathrm{XOR}(Y_{i,j}, P_{i,j})}{L}$$

Table 2: Results for image tagging experiments. All values are hamming losses.

	Train	Validation	Test
Unary	1.670	2.176	2.209
DeepStruct	1.135	2.045	2.045
DeepStruct++	1.139	2.003	2.057
SPENInf	1.121	2.016	2.061
NLTop	**1.111**	**1.976**	**2.038**

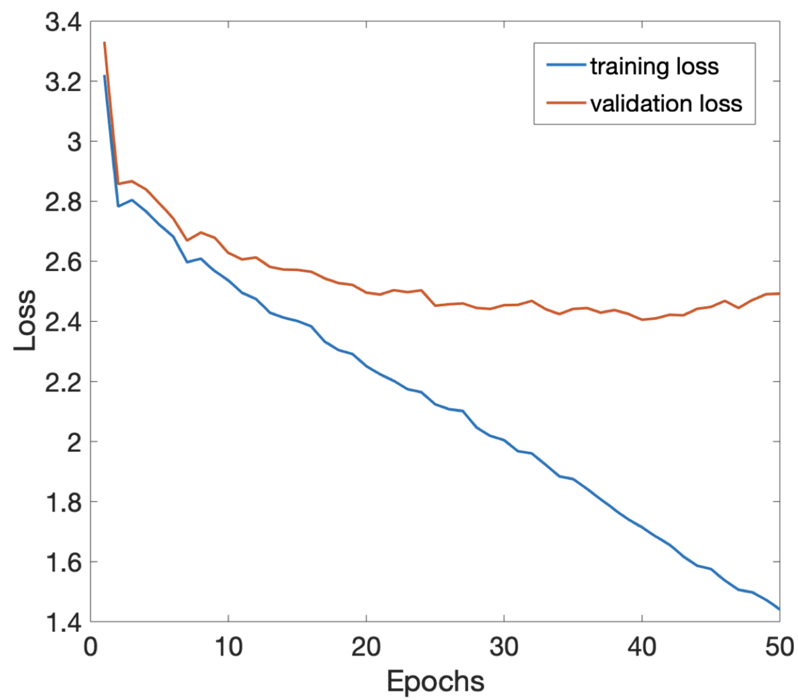
Reproduced Results

❖ Image Tagging

➢ Unary classifier: pre-trained Alexnet model provided by PyTorch

- Train loss: 1.7146, Val loss: 2.5042
- Test loss: 2.9804

Hyperparameters	
Optimizer	SGD
Learning rate	1e-4
Batch size	100
Epochs	50



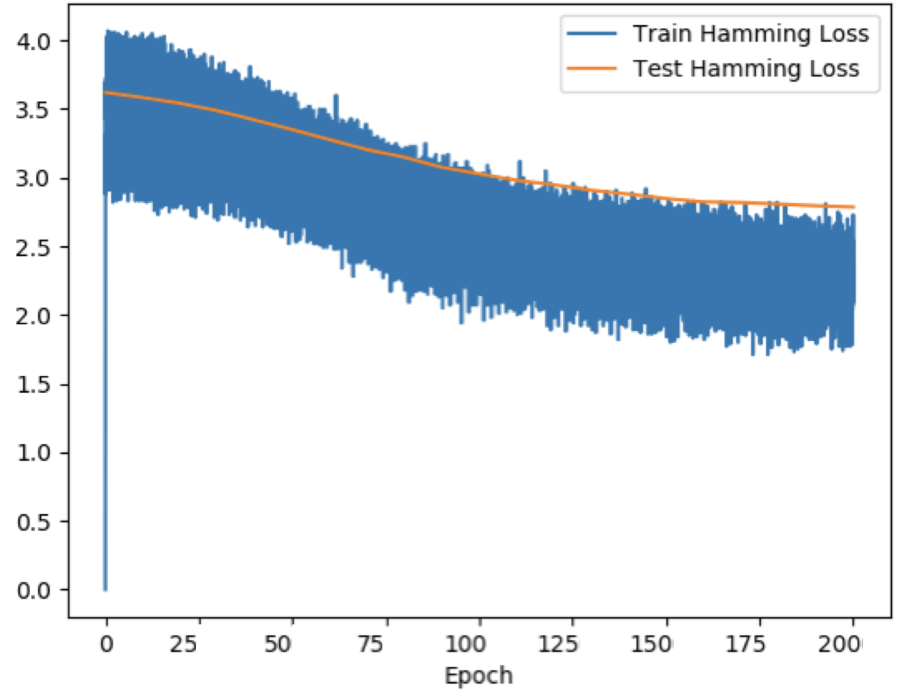
Reproduced Results

❖ Image Tagging

➢ DeepStruct

Hyperparameters	
Optimizer	SGD
Learning rate	5e-5
Batch size	100
Epochs	200
Hidden size	318
activation	hardtanh

- Train loss: 2.0368, Val loss: 2.7134
- Test loss: 2.7762



Reproduced Results

➢ DeepStruct++

- Train loss: 3.6605, Val loss: 3.6944
- Test loss: 3.7825

➢ SPENInf

- Train loss: 2.1934, Val loss: 2.788
- Test loss:2.8611

➢ NLTop

- Train loss: 2.0067, Val loss: 2.2627
- Test loss: 2.4950

Hyperparameters	
Optimizer	SGD
Learning rate	1e-5
Batch size	100
Epochs	100
Hidden size	1152
activation	hardtanh

Reproduced Results

➢ Results analysis

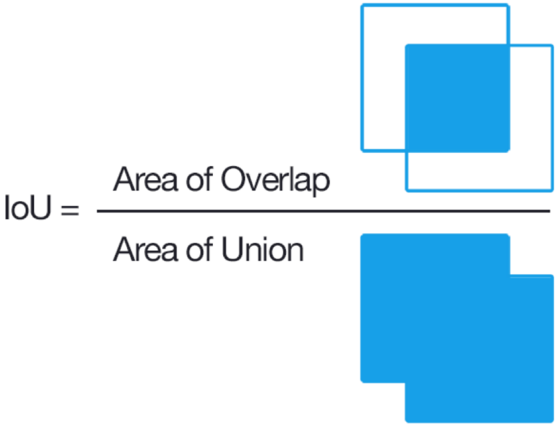
- Unary classifier converges fast and performs well
- The reproduced results are basically consistent with the experimental results in this paper
- Adding structure improves a non-structured model, and NLTop can capture global structure to further improve the performance
- It is difficult to achieve the state-of-the-art performance in this paper (too many hyperparameters, too much time to train 20+ h/100 epochs)

Experimental Results

- Semantic Segmentation
  - Weizmann Horses database
    - 328 = 196 + 66 + 66
    - No way to train from scratch
    - Use AlexNet pretrained on ImageNet
      - Remove first MaxPool
      - Change stride of second MaxPool (2 to 1)
  - Evaluation metrics
    - Intersection over Union (IoU)

Table 3: Results for segmentation experiments. All values are mean intersection-over-union

	Train	Validation	Test
Unary	0.8005	0.7266	0.7100
DeepStruct	0.8216	0.7334	0.7219
SPENInf	**0.8585**	**0.7542**	**0.7525**
NLTop	**0.8542**	**0.7552**	**0.7522**
Oracle	0.9260	0.8792	0.8633

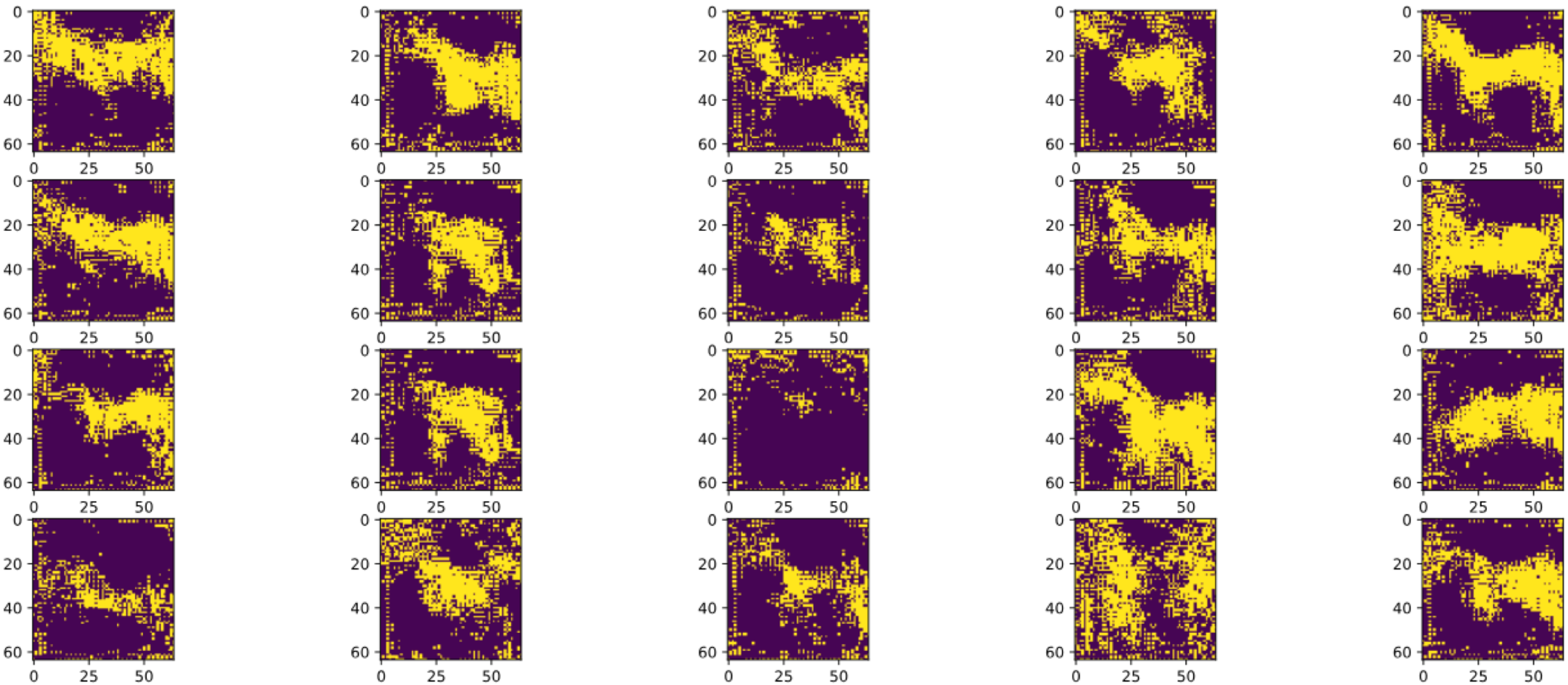


Reproduced Results

- Semantic Segmentation
  - Hyper-parameter searching
    - {optimizer, learning_rate, lr_scheduler, mp_eps, activation, … ...}
  - Unary model is already hard to tune
    - Performed 30+ experiments
    - Track parameter learning rate and gradient changes
  - The DeepStruct model take 0.5 hour to finish 1 epoch
    - Once encountered infinite loop when saving model
  - The NLTop model need to take the above two as input, thus block by the DeepStruct part.

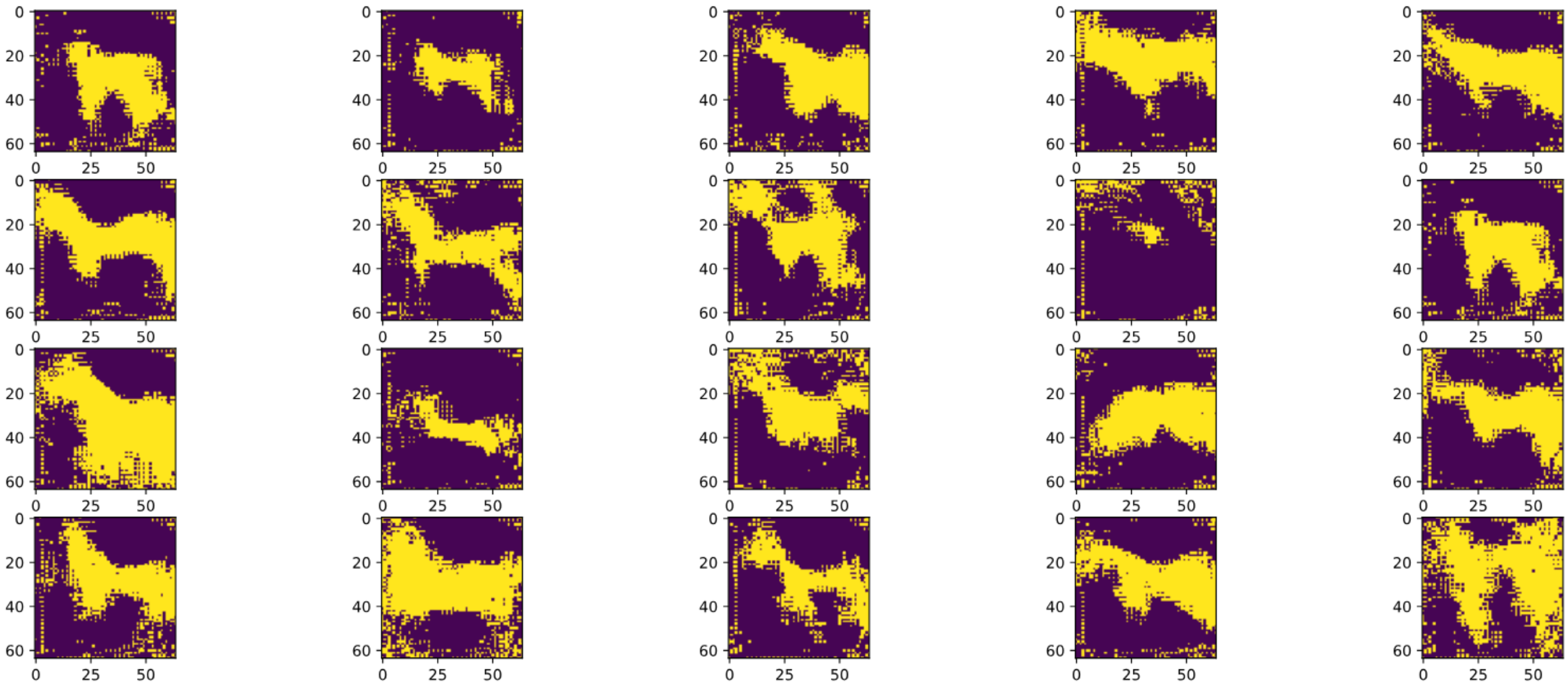
Model	Train	Validation	Testing
Unary	65.827	64.832	60.883
DeepStruct	32.419	32.339	32.162
NLTop	-	-	-

Reproduced Results



Sample prediction in development set from a IoU=53 model

Reproduced Results



Sample prediction in development set from a IoU=65 model

Experimental Analysis

- Adding structure improves model performance (DeepStruct vs unary)
- Adding implicit structure through output transformations improves an explicitly structured model (NLTop, LinearTop vs DeepStruct)
- Nonlinear transformation can get further improvement (NLTop vs LinearTop)
- Improvement in NLTop is not from increased number of parameters (NLtop vs DeepStruct++)

Conclusion and Future Work

- **Conclusion**
  - Proposed a framework that implicitly models higher-order structure as an intermediate layer in the deep net
  - Proposed an optimization framework which retains applicability of existing inference engines
  - Obtained performance improvement on a variety of tasks

- **Future Work**
  - Other possible architectures of the output transformation network
  - Other methods of solving inference
  - Accessing the applicability on tasks having variable sized outputs

Team Cooperation

- Paper discussion
- Slides
- Jupyter notebook
- Experiment:
  - Word Recognition: Guangtao Zheng
  - Multilabel Classification: Wenbo Pang
  - Image Tagging: Hanjie Chen
  - Semantic Segmentation: Sanxing Chen

References

1. J. Lafferty, A. McCallum, and F. Pereira. **Conditional Random Fields: Probabilistic Models for segmenting and labeling sequence data**. In *Proc. ICML*, 2001.
2. T. Finley and T. Joachims. **Training structural SVMs when exact inference is intractable**. In *Proc. ICML*, 2008.
3. B. Taskar, C. Guestrin, and D. Koller. **Max-Margin Markov Networks**. In *Proc. NIPS*, 2003.
4. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. **Large Margin Methods for Structured and Interdependent Output Variables**. *JMLR*, 2005.
5. A. Krizhevsky, I. Sutskever, and G. E. Hinton. **ImageNet Classification with Deep Convolutional Neural Networks**. In *Proc. NIPS*, 2012.
6. Carlo Ciliberto, Francis Bach, and Alessandro Rudi. **Localized structured prediction**. *arXiv preprint* arXiv:1806.02402, 2018.
7. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. **Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation**. In *Proc. NIPS*, 2014.
8. S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. **Conditional random fields as recurrent neural networks**. In *Proc. ICCV*, 2015.
9. T. de Campos, B. R. Babu, and M. Varma. **Character recognition in natural images**. 2009.
10. D. Belanger and A. McCallum. **Structured Prediction Energy Networks**. In *Proc. ICML*, 2016.
11. D. Belanger, B. Yang, and A. McCallum. **End-to-end learning for structured prediction energy networks**. In *Proc. ICML*, 2017.
12. L. Tu and K Gimpel. **Learning approximate inference networks for structured prediction**. In *Proc. ICLR*, 2018.
13. M. J. Huiskes and M. S. Lew. **The mir flickr retrieval evaluation**. In *Proc. ACM international conference on Multimedia information retrieval. ACM*, 2008.
14. E. Borenstein and S. Ullman. **Class-specific, top-down segmentation**. In *Proc. ECCV*, 2002.