MUST-CNN: A MUltilayer Shift-and-sTitch Deep Convolutional Architecture for Sequence-based Protein Structure Prediction

Zeming Lin, Jack Lanchantin, **Yanjun Qi**

Department of Computer Science at University of Virginia

Outline

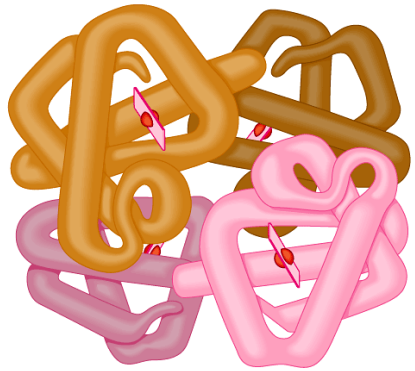
- Introduction
- Methods
- Experimental Design
- Results

Proteins

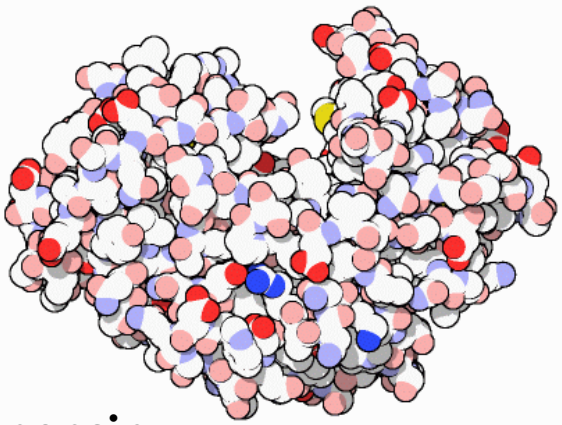
Protein Structure & Function

- Function: involved in almost everything

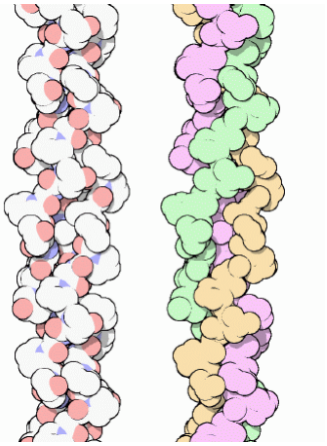
- Function depends on structure
  - 3-D structure
    - twisted, folded, coiled into unique shape



hemoglobin



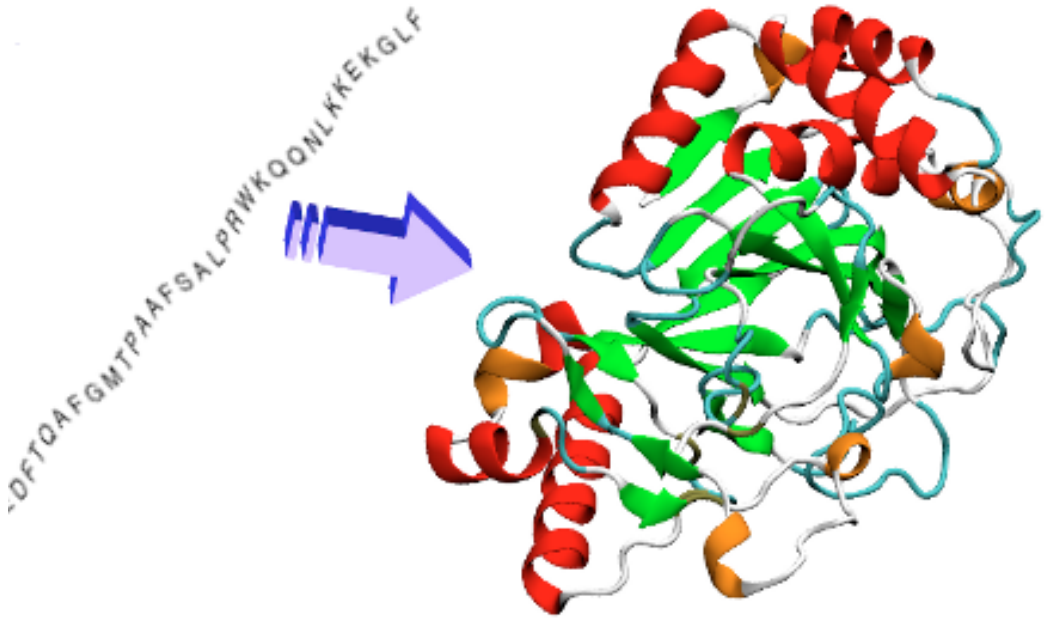
pepsin



collagen

Motivation

- Very time-consuming and expensive to measure protein structures experimentally



- Protein structure (right) is largely determined by primary sequence (left), but we don't know how!

Task: A Sequence to Sequence per-position classification task

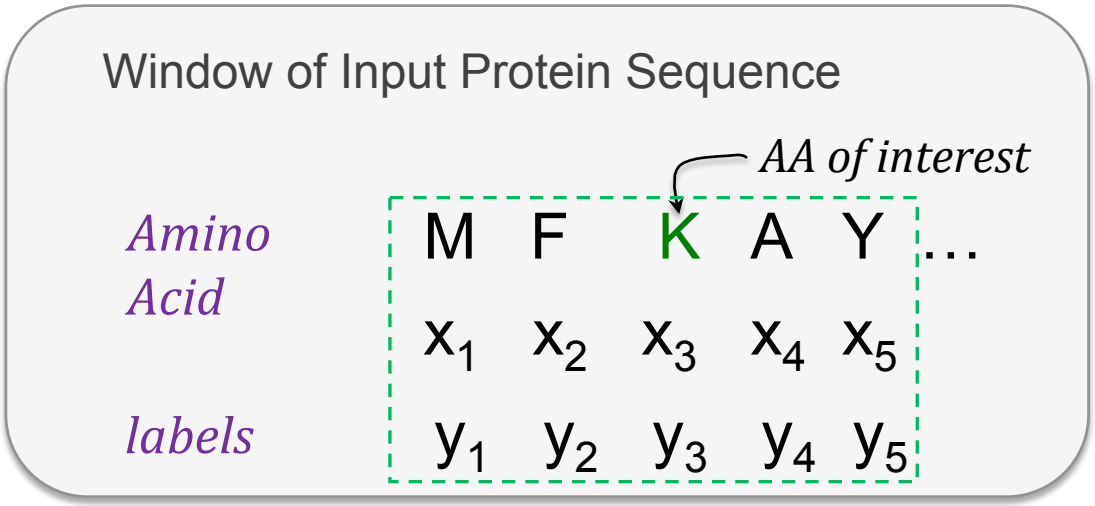
- Input X: Primary sequence (a string of amino acids -AA)

- Output Y: Structure Properties (a string of labels)
  - Secondary structure
  - Solvent accessibility

Multiple Output Targets:

Previous techniques:

- Create sliding "windows", predict per-position output one at a time with MLP
  - PSIPred, JPred, bioinfo-oriented projects



Labeling each amino acid (AA) using its context windows

Drawbacks

- Takes long time to train MLP- MultiLayer Perceptron (due to millions of labeled AA positions)
- Can not model long-range structured dependencies

This paper

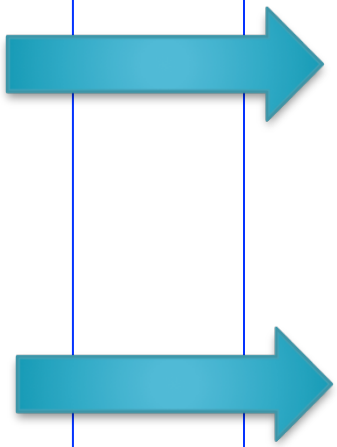
- Beat state of the art performance
- Fast training and testing, simple and scalable algorithm
- Generic enough to be applicable on other per-position labeling problems
  - E.g., NLP tagging tasks like part-of-speech tagging, name entity recognition

Outline

- Introduction
- Methods
- Experimental Design
- Results

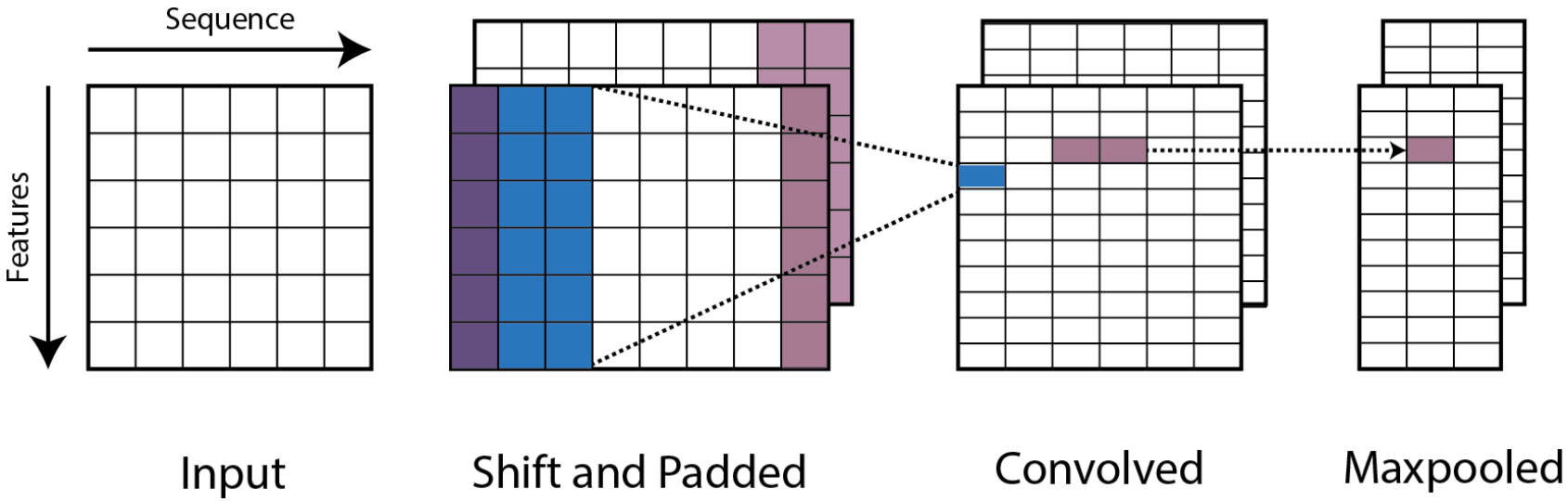
Proposed Solution: MUST-CNN

- Previous Drawbacks:
  - Long time to train (due to millions of labeled AAs)
  - Can not model long-range structured dependencies



- Proposed:
  - Convolution architecture + GPU
  - Deeper models

Convolutional Neural Network (CNN) for Sequence Input and Output

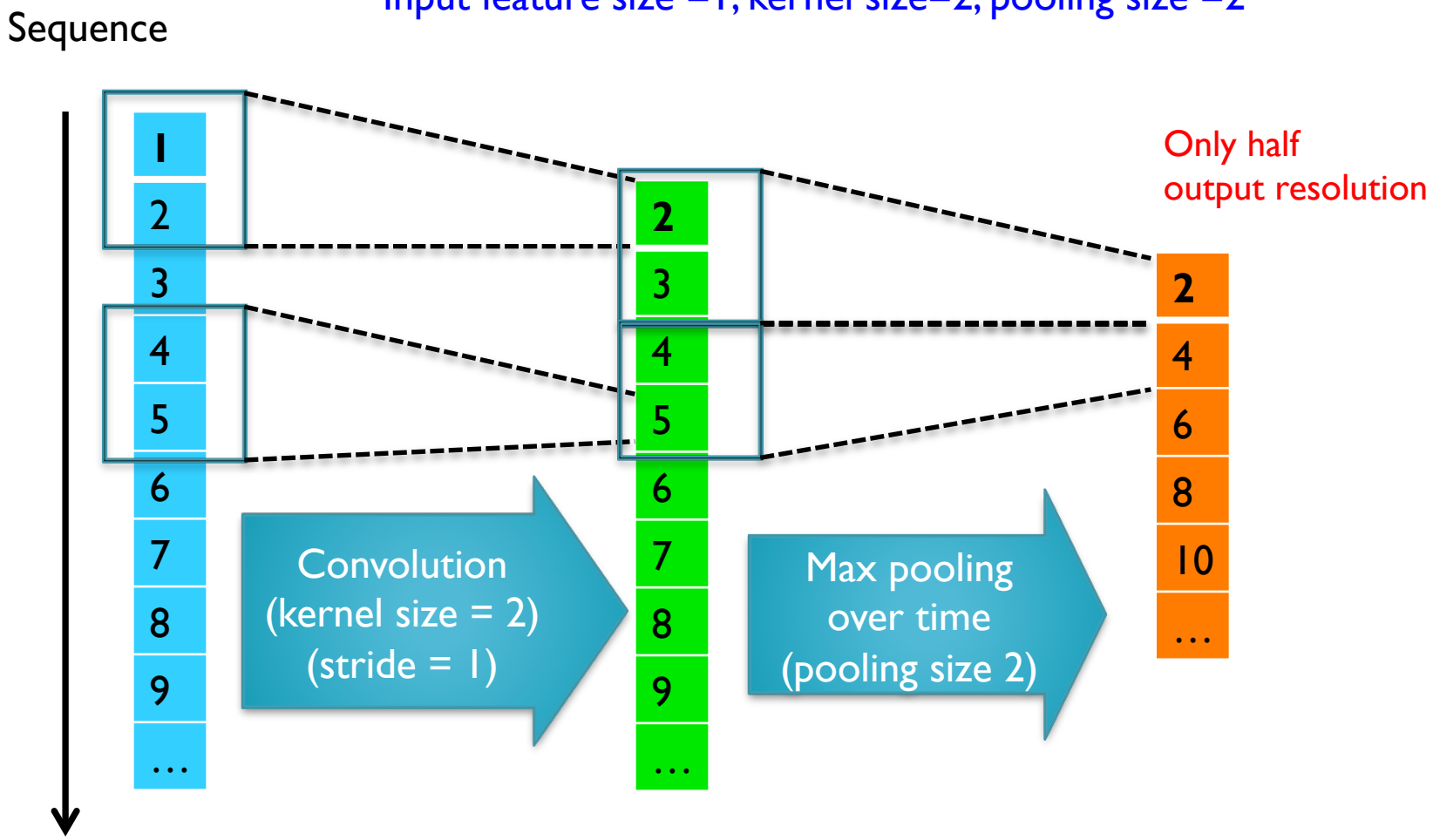


Not directly applicable !!!!

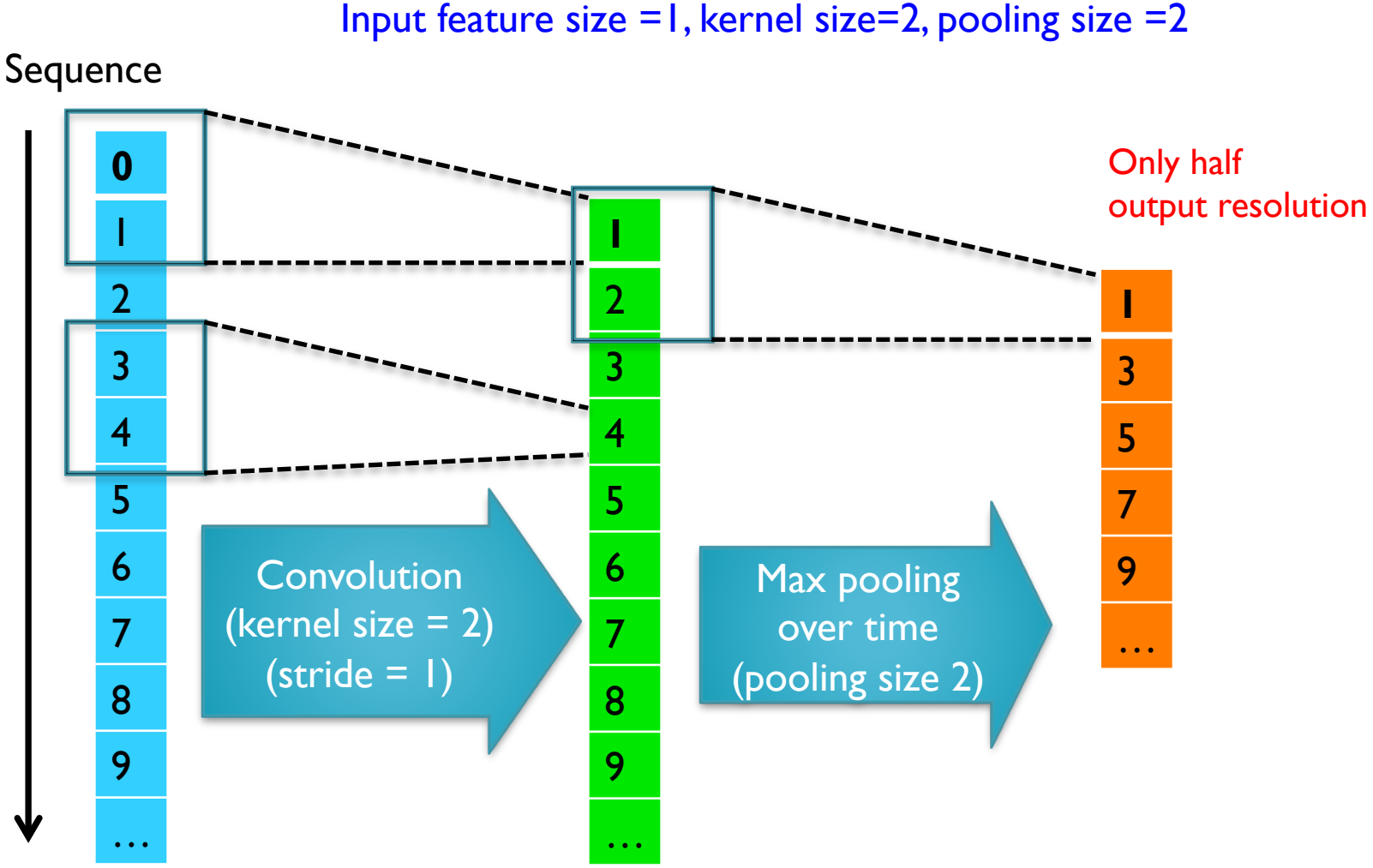
CNN's pooling step leads to a decreased output resolution.

CNN for Sequence Input and output : (toy case)

Input feature size =1, kernel size=2, pooling size =2

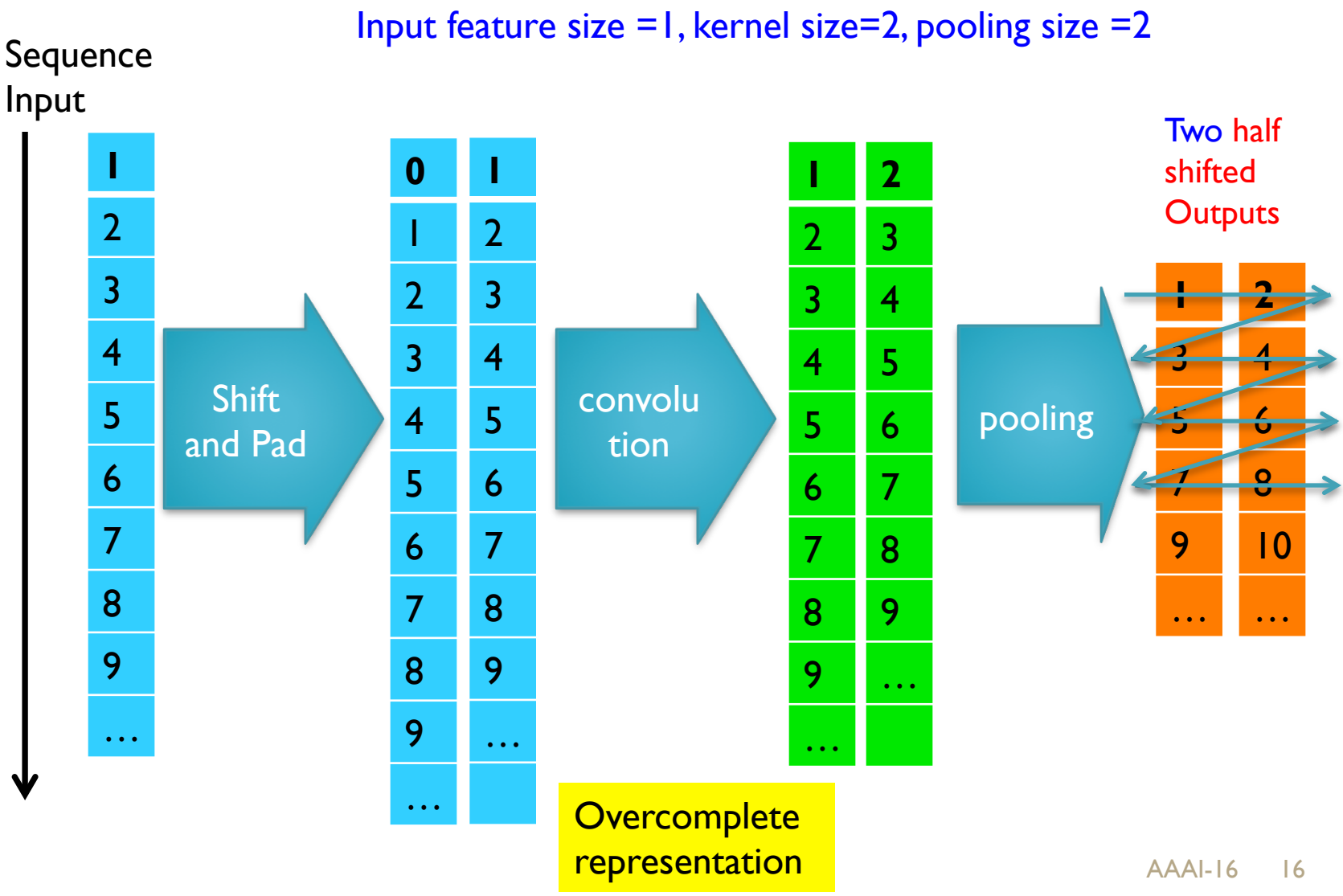


First Issue: Boundary Positions

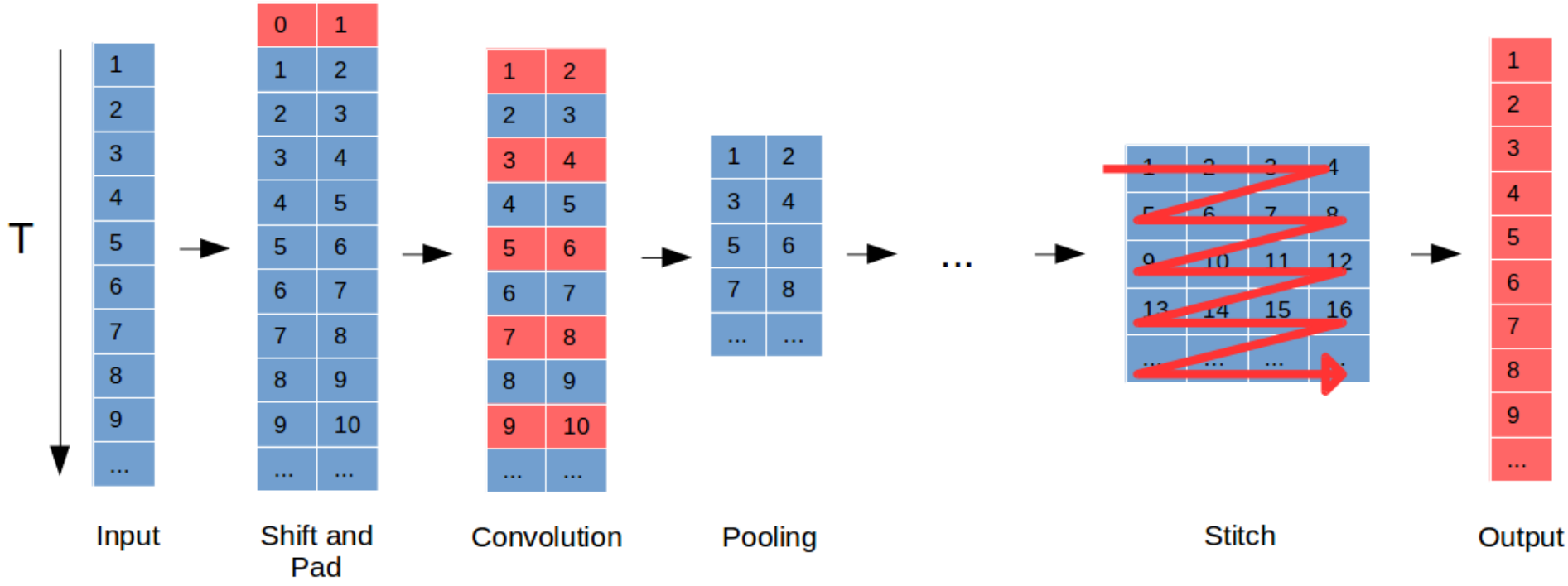


Second Issue: Per-position sequence to sequence classification

Solution: Shift and Stitch (toy example)



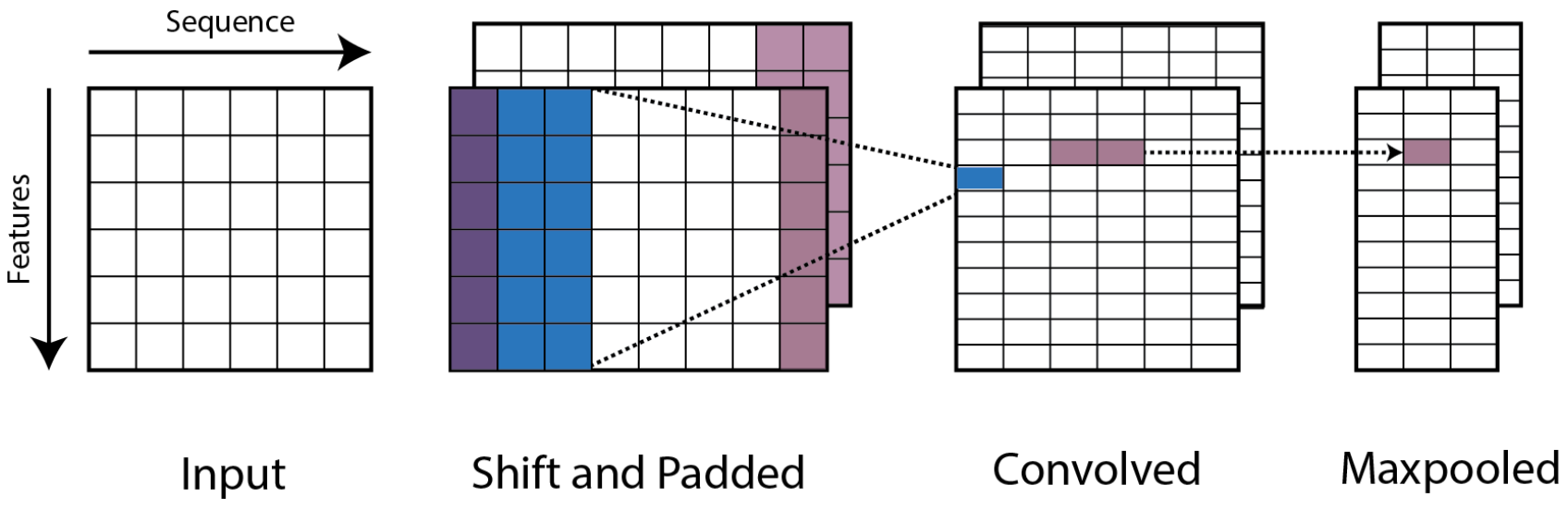
Generalize to Multilayer ➔ MUltilayer Shift and sTitch (MUST)



- MUST allows us to tag every element of an input with multilayer CNN all at once

MUST-CNN

- New sequence is a stratified version of previous sequence
- Simply view the output matrix in a different order to stitch sequence back together

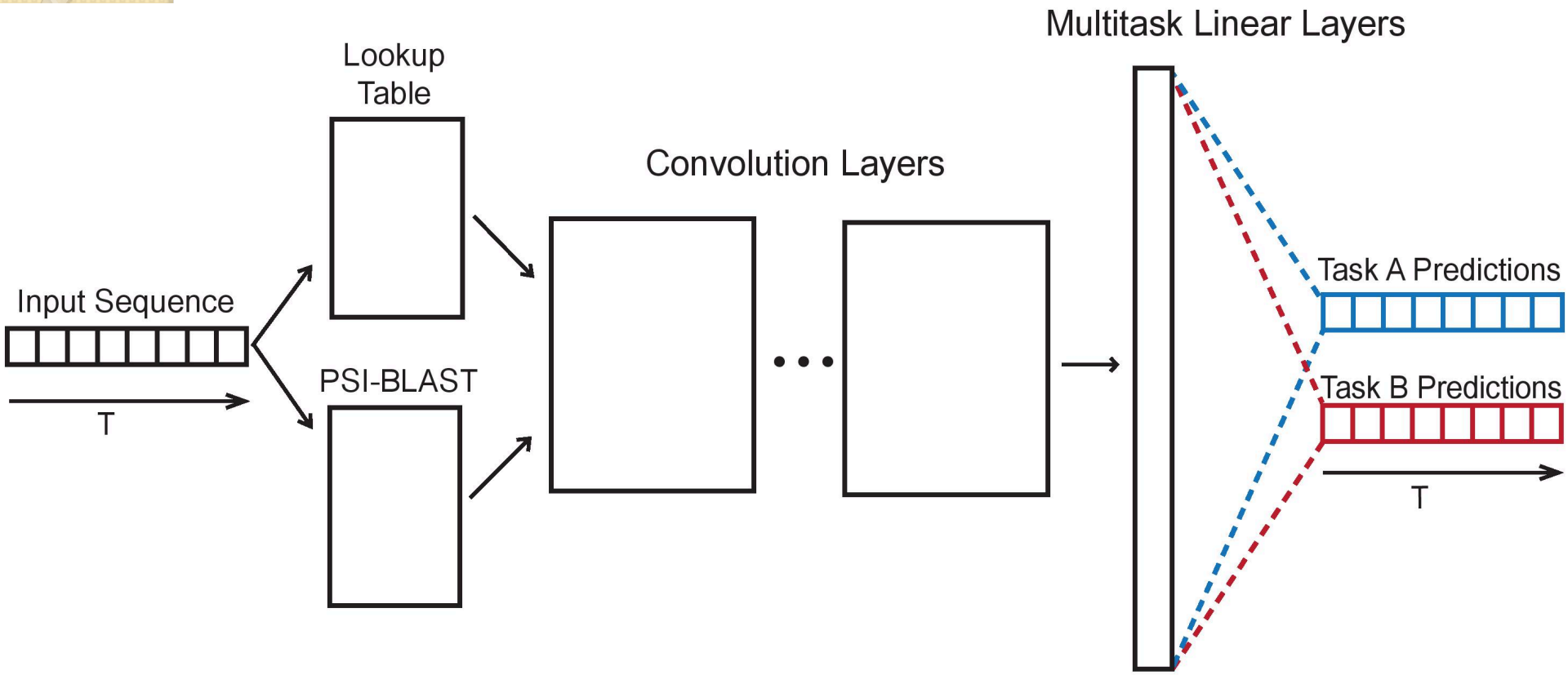


Advantages

- All operations are implemented easily.
  - Shift and pad is just duplication, zero padding, and concatenation
  - Stitching is just a vector reshape

- Fast, batched computations
  - Shift and stitch allows us to run batches of convolutions
  - CNN is highly parallelizable, easy to utilize parallel architecture like GPU
  - Output predictions for all-positions at once

Summary: End-to-End architecture



Connect to previous methods

- Conditional Neural Field (Wang, 2011)
  - Conditional Random Field with a NN feature extractor
  - Windowed, difficult to deal with long term dependencies, shallow
- Generative Stochastic Network (Zhou, 2014)
  - Trained similar as Restricted Boltzmann Machine
  - Slower convergence
- OverFeat (Sermanet et al. 2013)
  - Introduced shift-and-stitch, on per-pixel scene labeling
  - Not an end-to-end shift-stitch process (due to huge computational cost for 2D)
- LSTM (e.g., for machine translation, language model)
  - Protein sequences have no innate direction

Outline

- Introduction
- Methods
- Experimental Design
- Results

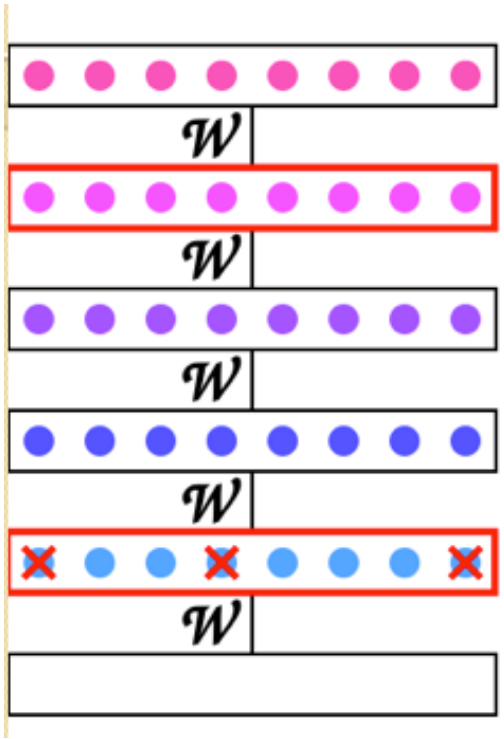
Implementation Details

- Each convolution layer also includes a nonlinearity.
  - ReLU Nonlinearity (Glorot, Bordes, and Bengio 2011)
  - PReLU nonlinearity (He et al. 2015)

- Pooling
  - Maxpooling does better than subsampling (Scherer, Müller, and Behnke 2010)

Implementation Details

- Dropout → randomized mask of outputs (Srivastava et al. 2014)
  - Randomly zeros out nodes of the network with probability during training
  - Act as a regularizer, and prevents overfitting



Implementation Details

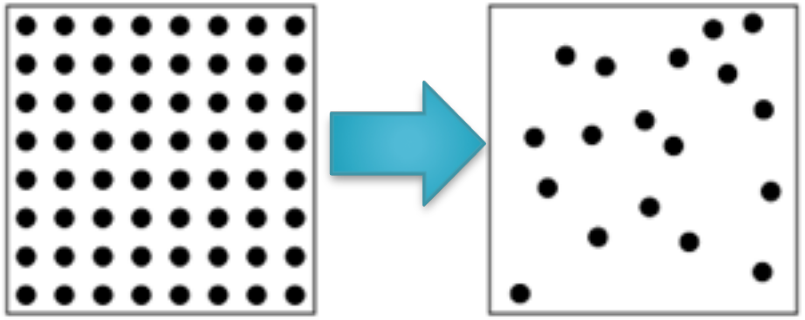
- Stochastic gradient descent for optimization
  - (Y. LeCun et al. 1998. Efficient BackProp.)

- Hardware:
  - All training and testing uses a Tesla C2050 GPU unit.

- Feature input of each amino acid
  - (AA embedding + PSIBlast)

Implementation Details

- Hyperparameter tuning (Snoek 15)
  - Grid search → sampling
  - Bayesian optimization



- Multi-task Learning
  - negative log-likelihood summing over all tasks and all elements in the sequences

Two large-scale Datasets

Data Name	Reference	Train Size (Num.AA)	Validate	Test
4prot	(from Qi et al. 2012.)	1.50 million	0.51 million	0.51 million
CullPDB	(from Zhou and Troyanskaya 2014.) Train with CullPDB, Test on CB513	0.95 million	0.24 million	85k

- Each dataset includes four labeled tasks

Outline

- Introduction
- Methods
- Experimental Design
- Results

First Data: 4prot

Four Tasks (num. out)	Qi. Et al (%)	Our final model (%)
Dssp (8)	68.2	76.7
Ssp (3)	81.7	89.6
Sar (2)	81.1	84.9
Saa (2)	82.6	86.1
Test Time	596k*	1597

- Q_C accuracy (C-class per-position accuracy)
- Test time given in milliseconds per million AA
- Baseline (Qi et al): window-based deep MLP (CPU)

Second Data: CullPDB

Methods	Q_8 accuracy (%)
CNF (Wang et al. 2011)	64.9
GSN (Zhou and Troyanskaya 2014)	66.4
LSTM (Kaae Sønderby and Winther 2014)	67.4
MUST-CNN (Ours)	68.4

- Our model is extremely simpler than three compared
- Trained on CullPDB, Tested on CB513.
- All models used the same data and same train-test splits.

Conclusion & Discussion

- Robustness: Same model does well on two different large datasets.
- Speed is key - predictions for half million samples in under two seconds.

- Fast and large Convnets outperform "fancier" complex approaches
- MUST-CNN is extendable as long as input and output sequence lengths match up

THANKS !

https://github.com/DeepLearning4BioSeqText/Paper16-AAAI-MUST-CNN