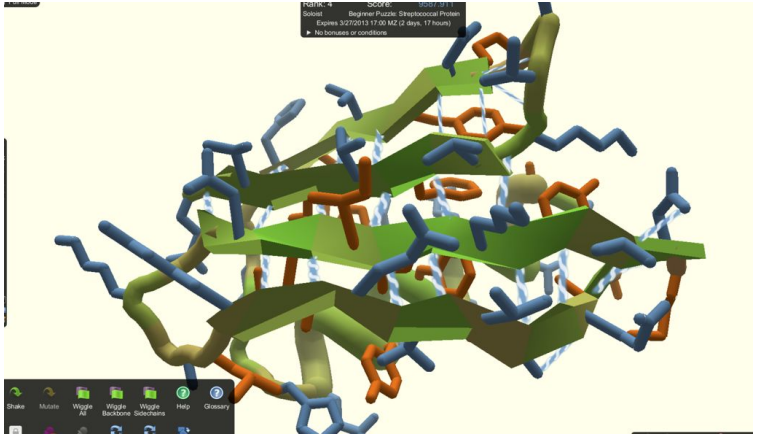
Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model

Charles Bournhonesque
David Golub
Charles Lu
Olivier Moindrot

Introduction

Sheng Wang, Siqi Sun, Zhen Li, Renyu Zhang and Jinbo Xu

Objective



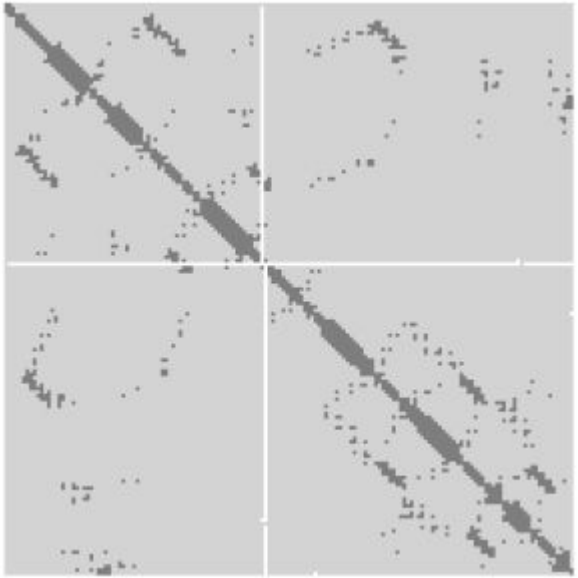
Objective:

- Predict the contacts in the 3D structure of the protein from the sequence

Motivation:

- Protein structure is crucial for understanding how the protein works
  - The structure specifies the function of the protein!
- Protein structure can be inferred from the **protein contact map**
  - contact-assisted protein folding

Data and output



Input:

- Protein sequence (amino-acids) of size L + predicted Secondary structure + solvent accessibility (Lx26)
- Coevolution features, pairwise potential (LxLx3)

Output:

- Protein contact map (boolean matrix of size LxL)
  - Contact = atoms closer than 8 Å

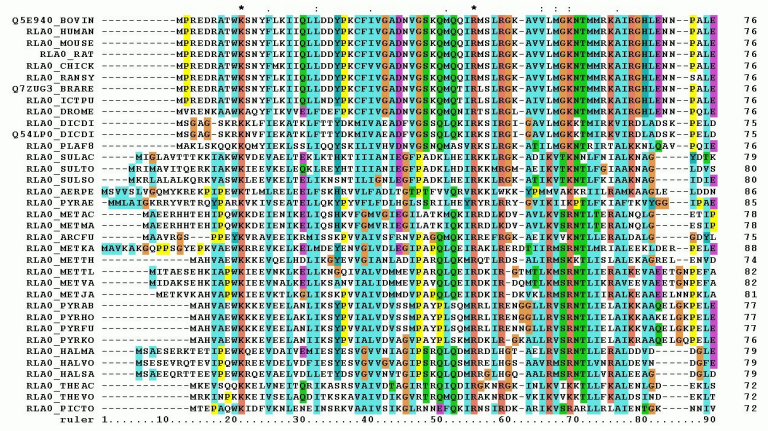
Train:

- 6767 proteins (from public PDB25 dataset)

Test:

- On public datasets (150 Pfam,105 CASP proteins, 76 CAMEO, 398 Membrane)

Existing methods



Evolutionary Coupling (EC)

- Multi Sequence Alignments (EVfold)
- Unsupervised

CON: needs a large number of sequences (several organisms) to be effective

Supervised ML methods

- use protein sequence + co-evolutionary information

CON: performances limited by shallow architecture (underfit ?)

Motivation for deep learning

- Predicting a contact-map is similar to predict the value of each ‘pixel’ of an image
  - Can use ideas borrowed from computer vision -> residual networks!

- Differences:
  - in computer vision, image-labelling is popular, not pixel labelling
  - images are resized to fixed length, but here sequence length varies
  - more complex features (sequential information + pairwise features)
  - unbalanced dataset (<10% contacts!)

- > Need to adapt the architecture of the model!

Model architecture

Model: Outline

1) Describe general architecture/motivation
2) Define what a ResNet Building Block is/motivation
3) Describe sequential architecture/motivation
4) Describe pairwise CNN/motivation
5) Training procedure/loss function

Model: General Architecture

Two ResNets

1st: encodes sequential features

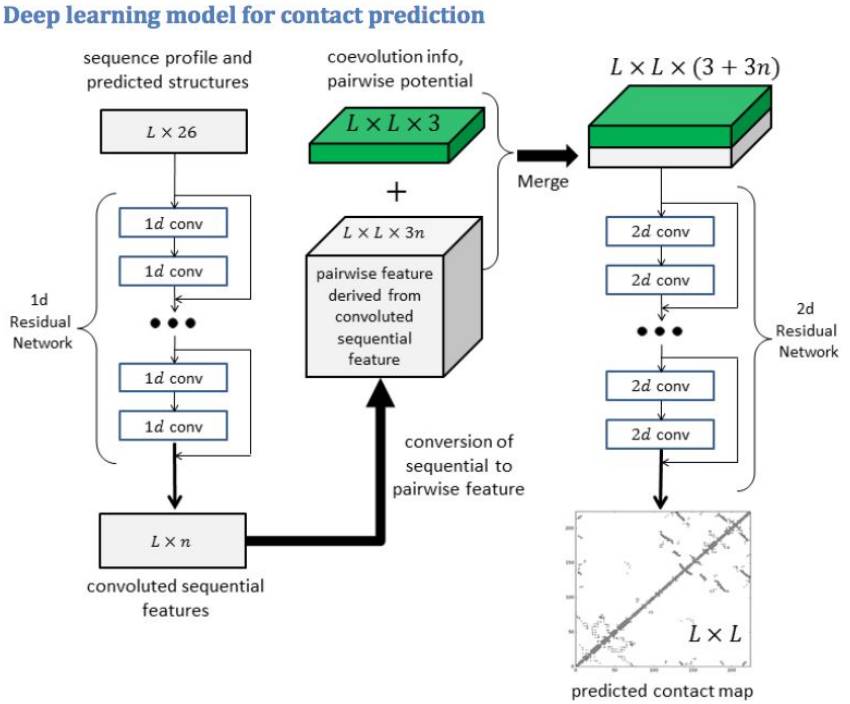
(6 layers)--fixed

2nd: encodes pairwise features & others

(60 layers)

Inspired by great success of ResNets,

frame protein contact as “pixel classification” problem



Model: ResNet Block Definition/Motivation

ResNet Block They Use:

f(x) = {1 Batch Norm

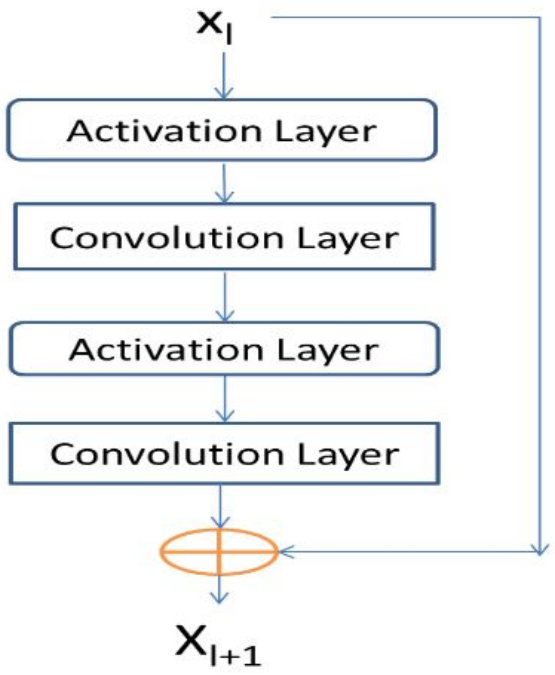
1 Activation Layer (ReLu)

1 Conv} x 2

res(x) = f(x) + x. Keep # filters fixed, i.e. 60.

(Use 0 padding to ensure dims f(x) and x match)

Motivation: Easier to train deeper neural nets.



Model: Sequential Architecture/Motivation

Architecture:

26 Input Features: seq. profile, pred. secondary structure and solvent accessibility

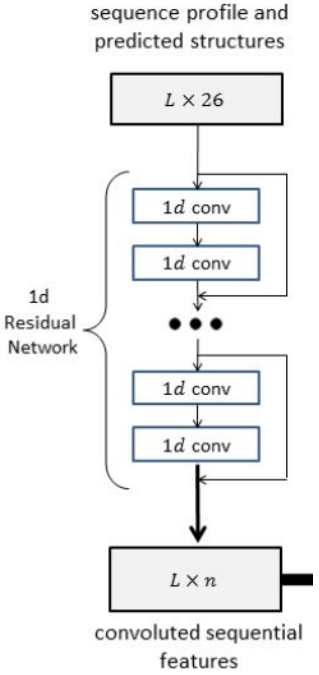
6 Convolutional Layers (i.e. 3 ResNet Blocks)

Filter Size = 17. Number of Filters = 60 in each layer.

Motivation:

Stacking convs together can model long-term dependencies between far-off features

Larger window size to compensate for for small # convs.



Model: Pairwise Architecture/Motivation

Pairwise Architecture:

~Features: Concatenate coev. info, pairwise potential with

Pairwise features g[i, j]: Concat {v_i, v_(i + j / 2), v_j}

~**DNN**: 60 convolutional layers, so 30 resnet blocks:

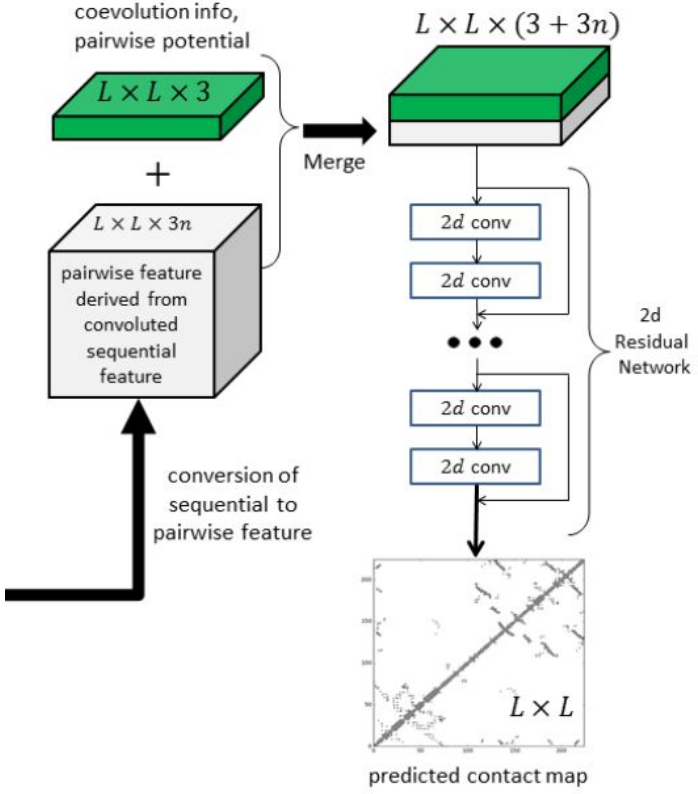
3x3 or 5x5 filters

~Logistic regression on top of outputs, binary classification

(fc layer? unclear)

Motivation: Deeper networks with smaller filters >

Shallower networks with larger filters



Model: Training Procedure/Loss Function

- Binary cross-entropy criterion for each contact point in LxL
- Upsample positives (so ⅛ weight of negatives)
- L2 Regularization to prevent overfitting
- “A stochastic gradient descent algorithm”--SGD, Adam, RMSProp?
- 20-30 epochs until convergence

Training set selection (6767 total). Sample ~6K for training, 700 for validation. Final model is average of 7 models (ensemble and take average prediction?):

- Exclude length < 26 > 700, resolution worse than 2.5 A
- Domains made up of multiple protein chains
- Inconsistency between PDB, DSSP, Astral sequences
- 25% identity or BLAST E-value 0.1 w/any test protein

Results

Comparison methodology

Compared to state-of-the-art methods:

- Direct evolutionary coupling analysis: PSICOV, Evfold, CCMpred
- Supervised learning: MetaPSICOV (could not evaluate PconsC2)

Evaluate accuracy of top *L/k* predicted contacts for *k*=10, 5, 2, 1

Contacts divided into short-, medium-, long-range categories for evaluation

Performance

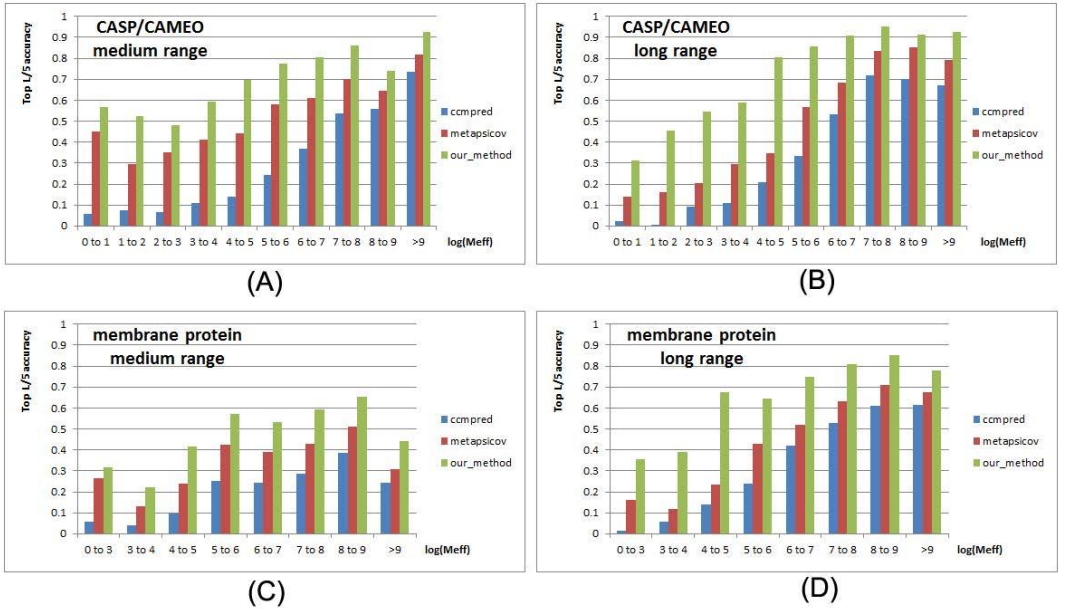
Long story short: outperformed everything for all categories, all *k*

- Smallest advantage on the 150 Pfam families, due to large number of sequence homologs
- *Sequence homolog*: a gene related to a second gene by descent from a common ancestral DNA sequence

Performance w.r.t. sequence homologs

Evaluated with respect to number of sequence homologs

- Measure effective number of sequence homologs with *Meff*
- Prediction accuracy increases w.r.t. *Meff*
- Outperforms MetaPSICOV and CCMpred
- Bigger advantage for *Meff*<1100

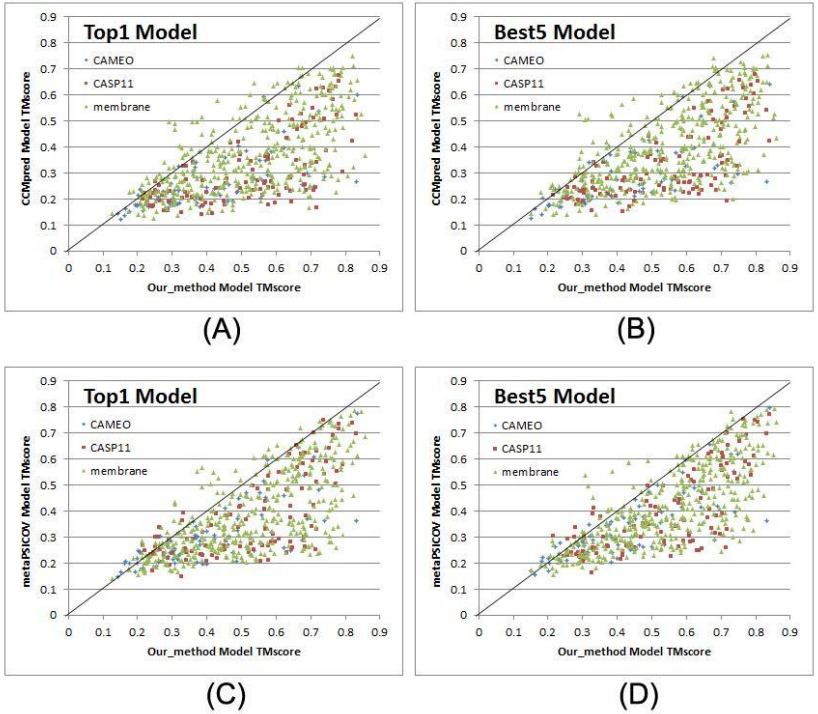


Contact-assisted protein folding

One main goal of contact prediction is to perform contact-assisted protein folding

Build structure model for all test proteins using top predicted contacts from different methods

- Feed top contacts as restraints into CNS suite to generate 3D models
- Measure quality of model using TMscore

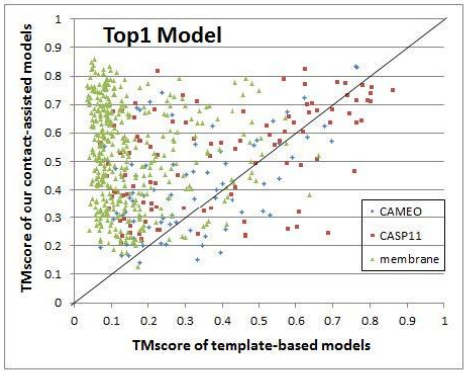


Contact-assisted vs. template-based models

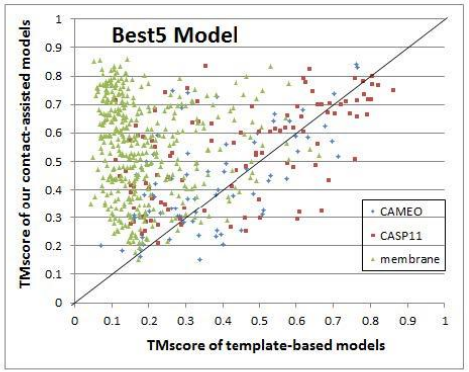
Template-based models generated for test proteins for comparison, using HHblits, HHsearch, MODELLER

This method still outperforms template-based models

- Even when query has no close templates
- Implies that this model does not predict contacts by simply copying contacts from training proteins
- Will be very useful for membrane proteins (many have no close templates in PDB)



(A)



(B)

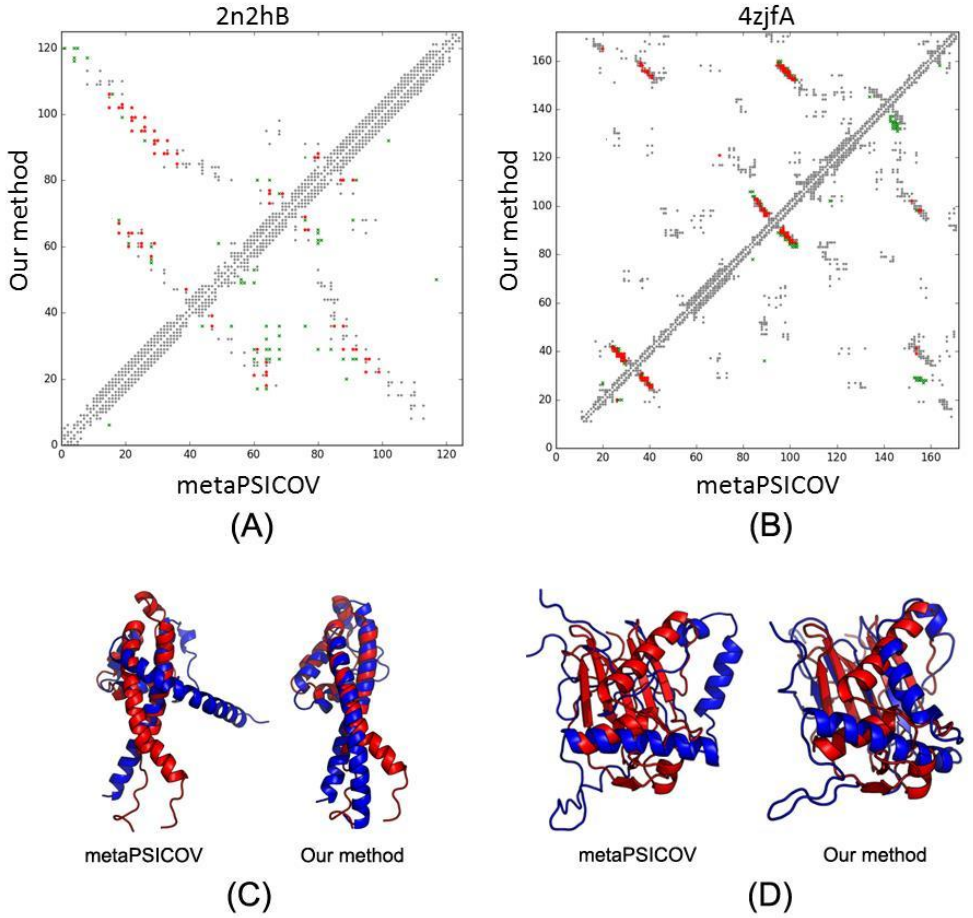
Examples

Two proteins

- Left: Sin3a
- Right: GP1

Triangles display top *L*/2 predicted contacts

- Grey: native contacts
- Red: correct predictions
- Green: incorrect predictions



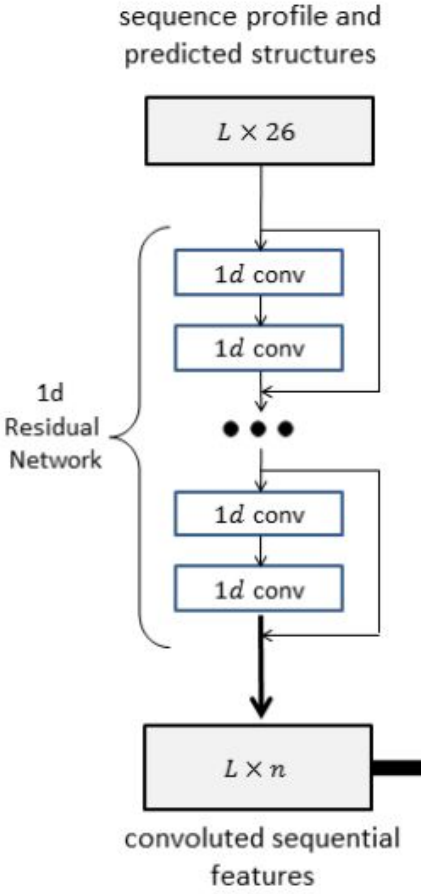
Comments

Choices of model architecture?

Architecture: sequential network

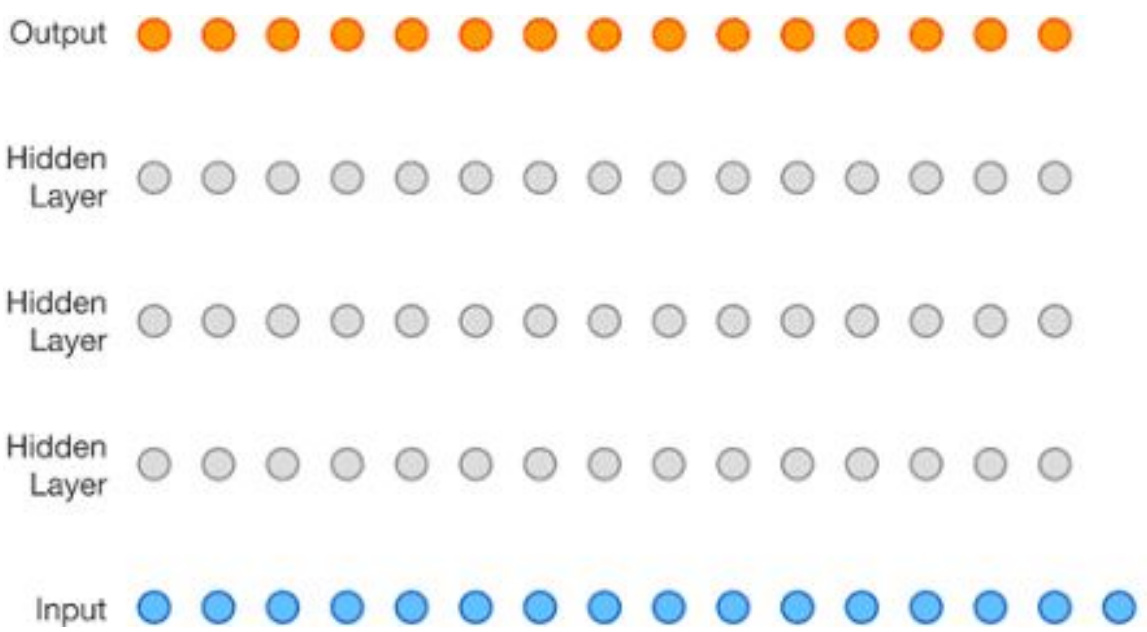
- Want to keep the length of the sequence
  - Sequence length: between 26 and 700
  - no downsampling (max pooling, stride > 1, padding = SAME)

- Tradeoff with the effective view field on the input
  - Without downsampling, and with only 6 convolutions with window size 17, 1 point in the output will only see a window of size 17 + 16*5 = **97** of the input
  - Forced to use very large window size (17) to be able to model **long-term dependencies**

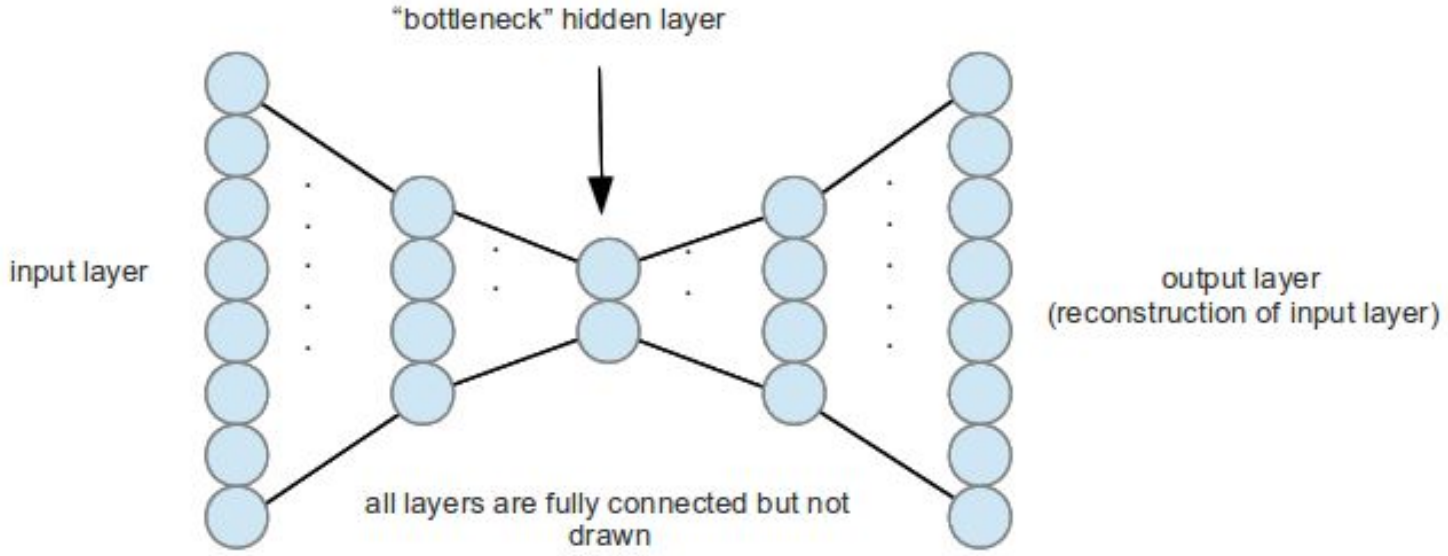


Atrous convolutions: increase the field of view

- Idea: use **atrous convolutions** (also called **dilated convolutions**)
  - No down-sampling
  - Fewer parameters



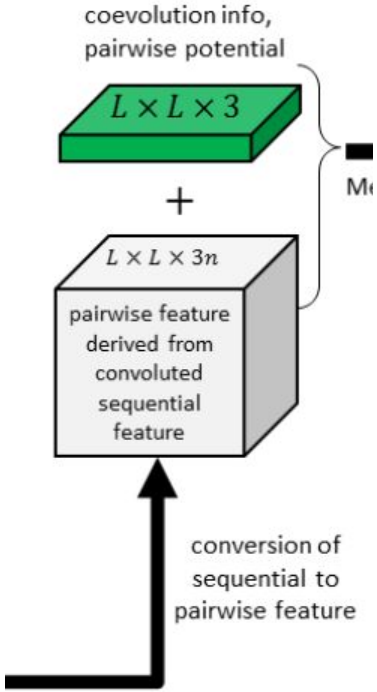
Encoder + Decoder: increase the field of view



From 1D to 2D: which transformation?

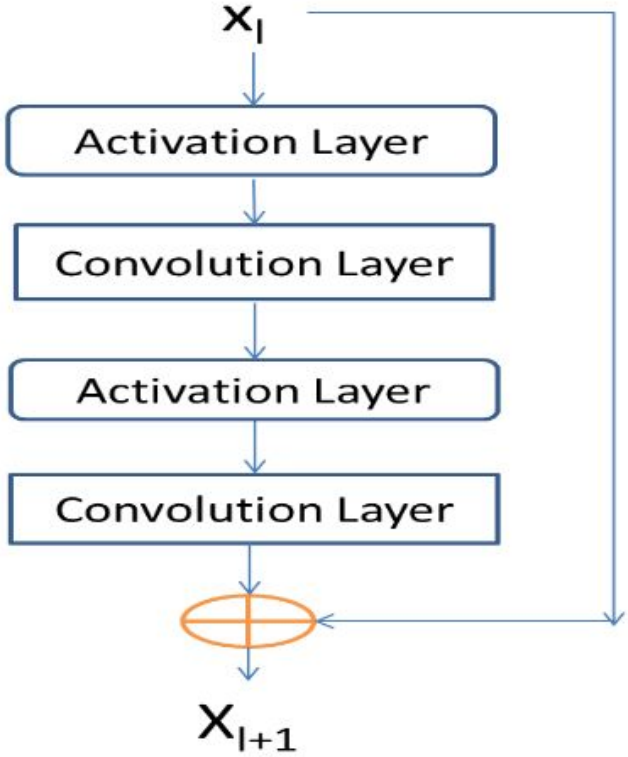
- For each pair (i, j), they use a combination of:
  - v_i
  - v_j
  - v_{i+j/2}
- Why use the middle amino-acid to predict contact between i and j?

- Could use simply:
  - Product scalar of v_i and v_j for every i, j
  - Would force v_i to be a real embedding of each amino-acid



Architecture: pairwise network (2D convolutions)

- 1x1 convolution instead of padding
  - When the dimension (i.e. number of filters) changes in ResNets
  - If x_l and x_{l+1} do not have the same dimension



Architecture: pairwise network (2D convolutions)

- They have 60 convolutions of window size 3 or 5
  - Field view on the input: 5+60*4 = 245
- Larger Resnet and less deep?
  - [Wide Residual Networks] (https://arxiv.org/abs/1605.07146)

depth	k	# params	CIFAR-10	CIFAR-100
40	1	0.6M	6.85	30.89
40	2	2.2M	5.33	26.04
40	4	8.9M	4.97	22.89
40	8	35.7M	4.66	-
28	10	36.5M	**4.17**	20.50
28	12	52.5M	4.33	**20.43**
22	8	17.2M	4.38	21.22
22	10	26.8M	4.44	20.75
16	8	11.0M	4.81	22.07
16	10	17.1M	4.56	21.59

Table 4: Test error (%) of various wide networks on CIFAR-10 and CIFAR-100.

Parallelization

- Couldn’t have more than 100 convolutions
  - Because of GPU memory limit
- Parallelization on multiple GPUs could have helped here
  - Bigger batch size
  - More depth
  - More filters

Training the model

- They use mini batches of examples of about the same length
  - Shorter examples are **padded with 0**
- Introduces a bias for each batch
  - The network trains on short sequences
  - Then on long sequences...
  - The resulting gradients could be noisy
- When testing, they **do not use padding**
  - Could have a small difference in performance compared to padding

Training the model

- Only 6,000 proteins
  - Small dataset
  - Too small to train very deep networks?
  - Should add even more regularization (with dropout)
- In general, difficult to train deeper models
  - Especially on a small dataset
  - Try to reduce the depth, and increase the field of view

Questions?

Thank you for your attention