Deep learning approaches for automated particle picking in cryo-EM

August 24, 2020

Israa Alqassem

Outline

- Background and problem definition
- Convolutional Neural Networks
- DeepPicekr
- DeepEM
- CrYOLO
- Topaz

Background and problem definition

Electron cryo-microscopy (cryo-EM) enables protein complex structure determination at high resolution

But, high-resolution cryo-EM requires the selection of hundreds of thousands of high-quality particles from micrographs

Manual picking:

- ➔ A labor-intensive step acts as a major obstacle for automated cryo-EM pipeline
- ➔ Time consuming
- ➔ Subjective, introduces bias and inconsistency

Solution, computational approaches

Existing computational methods

Generative

- Measures the similarity to a reference to identify particle candidates from micrographs; cross-correlation similarity measure
- E.g., template matching technique

☹ Requires the user to prepare an initial set of high-quality reference particles used as templates

☹ Error-prone, and fails when dealing with non-ideal datasets

Unsupervised

- Relies on an unsupervised learning manner, without any labeled training data

☹ Not fully automated, i.e., often combined with the template-matching or classification based approaches to achieve decent picking result

Discriminative

- First trains a classifier based on a labeled dataset of +ve and -ve examples, and then applies this trained classifier to detect and recognize particle images from micrographs

☹ Demands the user to manually pick a number of positive and negative samples for initial training

Existing computational methods

Generative

- Measures the similarity to a reference to identify particle candidates from micrographs, cross-correlation similarity measure
- E.g., template matching technique
- ☹ Requires the user to prepare an initial set of high-quality reference particles used as templates

Unsupervised

- Relies on an unsupervised learning manner without any labeled training data
- ☹ Not fully automated, i.e., often combined with the template-matching or classification based approaches to achieve decent picking result

Discriminative

- First trains a classifier based on a labeled dataset of +ve and -ve examples, and then applies this trained classifier to detect and recognize particle images from micrographs
- ☹ Demands the user to manually pick a number of positive and negative samples for initial training

X Not fully automated

Deep learning comes in handy...

Deep learning approaches can achieve better performance especially in large-scale data analysis than traditional machine learning approaches

Recent deep learning methods were capable of achieving an impressive and amazing level of artificial intelligence that can mimic the problem solving skills of human experts or even better!

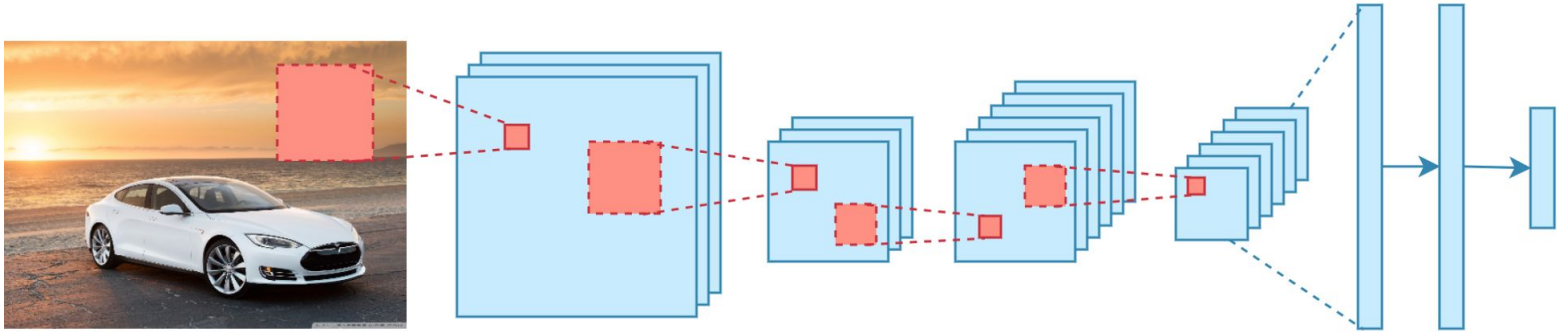
- ➔ Progressive Growing of GANs for Improved Quality, Stability, and Variation. 2017. https://www.youtube.com/watch?v=G06dEcZ-QTg

- ➔ You Only Look Once: Unified, Real-Time Object Detection. 2016
- ➔ YOLO9000: Better, Faster, Stronger. 2017

  https://www.youtube.com/watch?v=MPU2HistivI

Convolutional Neural Networks (CNNs / ConvNets)

- CNN is the state-of-the-art method for pattern recognition and object detection
- The main advantage of CNN is that it automatically detects the important features without any human supervision
- Computationally efficient; it uses special convolution and pooling operations and performs parameter sharing (next few slides)
- In CNN, the hidden layers mainly include convolutional, pooling and fully-connected layers
- The output layer is a softmax layer; probability value for each class

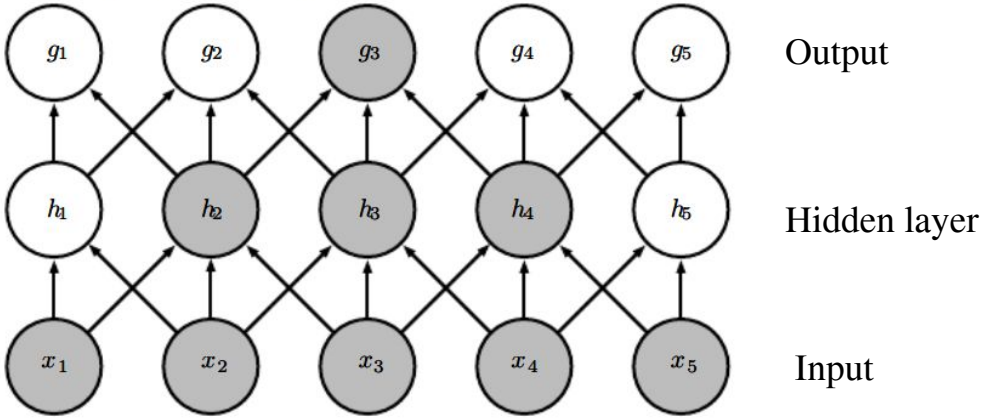


Convolutional Neural Networks (CNNs / ConvNets)

- **Full connectivity:** In traditional neural networks, every output unit interacts with every input unit
  - For m inputs, n outputs, the matrix multiplication requires m × n parameters
  - O(m x n) runtime per example
- **Sparse connectivity:** In CNN, by limiting the number of connections each output may have to k
  - k is several orders of magnitude smaller than m
  - O(k x n) runtime
- More efficient than dense matrix multiplication in terms of memory requirements, statistical efficiency (avoiding overfitting), and parameter sharing
- The convolution operation is performed by sliding the filter/kernel over the input. At every location, we do element-wise matrix multiplication and sum the result, think of it as a weighted sum, on the top of that a nonlinear function is applied
- Kernel ➡ weights ➡ extracts specific local features of an image, e.g., corners or edges

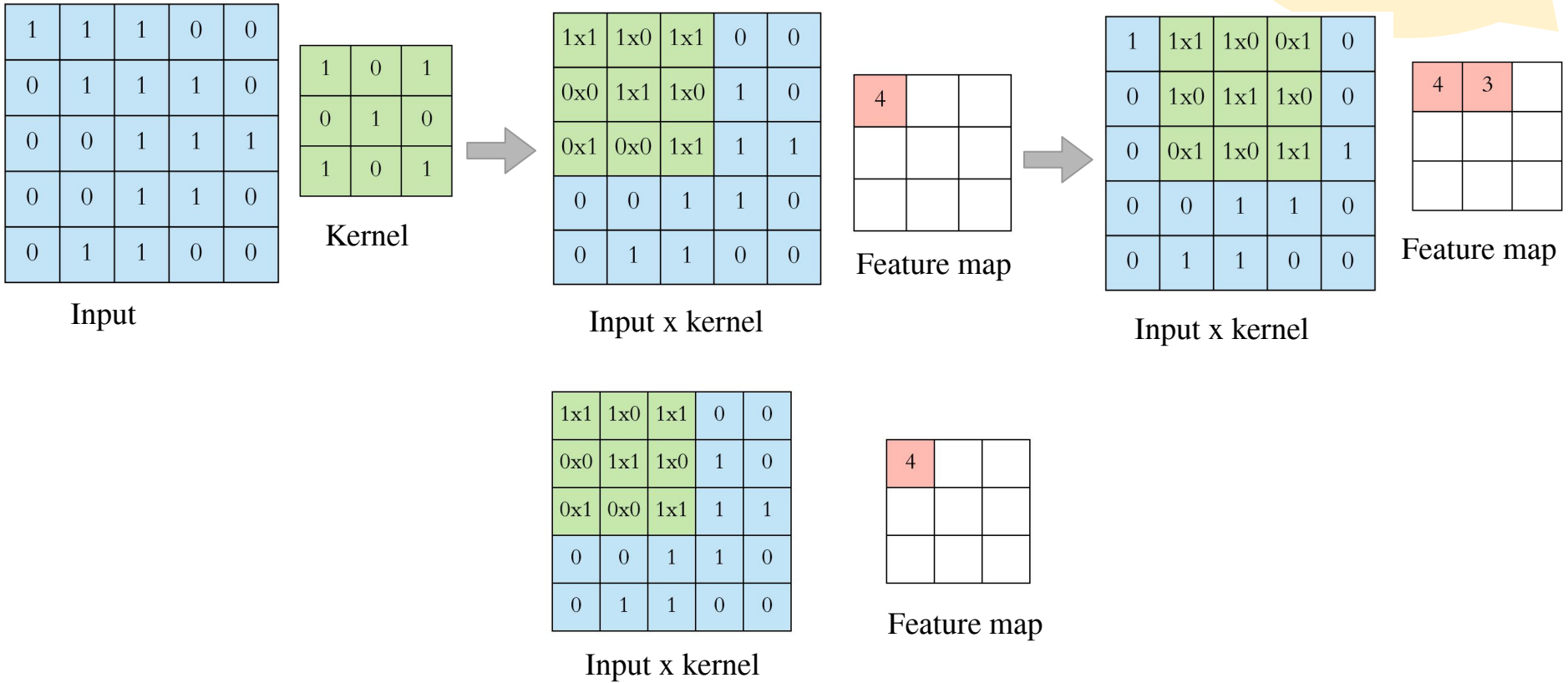
But, can sparse connectivity be efficient?

Even though, direct connections are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image



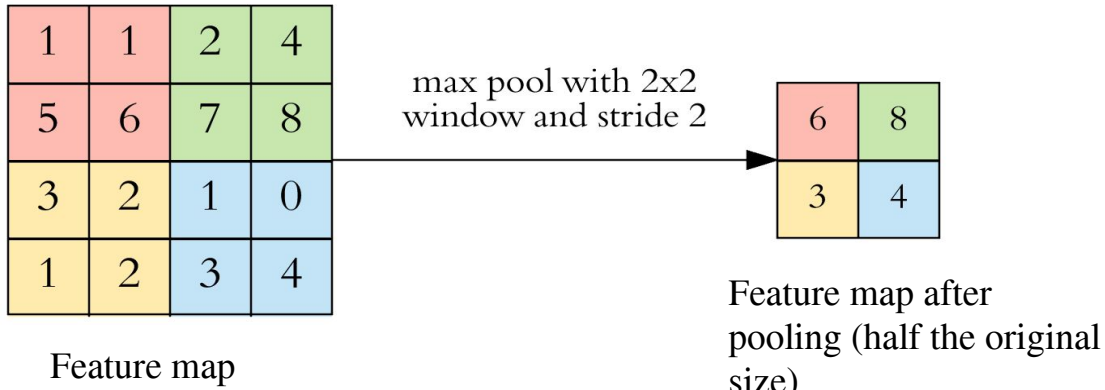
CNN Example:
3x3 convolution with stride of 1

Stride: how many steps we move the convolution filter at each step



Pooling Function

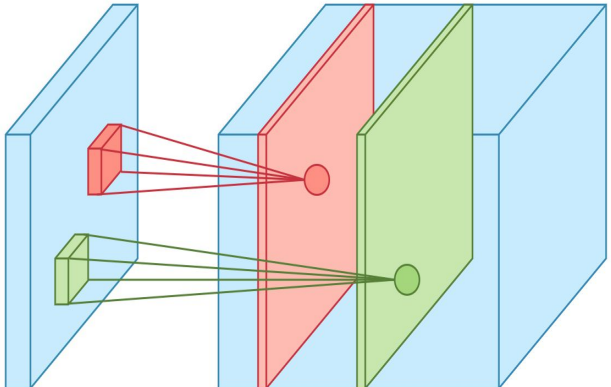
- Divides the input image into non overlapping windows
- Replaces the output of each window with a summary statistic of the nearby outputs
- Pooling function examples:
  - Max, average, L2 norm, or a weighted average based on the distance from the central pixel
- Pooling helps in making the representation approximately invariant to small translations of the input
  - If we translate the input by a small amount, the values of most of the pooled outputs do not change
- Reduces the number of parameters and thus the complexity of the learning model helps in avoiding overfitting



CNNs - In reality!

- An image is represented as a 3D matrix height, width and depth, where depth corresponds to color channels (RGB)
- Multiple convolutions can be performed, using different filters and resulting in distinct feature maps. After stacking all feature maps together we get the final output of the convolution layer
- The values in the final feature maps are not actually the sums
- The result of the convolution operation is passed through a nonlinear function, e.g., tanh(x), sigmoid(x), ReLU(x)

$$f(x) = \max(0, x + Y), \text{ with } Y \sim \mathcal{N}(0, \sigma(x))$$



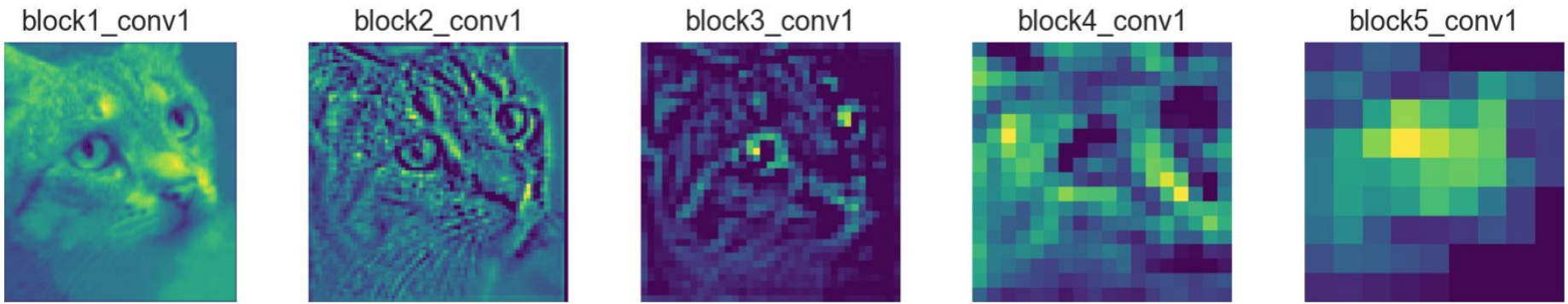
Data augmentation

Data augmentation:

- ➔ is a way to generate more training data from our current set.
- ➔ It enriches or “augments” the training data by generating new examples via random transformation of existing ones

Visualizing Feature Maps

- The first layer feature maps (block1_conv1) retain most of the information
- The deeper feature maps encode high level concepts like “cat nose” while lower level feature maps detect simple edges and shapes



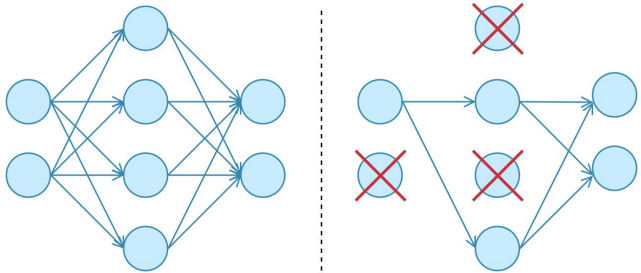
CNNs: Other definition

CNN Hyperparameters:

1) Filter size: we typically use 3x3 filters, but 5x5 or 7x7
2) Filter count: it's a power of two anywhere between 32 and 1024
   a) more filters leads to a more powerful model, but with the risk overfitting
3) Stride: how many steps we move the convolution filter at each step
4) Padding: used to maintain the same dimensionality of the feature maps, i.e., by surrounding the feature map with zeros
5) Dropout rate (p): at each iteration, temporarily "drop" or disabled a neuron with probability p (typically 0.5), applied only on training. (Why?)

CNN architecture composed of two components

a) Feature extraction part, convolution + pooling
b) Classification part, fully connected layers



Contents lists available at ScienceDirect

Journal of Structural Biology

journal homepage: www.elsevier.com/locate/yjsbi

DeepPicker: A deep learning approach for fully automated particle picking in cryo-EM

Feng Wang [c,1], Huichao Gong [b,1], Gaochao Liu [a,d,1], Meijing Li [a,d], Chuangye Yan [a,e], Tian Xia [c,*], Xueming Li [a,d,e,*], Jianyang Zeng [b,*]

[a] School of Life Sciences, Tsinghua University, Beijing 100084, China
[b] Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China
[c] School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China
[d] Beijing Advanced Innovation Center for Structure Biology, Tsinghua University, Beijing 100084, China
[e] Tsinghua-Peking Joint Center for Life Sciences, Tsinghua University, Beijing 100084, China

DeepPicker

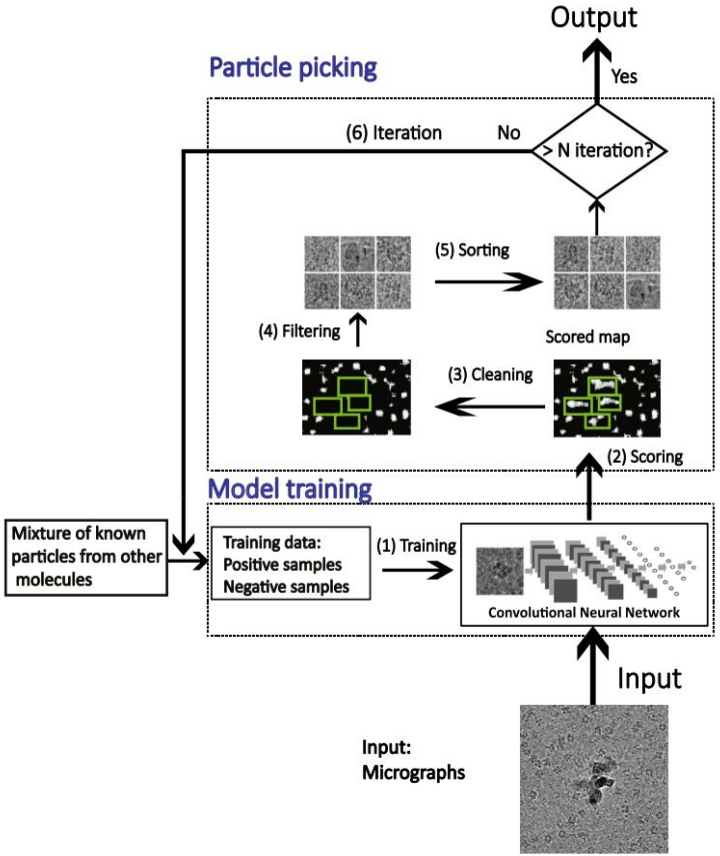
Automated particle picking pipeline

Employs a novel cross-molecule training strategy, using +ve and -ve labels

Particle picking: a sliding window of a fixed size (**?**) is used to scan each micrograph from the top left to the bottom right corner

The negative samples are selected from the regions at least 0.6 times the size of sliding window away from positive samples.

The positive samples are either from the known particles of other molecules (fully automated) or the manually picked particles of the target molecule (semi-automated)



https://github.com/nejyeah/DeepPicker-python

Cross-molecule training

Particles of different molecular complexes display distinct sizes and shapes

Two strategies:

- ➔ First, known particles from multiple types of other molecular complexes are combined to train the deep learning classifier and enable it to capture common features of particle
- ➔ Second, the particles of the target molecule identified from **the deep learning classifier trained by a cross-molecule manner in the first iteration** are further used to refine the model to better describe the features of the current target complex

DeepPicker: CNN Architecture

- Four convolutional layers, each uses an ReLU function and is followed by a max pooling layer followed by two fully-connected layers FC.
- The output of the last FC layer is fed into a softmax layer with two units, for the positive and negative labels
- Dropout is set to 0.5, applied in the first FC layer
- Input images normalized to 64x64 pixels

1	**No. kernels = 8**	**Kernel size = 1x9x9**
2	**No. kernels = 16**	**Kernel size = 8x5x5**
3	**No. kernels = 32**	**Kernel size = 16x3x3**
4	**No. kernels = 64**	**Kernel size = 32x2x2**
	1st FC has 256 neurons	**2nd FC has ? neurons**

$$P(Y_i) = \frac{e^{W_i X + b_i}}{\sum_j e^{W_j x + b_j}}$$

Yi: output i
i and j: the indexes of the artificial neurons in the last layer
X: input to soft-max
Bi: bias of the ith unit

Zhu et al. BMC Bioinformatics (2017) 18:348
DOI 10.1186/s12859-017-1757-y

BMC Bioinformatics

METHODOLOGY ARTICLE Open Access

A deep convolutional neural network approach to single-particle recognition in cryo-electron microscopy

Yanan Zhu[1], Qi Ouyang[1,2,3] and Youdong Mao[1,2,4*]

Abstract

Background: Single-particle cryo-electron microscopy (cryo-EM) has become a mainstream tool for the structural determination of biological macromolecular complexes. However, high-resolution cryo-EM reconstruction often requires hundreds of thousands of single-particle images. Particle extraction from experimental micrographs thus can be laborious and presents a major practical bottleneck in cryo-EM structural determination. Existing computational methods for particle picking often use low-resolution templates for particle matching, making them susceptible to reference-dependent bias. It is critical to develop a highly efficient template-free method for the automatic recognition of particle images from cryo-EM micrographs.

Results: We developed a deep learning-based algorithmic framework, DeepEM, for single-particle recognition from noisy cryo-EM micrographs, enabling automated particle picking, selection and verification in an integrated fashion. The kernel of DeepEM is built upon a convolutional neural network (CNN) composed of eight layers, which can be recursively trained to be highly "knowledgeable". Our approach exhibits an improved performance and accuracy when tested on the standard KLH dataset. Application of DeepEM to several challenging experimental cryo-EM datasets demonstrated its ability to avoid the selection of un-wanted particles and non-particles even when true particles contain fewer features.

Conclusions: The DeepEM methodology, derived from a deep CNN, allows automated particle extraction from raw cryo-EM micrographs in the absence of a template. It demonstrates an improved performance, objectivity and accuracy. Application of this novel method is expected to free the labor involved in single-particle verification, significantly improving the efficiency of cryo-EM data processing.

Keywords: Cryo-EM, Particle recognition, Convolutional neural network, Deep learning, Single-particle reconstruction

DeepEM

- Their model needs to be trained with a manually assembled dataset, sampling both true particle images (positive training data) and non-particle images (negative training data)
- Error back-propagation method is used to train the network
- The objective function is the squared-error loss function: $E_N = \frac{1}{2N}\sum_{n=1}^{N}\|t_n - y_n\|^2,$ where t_n is the target and y_n the value of the output layer in response to the nth input training image, 0 for non-particle and 1 for true particle
- The weights and biases of the CNN model are initialized with a random number between 0 and 1, and are then updated in the training process squared-error loss function used as the objective function
- Data augmentation: each particle image in the training dataset was rotated by 90°, 180° and 270°

DeepEM: CNN Architecture

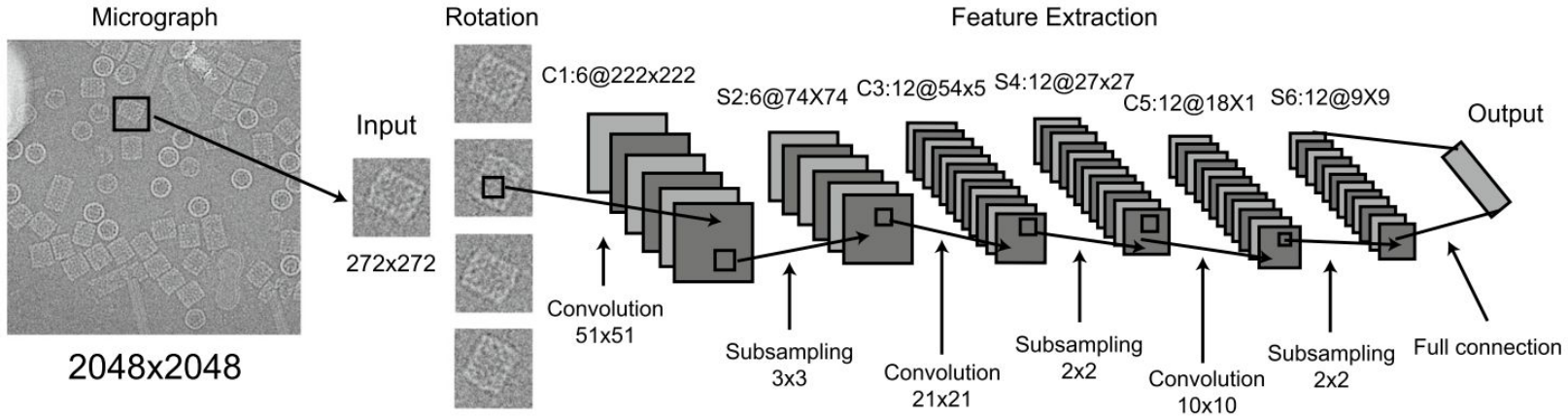


Fig. 1 The architecture of the convolutional neural network used in DeepEM. The convolutional layer and the subsampling layer are abbreviated as C and S, respectively. C1:6@222×222 means that it is a convolutional layer and is the first layer of the network. This layer is comprised of six feature maps, each of which has a size of 222 × 222 pixels. The *symbols* and *numbers* above the feature maps of other layers have the equivalent corresponding meaning

DeepEM: CNN Architecture (Cont'd)

- Sigmoid function is used as nonlinear activation function, $\text{sigmoid}(x) = 1/(1+e^{-x})$
- Subsampling (pooling) layers are used to reduce computational cost and potential overfitting
- Subsampling average is used
- The basic network architecture of DeepEM contains 3 convolutional layers each uses sigmoid function followed by subsampling average layer, and the last layer is FC
- They replaced the sigmoid activation function with a rectified linear unit;
  ReLU function gives rise to a slightly inferior accuracy in particle recognition than the sigmoid function

CrYOLO

Existing CNN models apply a specific classifier for an object and evaluate it at every position using sliding window

- High computational cost
- Not able to learn the larger context of a particle

YOLO framework (real-time object detection):

1) Needs only a single pass of the full image
2) Splits the image into a grid and predicts for each grid cell if it contains the center of a particle
3) If that's the case, it applies regression to predict the relative position of the particle

CrYOLO (Cont'd)

- CrYOLO: Python-based open source and uses the deep learning library Keras (neural network API)
- For feature extraction: 16 convolutional and four max-pooling layers
- For detection, a dropout layer (0.2) in front of (1x1) convolutional
- Data augmentation: flipping, blurring, adding noise and random contrast changes

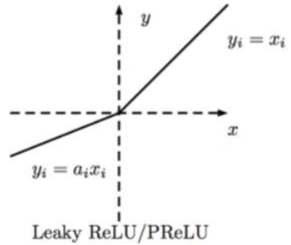
Feature Extraction			
1.	Convolutional	32	3x3
	Max-Pooling		2x2
2.	Convolutional	64	3x3
	Max-Pooling		2x2
3.	Convolutional	128	3x3
4.	Convolutional	64	1x1
5.	Convolutional	128	3x3
	Max-Pooling		2x2
6.	Convolutional	256	3x3
7.	Convolutional	128	1x1
8.	Convolutional	256	3x3
	Max-Pooling		2x2
9.	Convolutional	512	3x3
10.	Convolutional	256	1x1
11.	Convolutional	512	3x3
12.	Convolutional	256	1x1
13.	Convolutional	512	3x3
14.	Convolutional	1024	3x3
15.	Convolutional	512	1x1
16.	Convolutional	1024	3x3

Topaz

- Positive-unlabeled convolutional neural networks
- Classifier: parametric rectified linear units (PReLU)
- Three-layer convolutional neural network with striding and batch normalization
    1. 32 conv7x7 filters with batch normalization and PReLU, stride = 2
    2. 64 conv5x5 filters with batch normalization and PReLU, stride = 2
    3. 128 conv5x5 filters with batch normalization and PReLU
    4. A fully connected layer with a single output

$$f(\alpha, x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

Equation for Parametric Rectified Linear Unit(PReLU)



Leaky ReLU/PReLU

Discussion and Questions

- These methods vary widely in :
  - data preprocessing, e.g., image augmentation
  - Input resolution
  - Network architecture
  - The definition of negative labels
  - Cross-molecule vs. same molecule training
  - Output format, classification vs. regression
- Output accuracy is not enough, 3D molecular reconstruction is required to compare accuracies

Thanks!

References

- ★ Stanford course: CS231N Convolutional Neural Networks for Visual Recognition

  Materials: http://cs231n.github.io/

  Videos: https://www.youtube.com/playlist?list=PLkt2uSq6rBVctENoVBg1TpCC7OQi31AlC

- ★ Applied Deep Learning - Part 4: Convolutional Neural Networks
  https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

- ★ Deep Learning book
  https://www.deeplearningbook.org/

- ★ Topaz: https://github.com/tbepler/topaz

- ★ DeepPicker: https://github.com/nejyeah/DeepPicker-python/

- ★ DeepEM: http://ipccsb.dfci.harvard.edu/deepem/