What Did My CNN Learn?

-- an attempt to extract information from trained convolutional neural networks

Kuan-Hao Waylon Chen

Battelle Job Interview - Data Scientist III

May 7, 2019

Outline

What Did My CNN Learn?

0. My background

I. Motivation and Introduction to CNN

II. The approaches
   1. Visualize intermediate activations
   2. My CNN encoder and interpreters – some hybrid models
   3. Did my CNN learn translational symmetry?

III. Conclusion

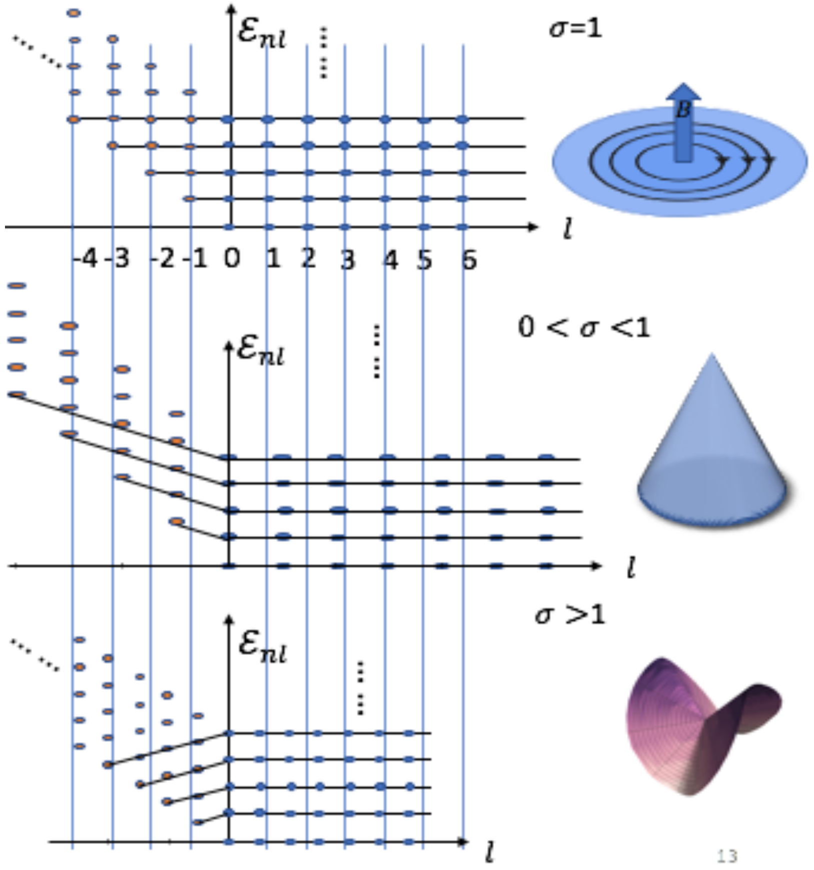
My Background

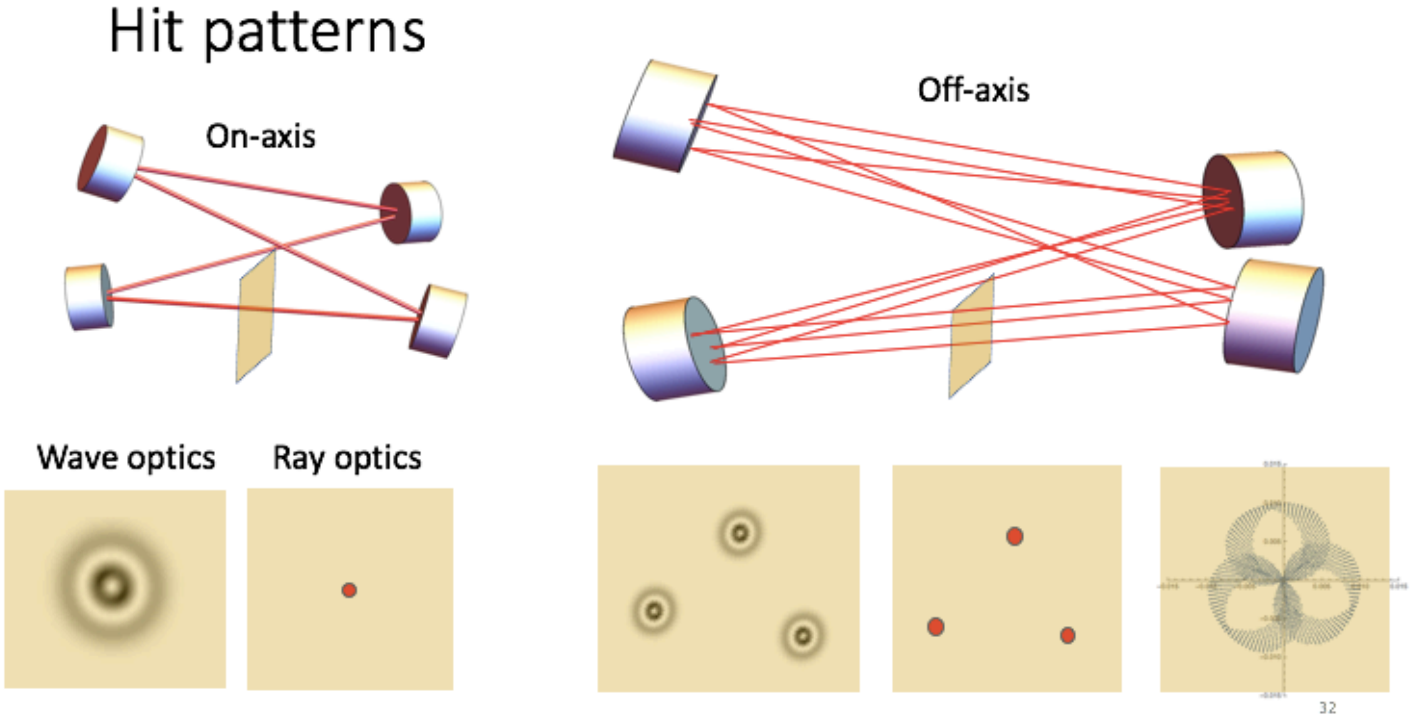
1. PhD research in Physics
2. Machine learning engineer intern

My background-I

The Ohio State University

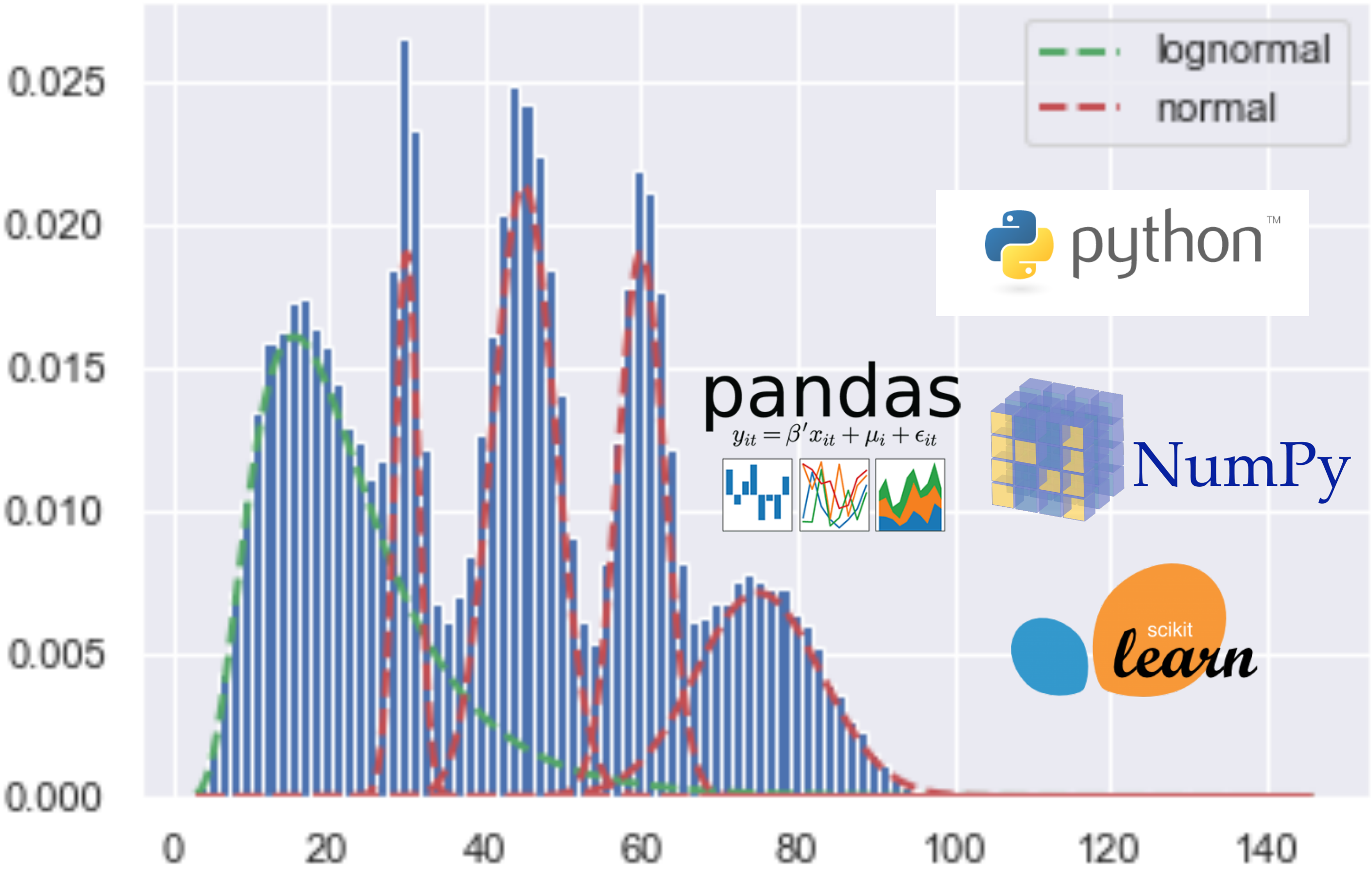
- My PhD in Physics, 2013-2018
  - Geometry and symmetry of quantum Hall physics with ultra-cold atoms and Laser
  - Discovered a novel quantum Hall phenomenon in Laser system





My background-II

- Machine Learning Engineer Intern, Owens Corning, 2018
  - Parametrize custom mixture model with normal and log-normal
  - Automate data cleansing, parsing, lag-time aggregation

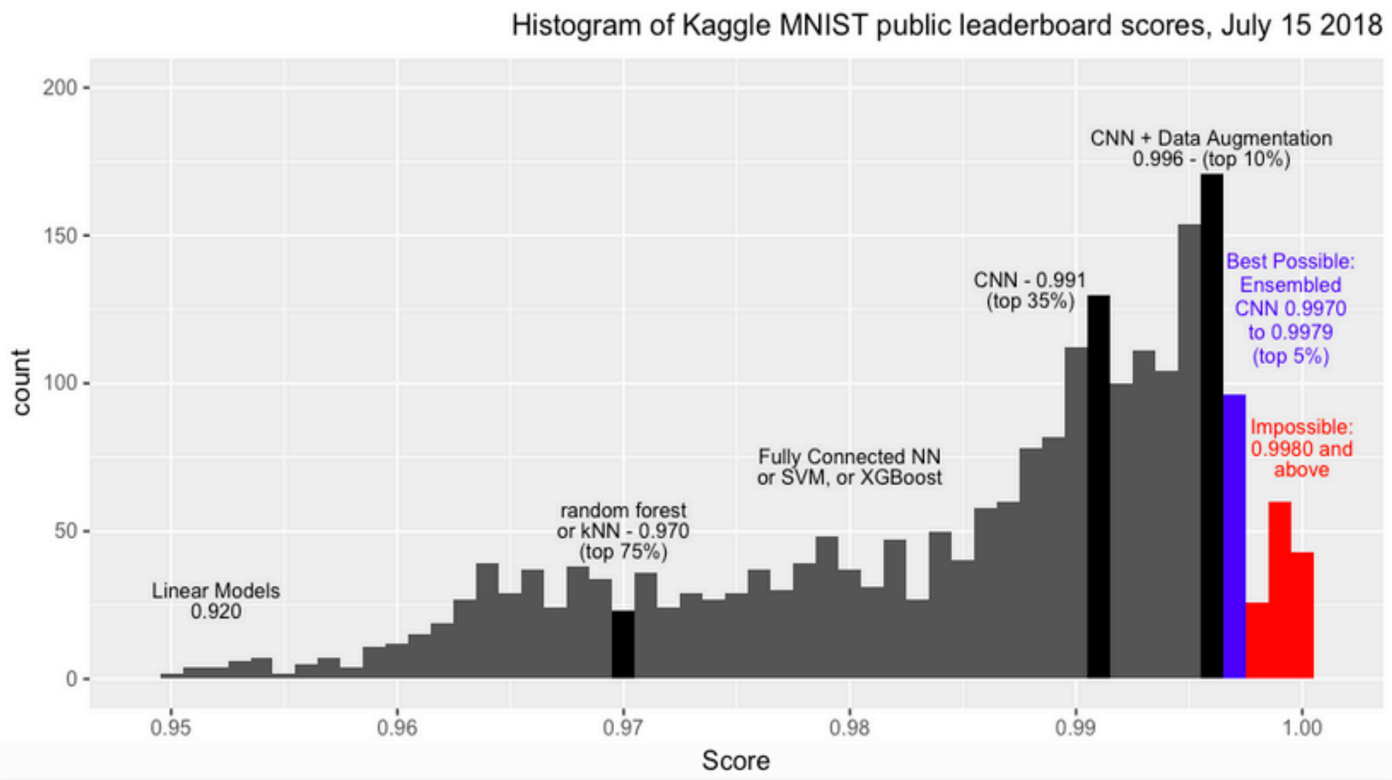


I. Motivation and Introduction to CNN

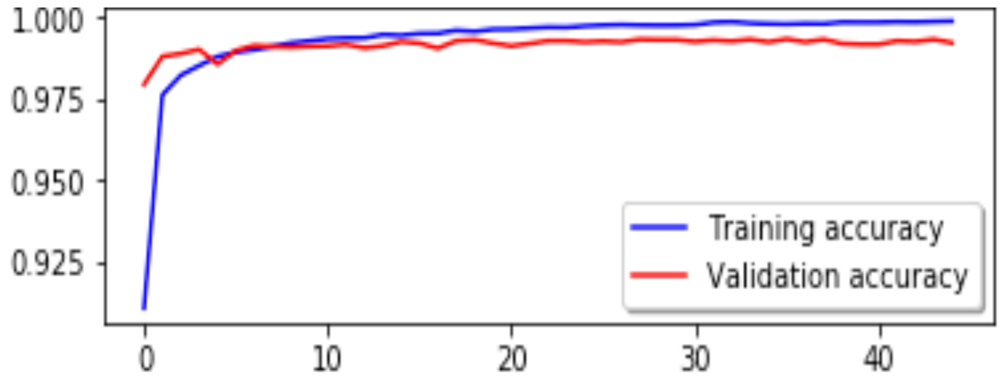
1. Motivation
2. Neural network
3. Convolutional network

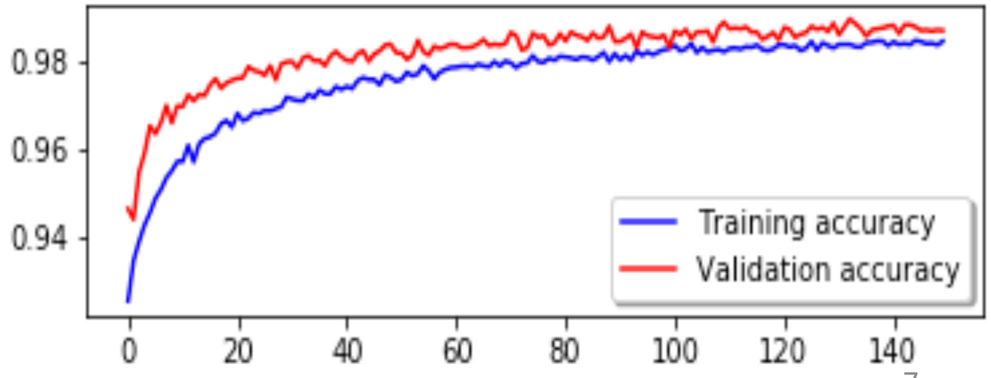
Motivation

1. My CNN classifier achieved 99.3% accuracy on MNIST hand-written digit data test set, but I cannot interpret it!
2. Knowing the information it did/did not learn can be the key to improvements.
3. *Validation accuracy exceeds training accuracy after data augmentation!*



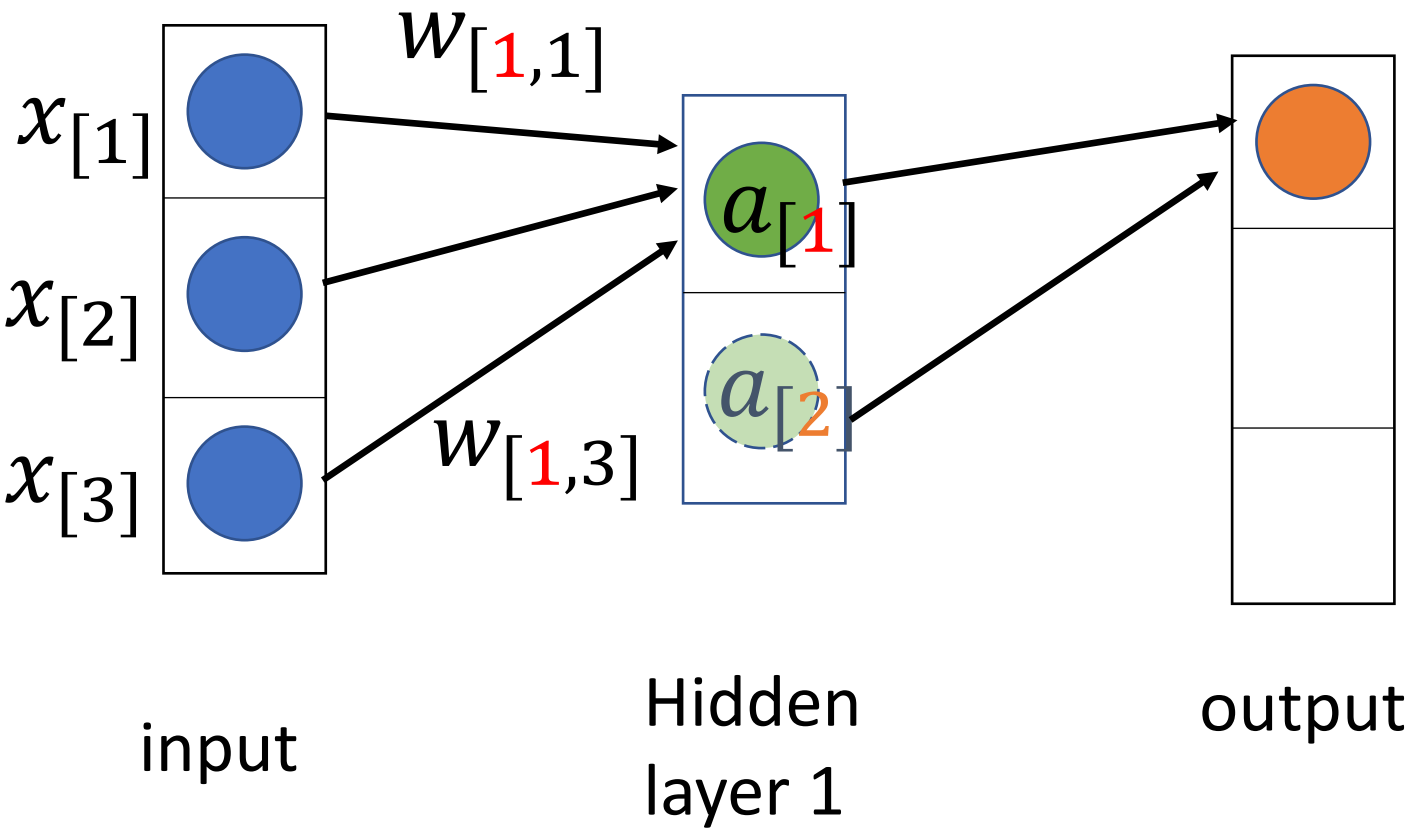
https://www.kaggle.com/cdeotte/25-million-images-0-99757-mnist





Neural network

$$a_{[1]} = \sigma\left(w_{[1,1]}x_{[1]} + w_{[1,2]}x_{[2]} + w_{[1,3]}x_{[3]} + b_{[1]} \cdot 1\right)$$



σ: activation function, nonlinear

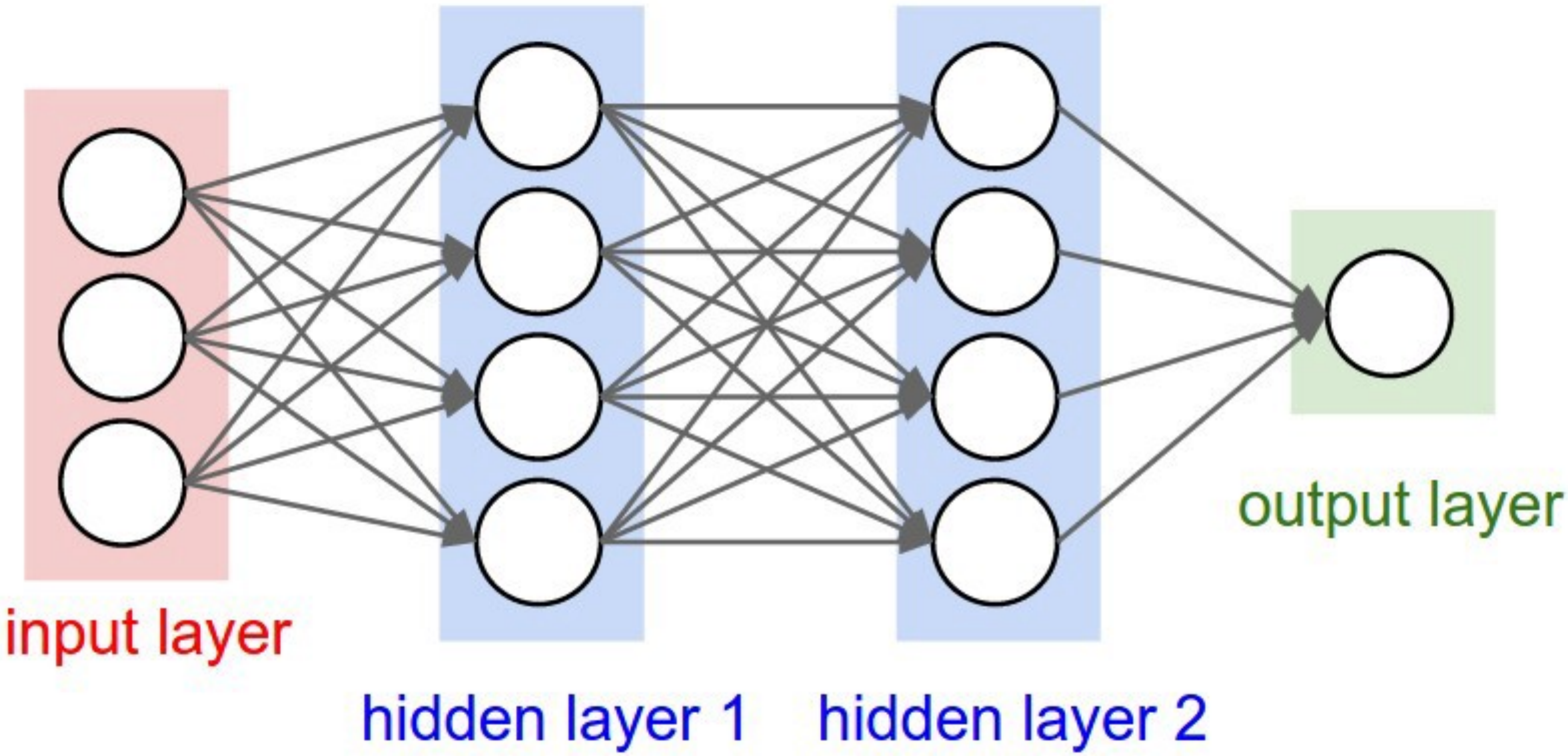
$$\vec{a} = \sigma\left(\overleftrightarrow{w} \cdot \vec{x} + \vec{b}\right)$$

$$\vec{a}^{(1)} = \sigma\left(\overleftrightarrow{w}^{(1)} \cdot \vec{x} + \vec{b}^{(1)}\right)$$

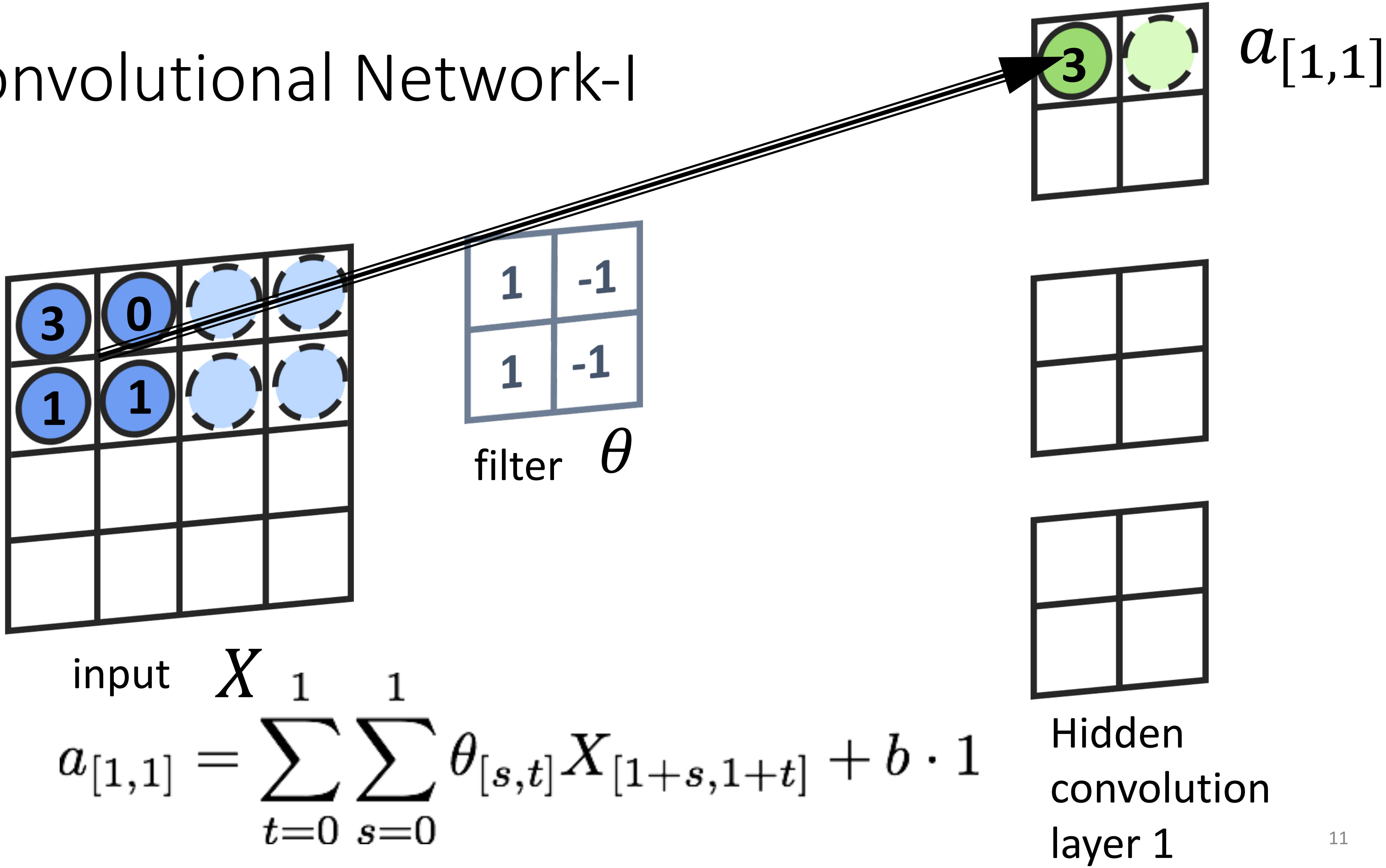
$$\vec{a}^{(2)} = \sigma\left(\overleftrightarrow{w}^{(2)} \cdot \vec{a}^{(1)} + \vec{b}^{(2)}\right)$$

$$\vec{a}^{(h)} = \sigma\left(\overleftrightarrow{w}^{(h)} \cdot \vec{a}^{(h-1)} + \vec{b}^{(h)}\right)$$

Neural Network – fully connected network



Convolutional Network-I

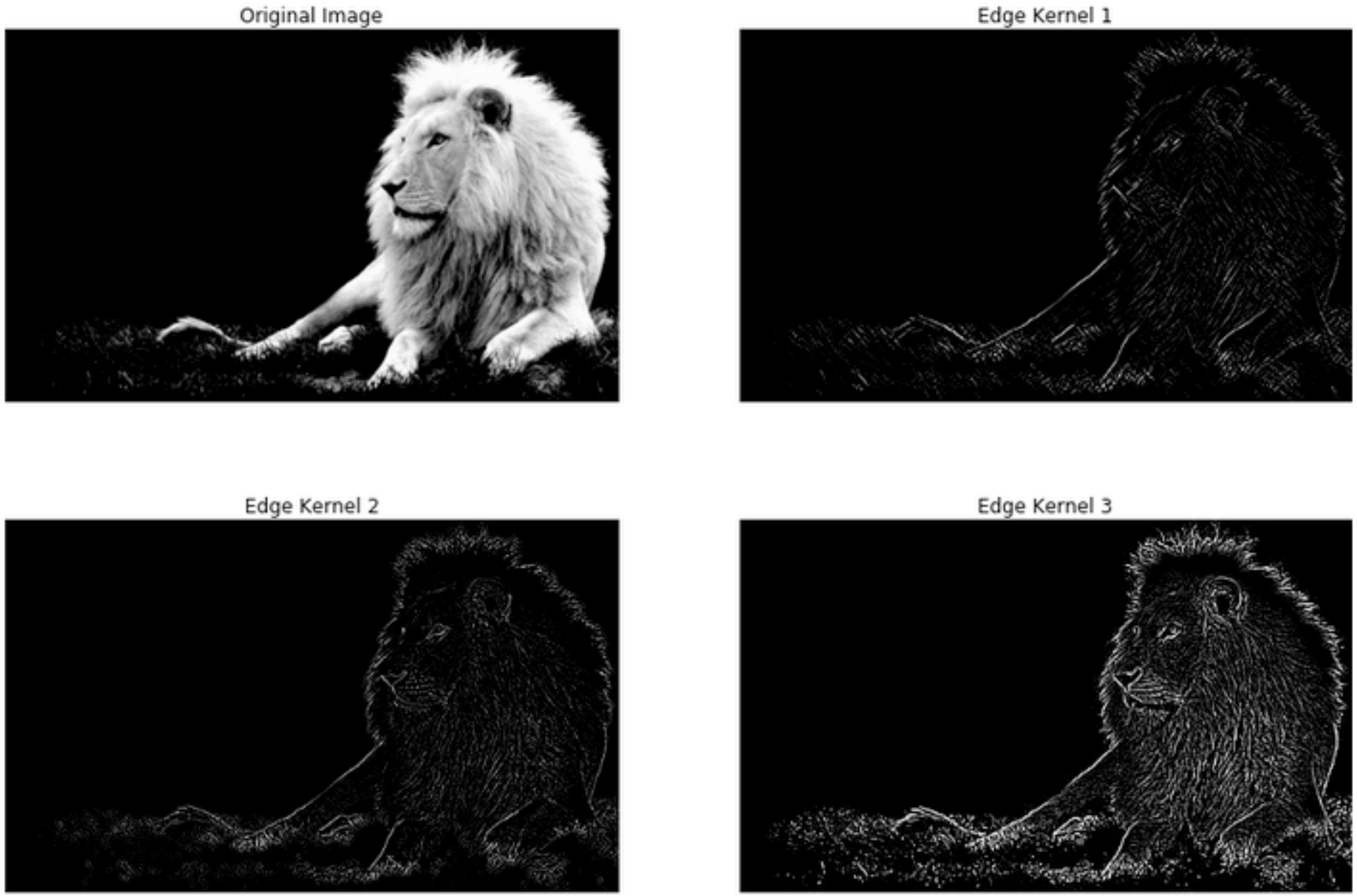


$$a_{[1,1]} = \sum_{t=0}^{1} \sum_{s=0}^{1} \theta_{[s,t]} X_{[1+s,1+t]} + b \cdot 1$$

Convolution

- Convolution can do edge detection, blurring, sharpening and etc.

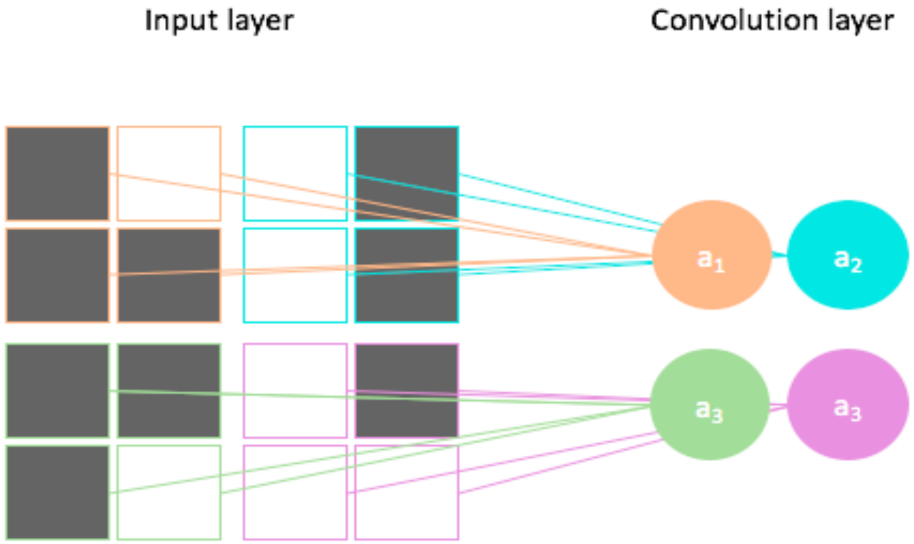
```
edge_kernel_1 = np.array([[1, 0, -1],
                          [0, 0, 0],
                          [-1, 0, 1]])

edge_kernel_2 = np.array([[0, 1, 0],
                          [1, -4, 1],
                          [0, 1, 0]])

edge_kernel_3 = np.array([[-1, -1, -1],
                          [-1, 8, -1],
                          [-1, -1, -1]])
```

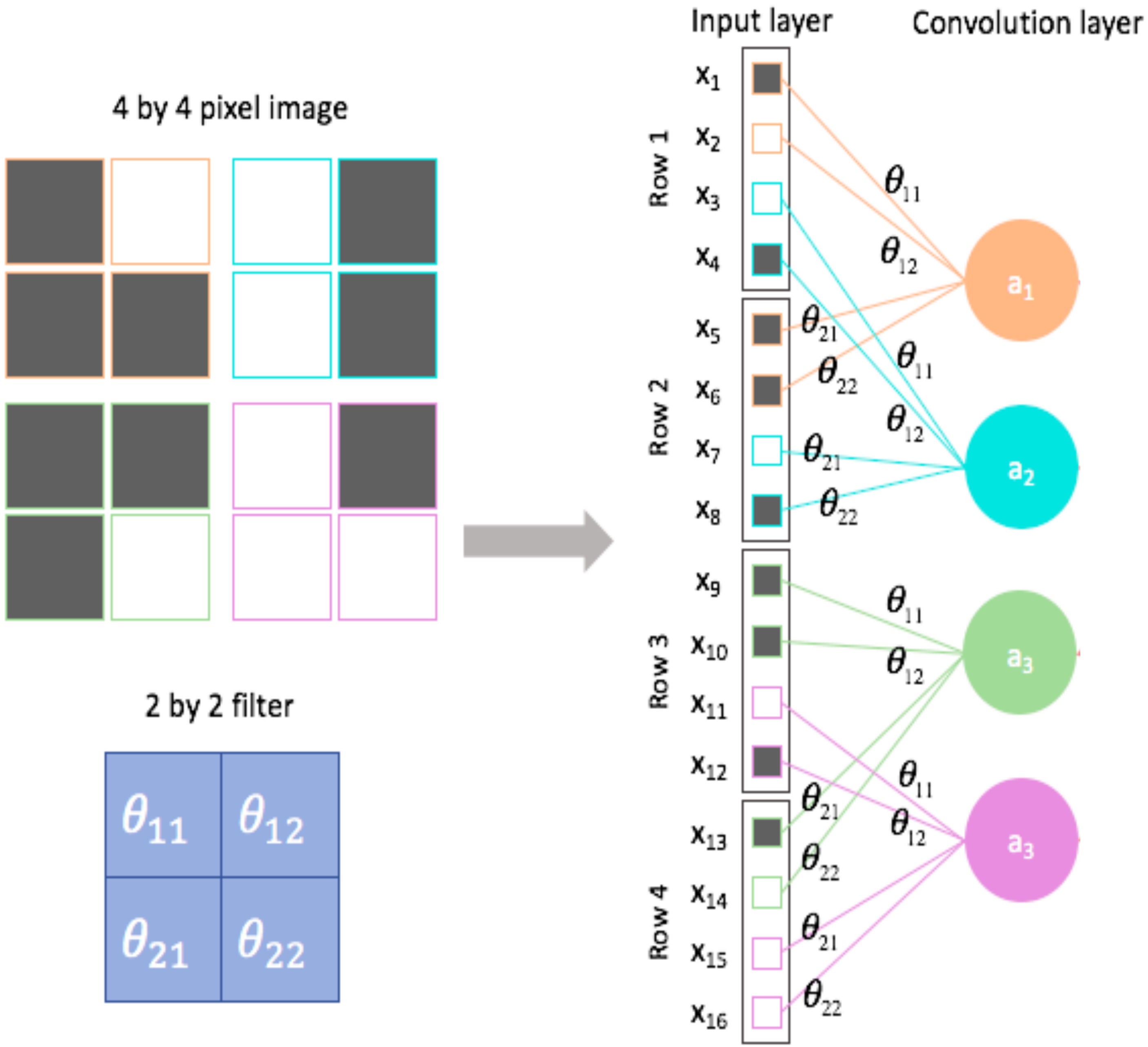


https://beckernick.github.io/convolutions/

Conv Net-II

1. Locality is preserved with convolution
2. Sparsely connected NN





https://www.jeremyjordan.me/convolutional-neural-networks/

The approaches

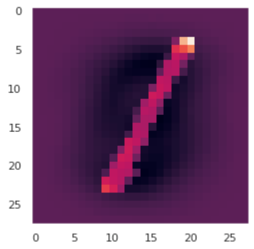
1. Visualize the intermediate activations
2. CNN encoder and interpreters
3. Did my CNN learn translational symmetry?

Visualize the intermediate activations-I

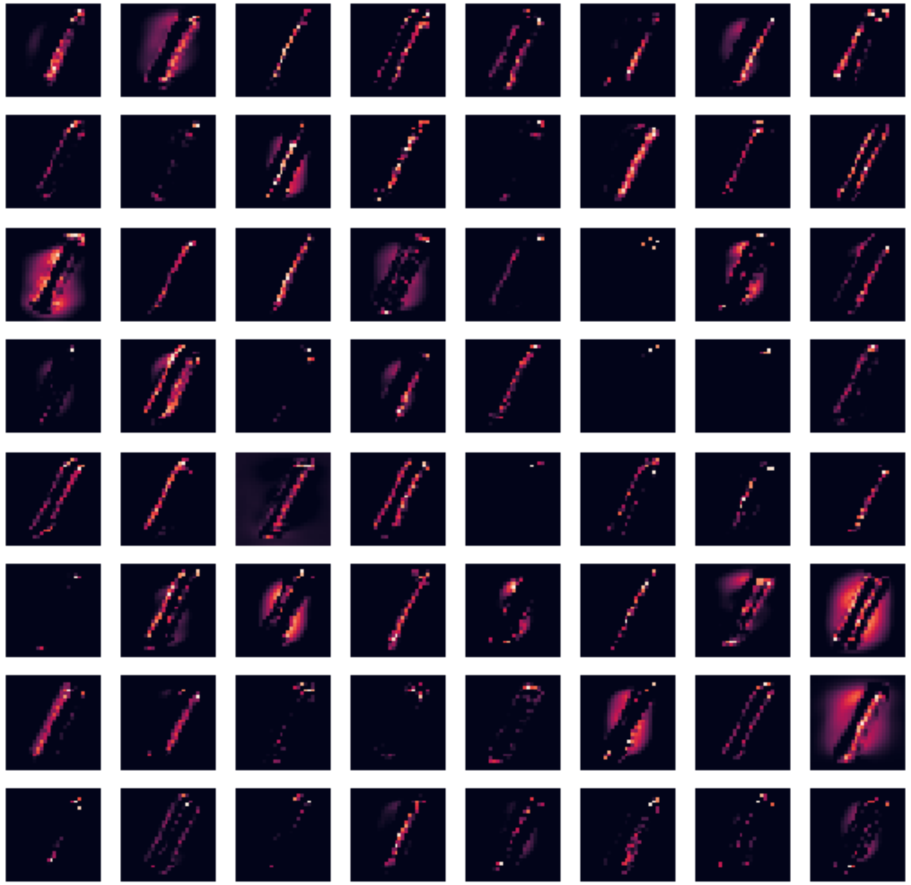
h:hiddne layer index

conv2d_1 $h = 1$

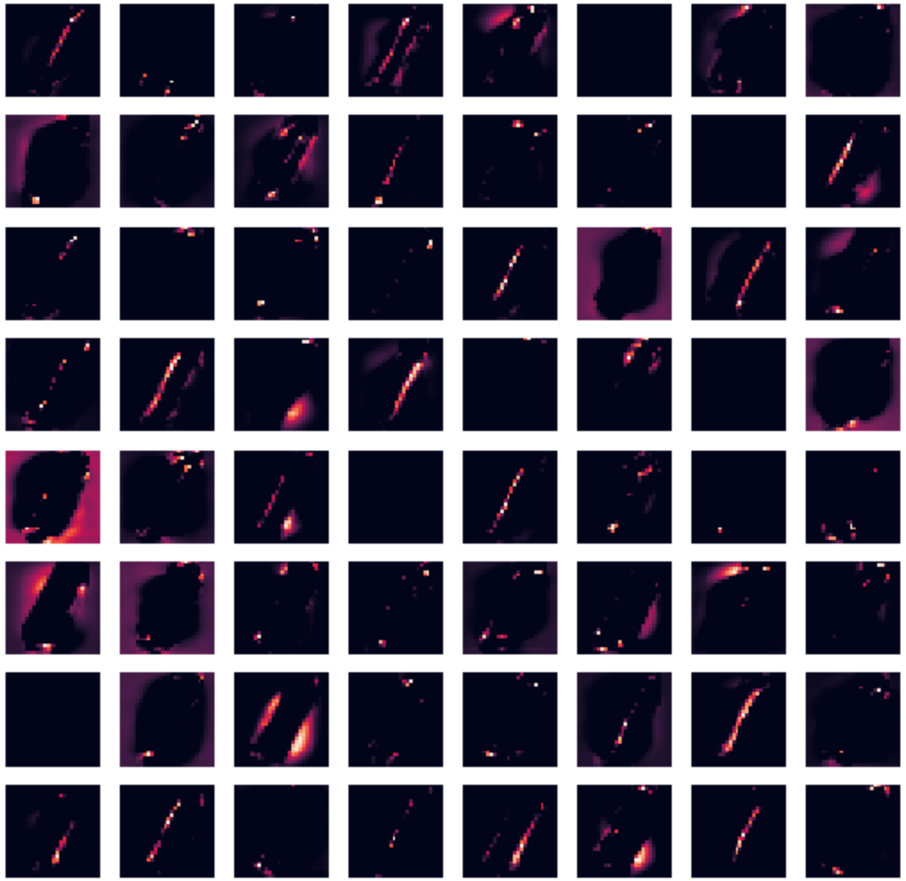
conv2d_2 $h = 2$



28*28=784



(28*28)*64=50,176



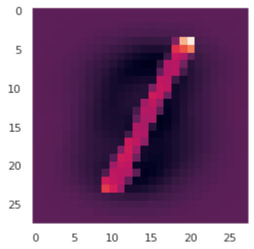
(28*28)*64=50,176

Visualize the intermediate activations-II

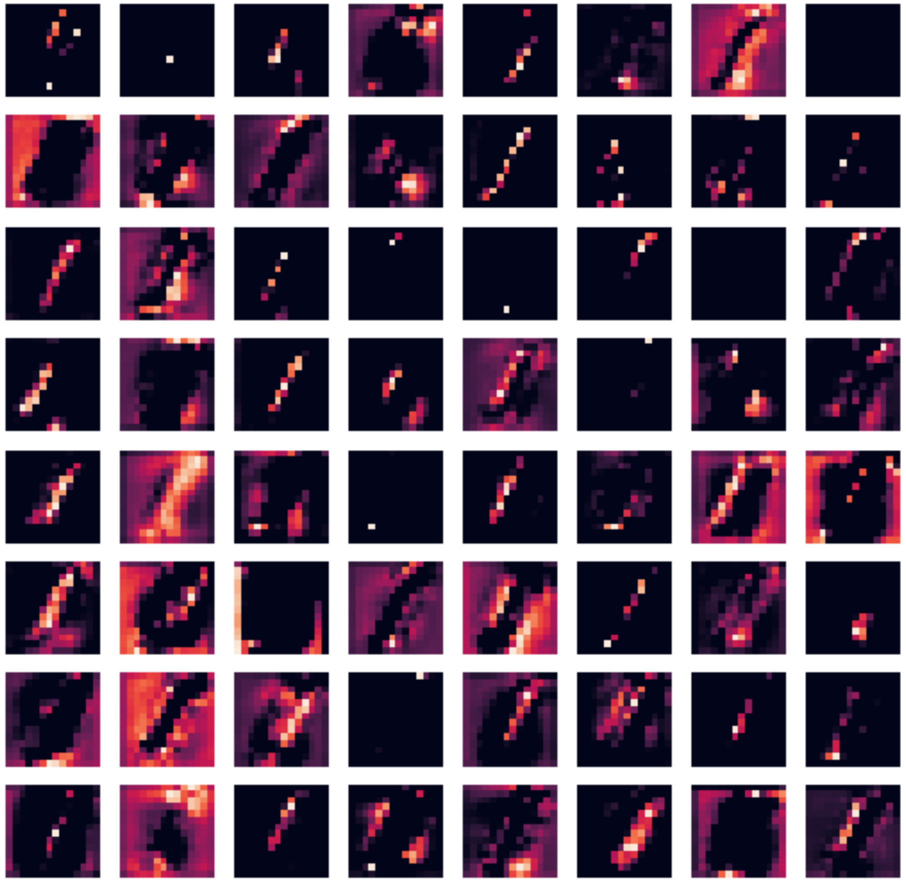
h:hiddne layer index

conv2d_3 $h = 5$

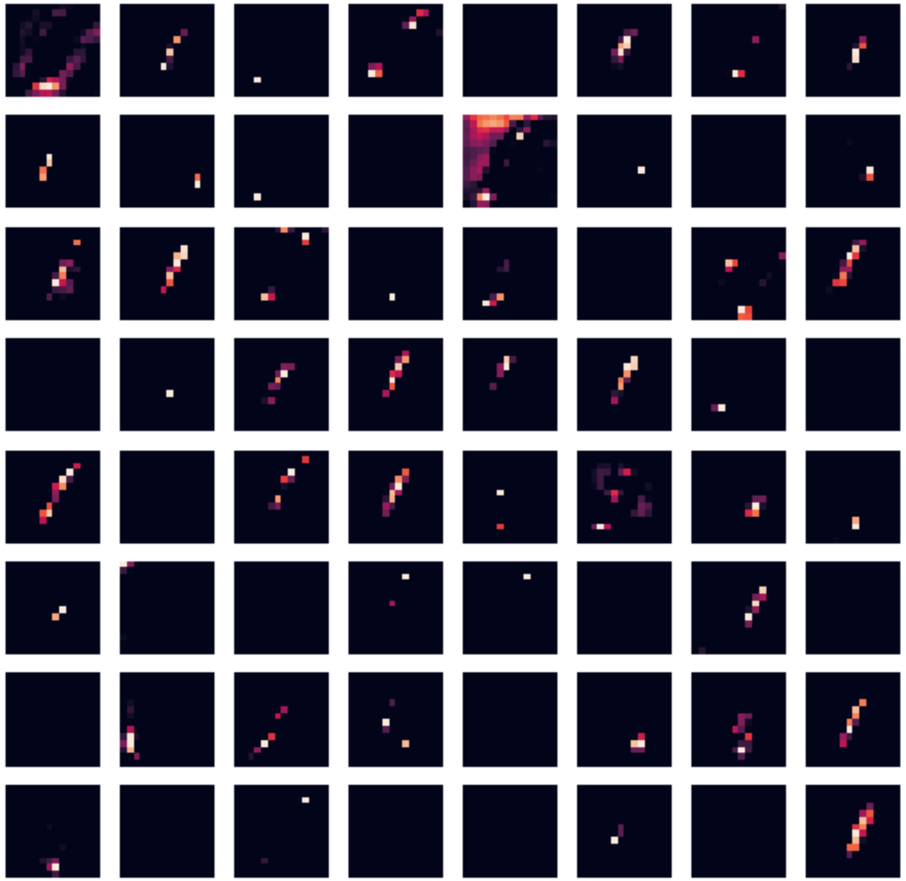
conv2d_4 $h = 6$



28*28=784



(14*14)*64=12,544

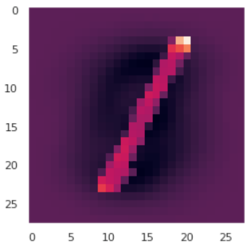


(14*14)*64=12,544

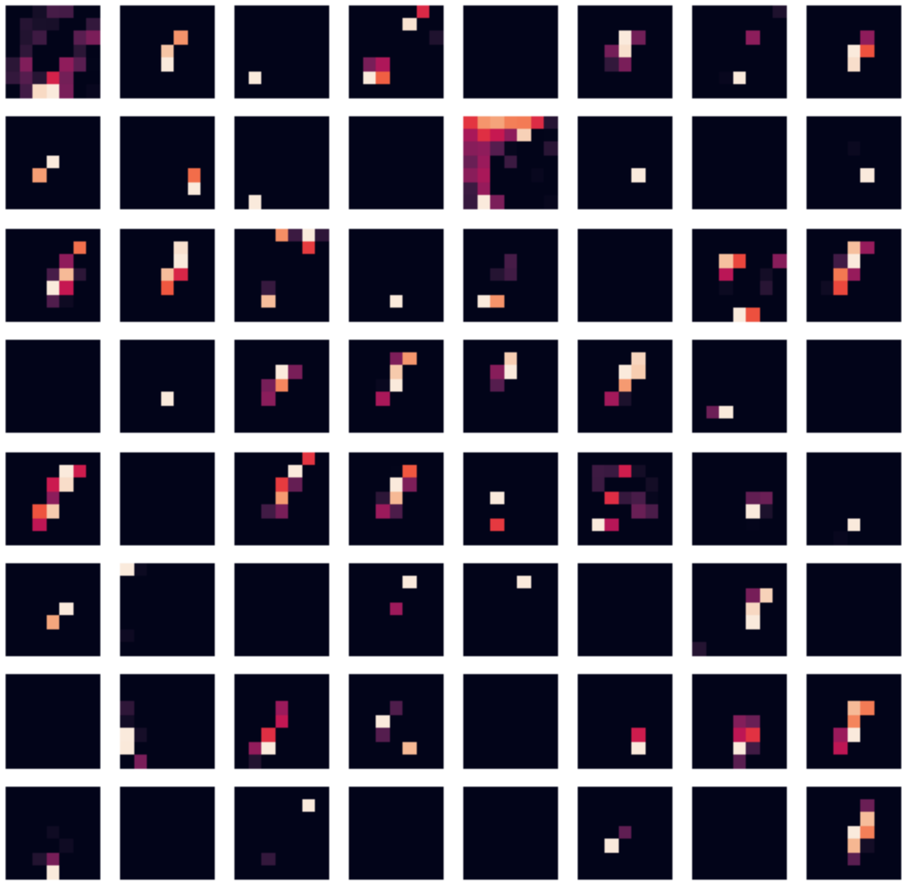
Visualize the intermediate activations-III

h:hiddne layer index

dropout_2 $h = 8$

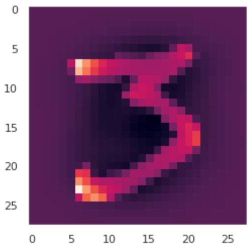


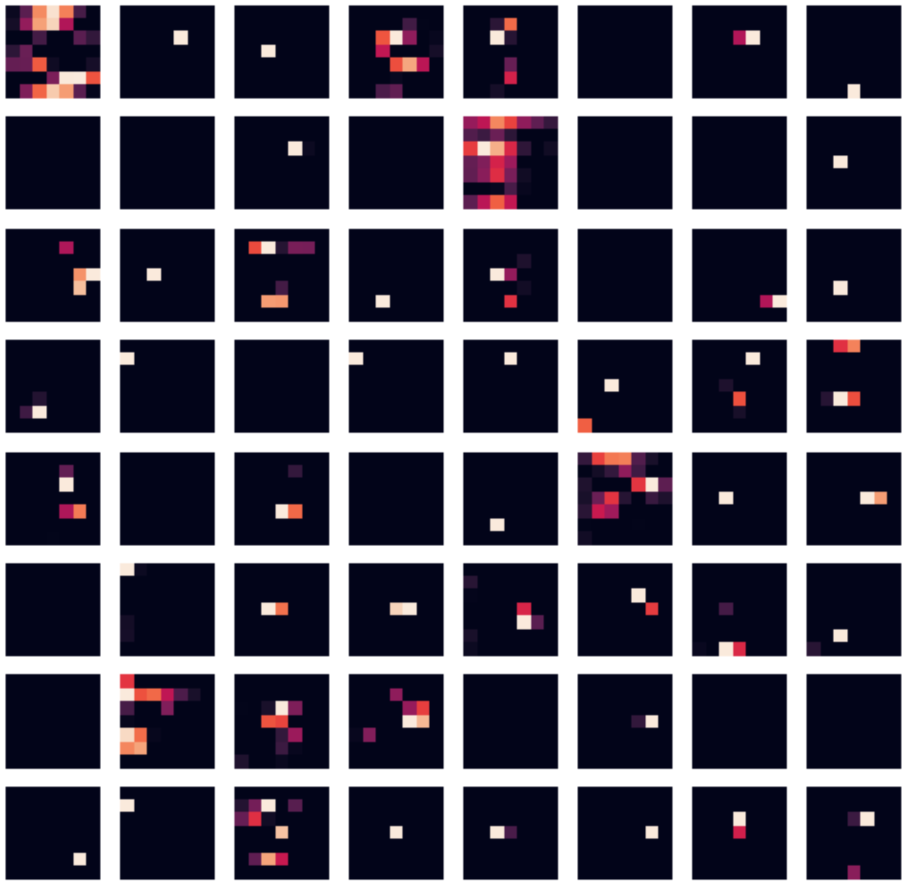
28*28=784



(7*7)*64=3,136

dropout_2 $h = 8$





(7*7)*64=3,136

CNN encoder and interpreters

the architecture

```
#The model structure
my_CNN_clf=Sequential([
    ############################## CNN-encoder
    Conv2D(filters = 64, activation = 'relu',kernel_size=(5,5),padding='Same',input_shape = (28,28,1)),
    Conv2D(filters = 64, activation = 'relu',kernel_size=(5,5),padding='Same'),
    MaxPool2D(pool_size=(2,2),strides=(2,2)),
    Dropout(0.25),
    Conv2D(filters = 64, activation = 'relu',kernel_size=(3,3),padding='Same'),
    Conv2D(filters = 64, activation = 'relu',kernel_size=(3,3),padding='Same'),
    MaxPool2D(pool_size=(2,2),strides=(2,2)),
    Dropout(0.25),
    Flatten(), # d=64*7*7=3136
    ############################## CNN-encoder

    ##################           compressor
    Dense(256, activation='relu'),
    Dropout(0.5), #d = 256
    ##################           compressor

    ############################## interpretor
    Dense(10, activation='softmax'), #a normalized exponential functions for probability
    ############################## interpretor
    ])
```

$h = 1$
$h = 2$

$h = 5$
$h = 6$

$h = 8$

Encoder: input→ 3,136-dimensional vector

$h = 10$

Compressor: 3,136 → 256 dimensional

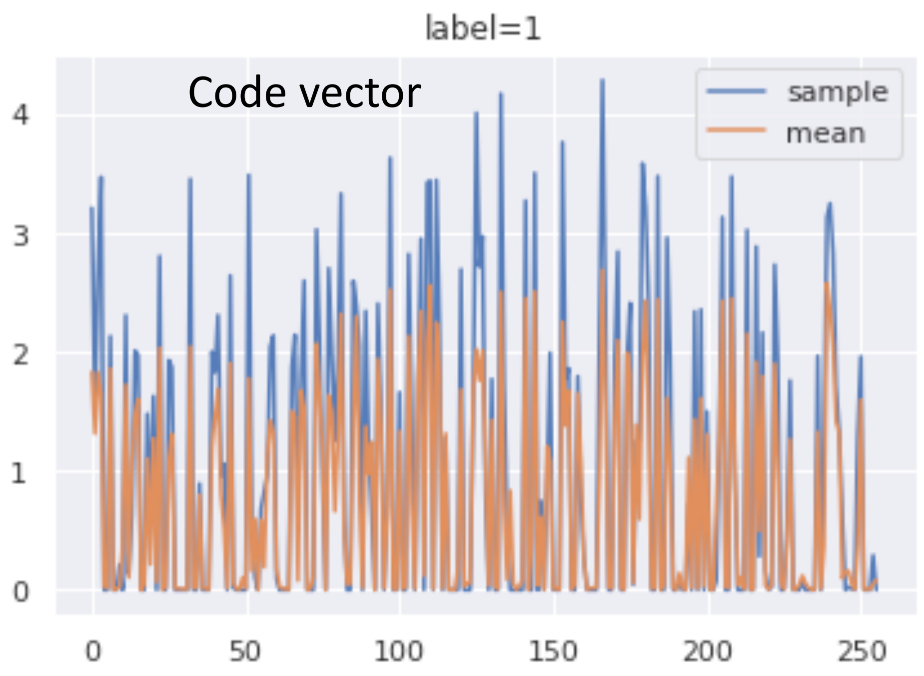
The code vector

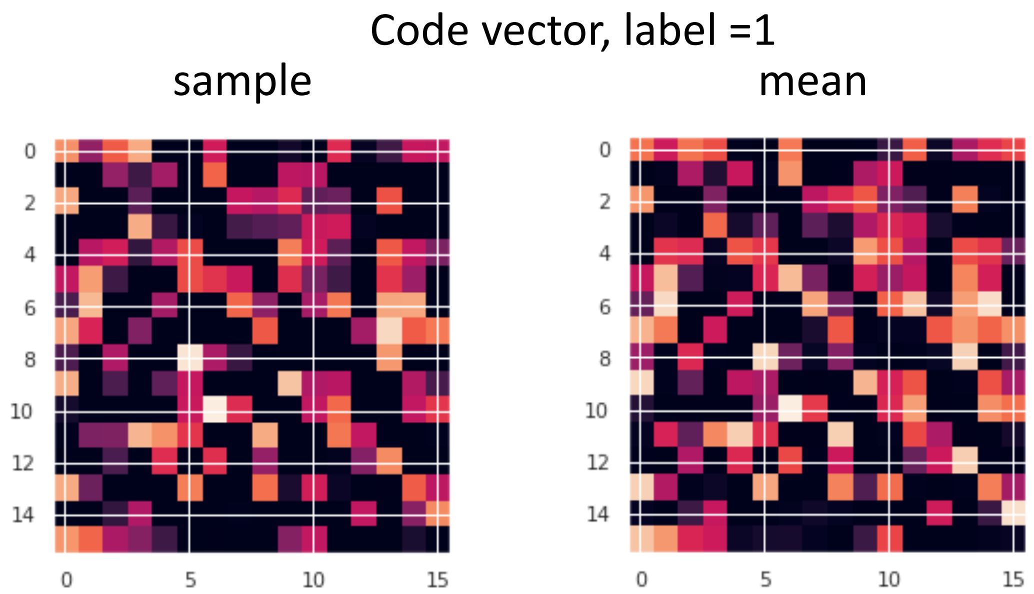
Interpreter: 256 → 10 dim. probability vector for predictions

CNN encoder and interpreters
the code vectors

1. Consistent and self-similarity within a class(both code vectors and probabilities)
2. Different interpreters show similar accuracy
   => encoder is the key for acc.

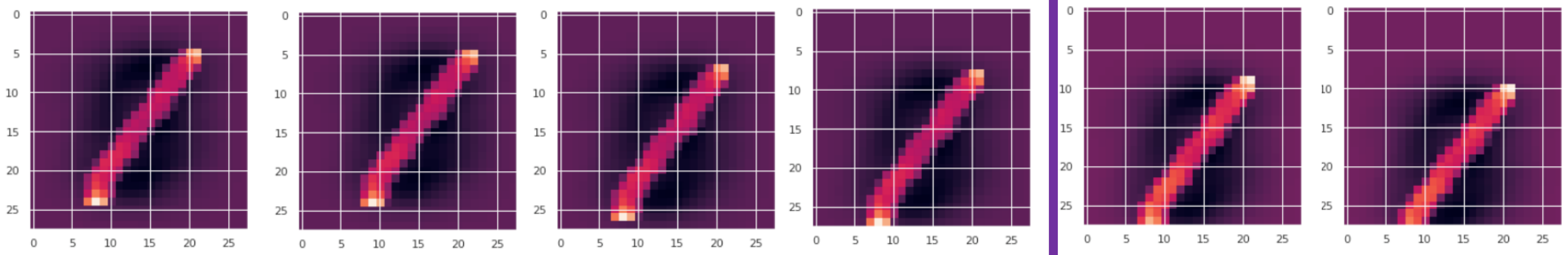
Interpreters	accuracy
1 Dense layer NN	99.44%
SVM	99.35%
Random Forests (100 trees)	99.32%
Gradient Boost (100 trees)	99.24%
Decision Tree	98.46%





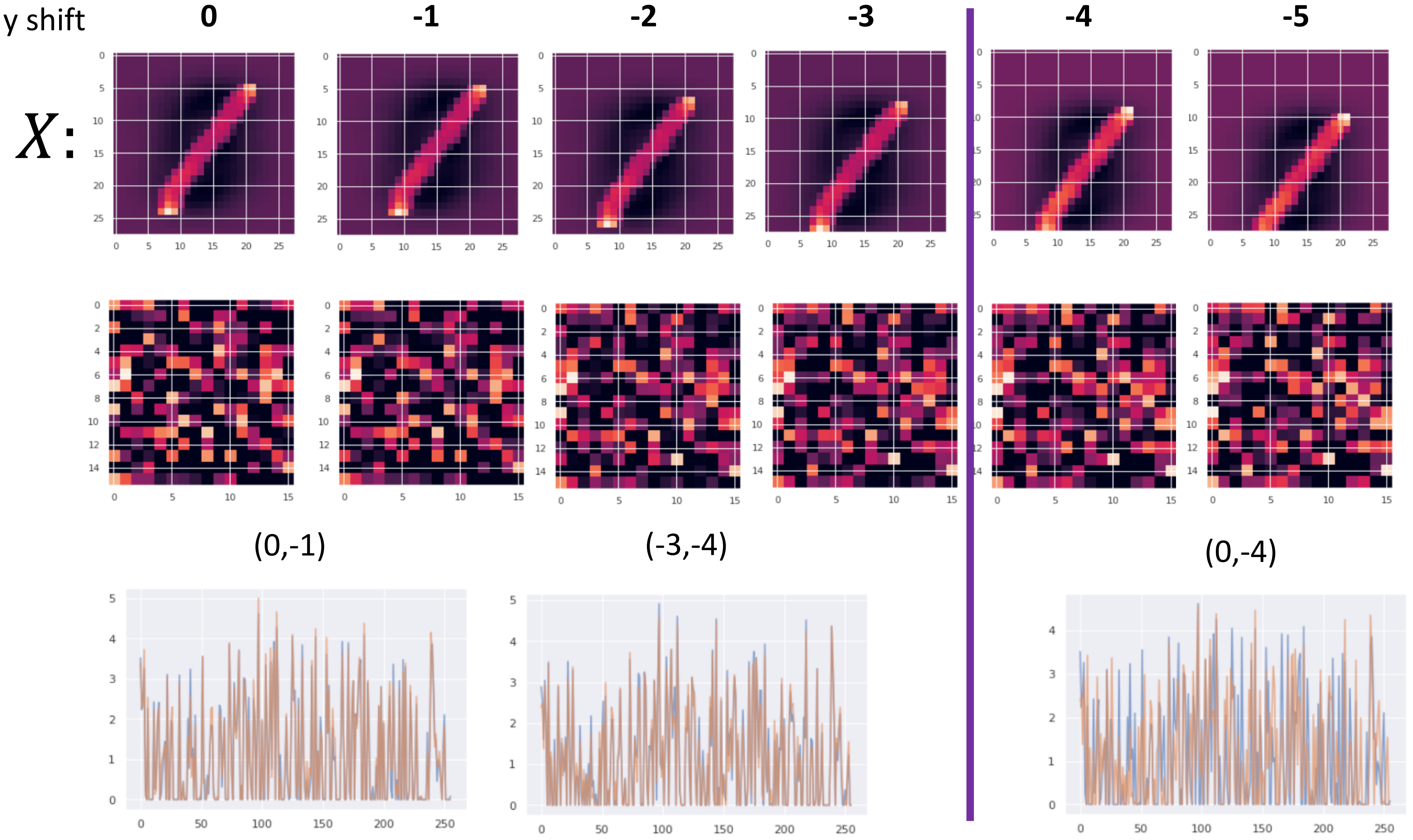
Did my CNN learn translational symmetry?

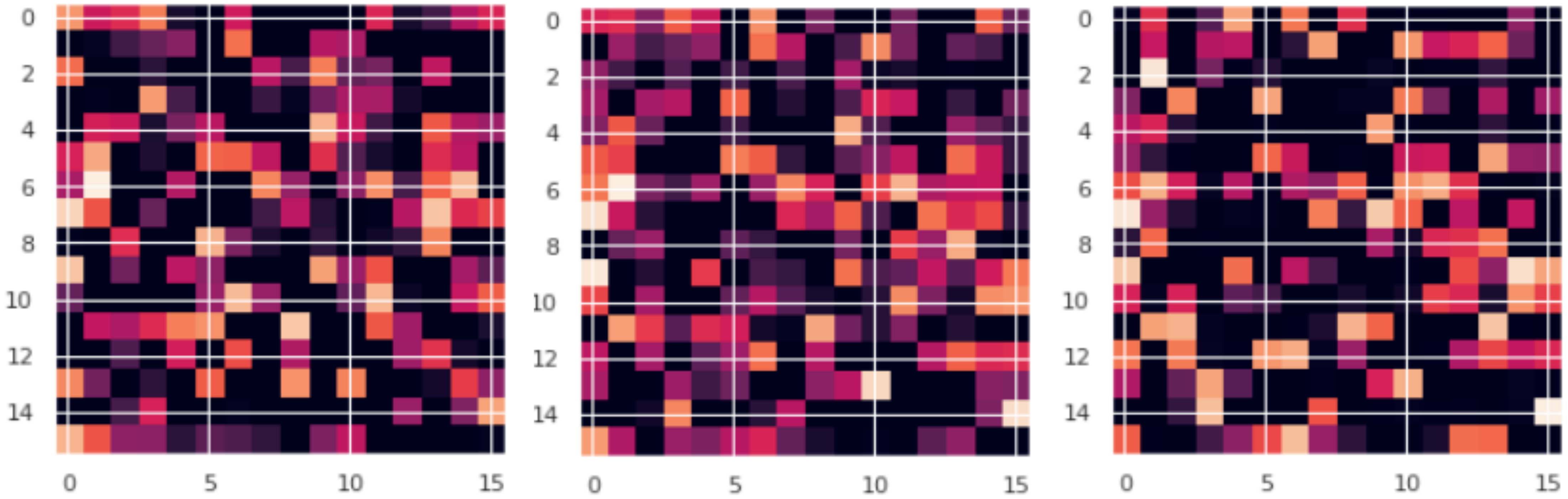
$Prob(y = 1|X)$ X:



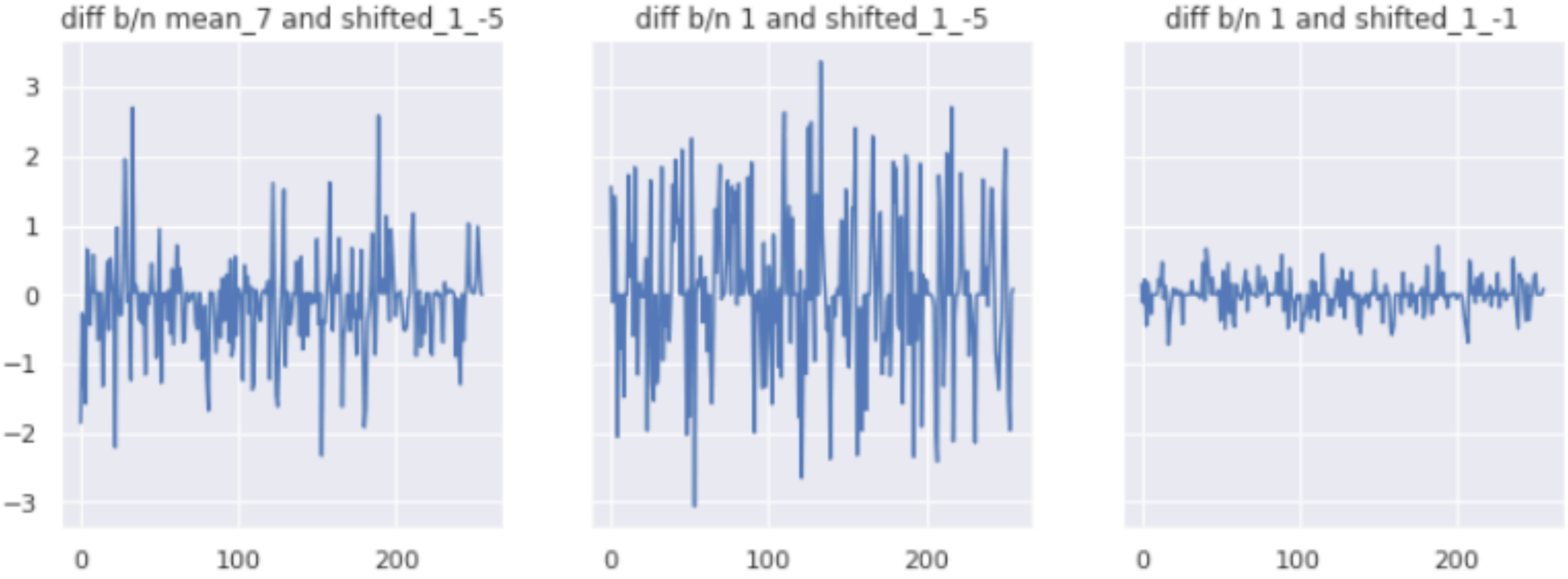
y shift (pixels)	0	-1	-2	-3	-4	-5
CNN (CNN+NN) (acc = 99.3%)	100%	100%	99.99%	99.4%	0.71% (7: 99.29%)	1.4E-4% (7: 99.99%)
Random Forest (RF) (acc = 96.6%)	99.5%	84.9% (7: 9.8%)	34.1% (7: 39.6%)	9.1% (7: 56.3%)	5.4% (7: 70.1%)	4.8% (7: 70.2 %)
SVM (acc = 96.1%)	99.6%	42.8% (7: 50.8%)	0.85% (7: 94.5%)	0.42% (7: 81.3 %)	0.43% (7: 79.5%)	0.42% (80.8%)
CNN+RF, T=1000 (acc = 99.3)	100%	96.6%	67.7% (7: 18.4)	52.8% (7: 28.3%)	44.4% (7: 35.5%)	34.8% (7: 47.6 %)
CNN+SVM (acc = 99.3%)	99.9%	99.8%	97.8%	79.9% (9: 18.8%)	21% (9: 77.76%)	1% (9: 98.9%)

Did my CNN learn translational symmetry?





0
2
4
6
8
10
12
14
0
5
10
15


diff b/n mean_7 and shifted_1_-5
diff b/n 1 and shifted_1_-5
diff b/n 1 and shifted_1_-1
3
2
1
0
−1
−2
−3
0
100
200

Conclusion

- Code vectors in hidden layers of CNN reduce the dimensionality of images.
- Information learned in CNN is **position sensitive**. And convolution is the key for high accuracy performance.
- The predicted probabilities of CNN are not continuous with respect to translations.
- Translations deform CNN code vectors more softly.