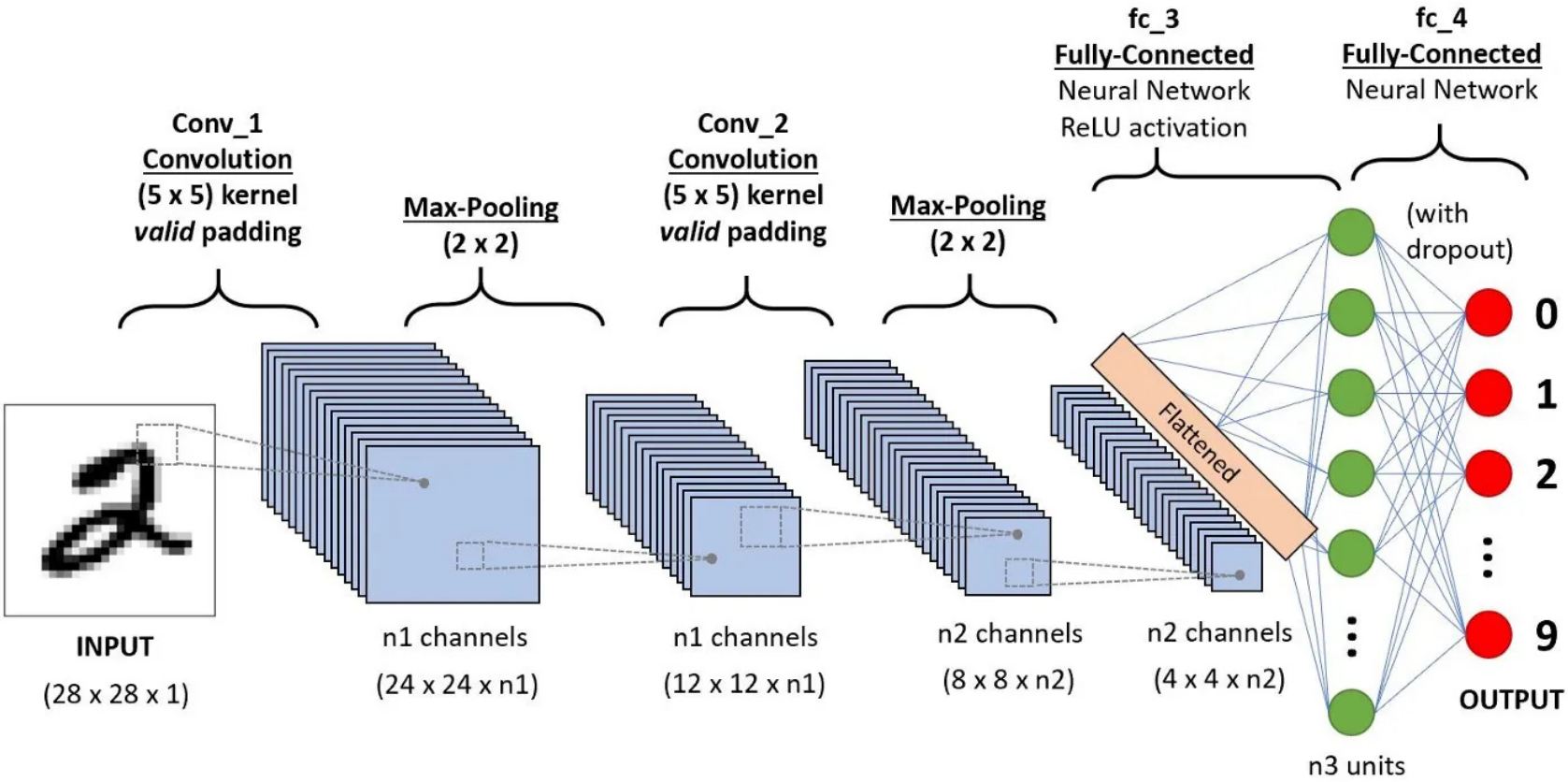
Lecture 11: Convolutional Neuron Networks (CNN)

Yen-Yi Ho

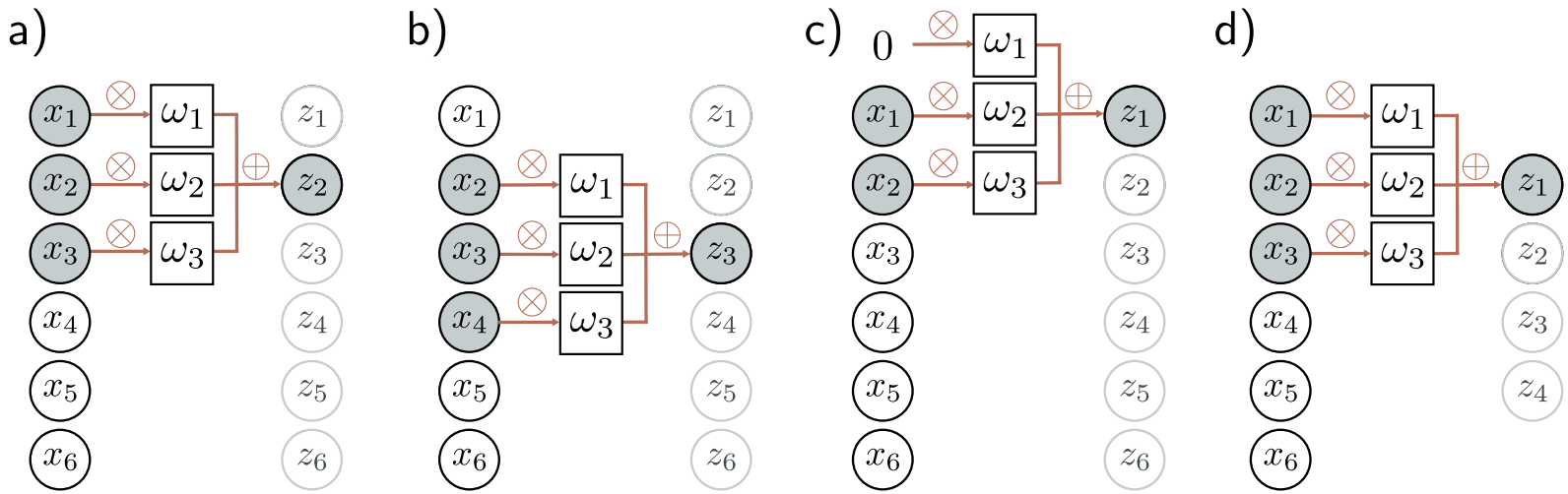
Department of Statistics

An Example of CNN

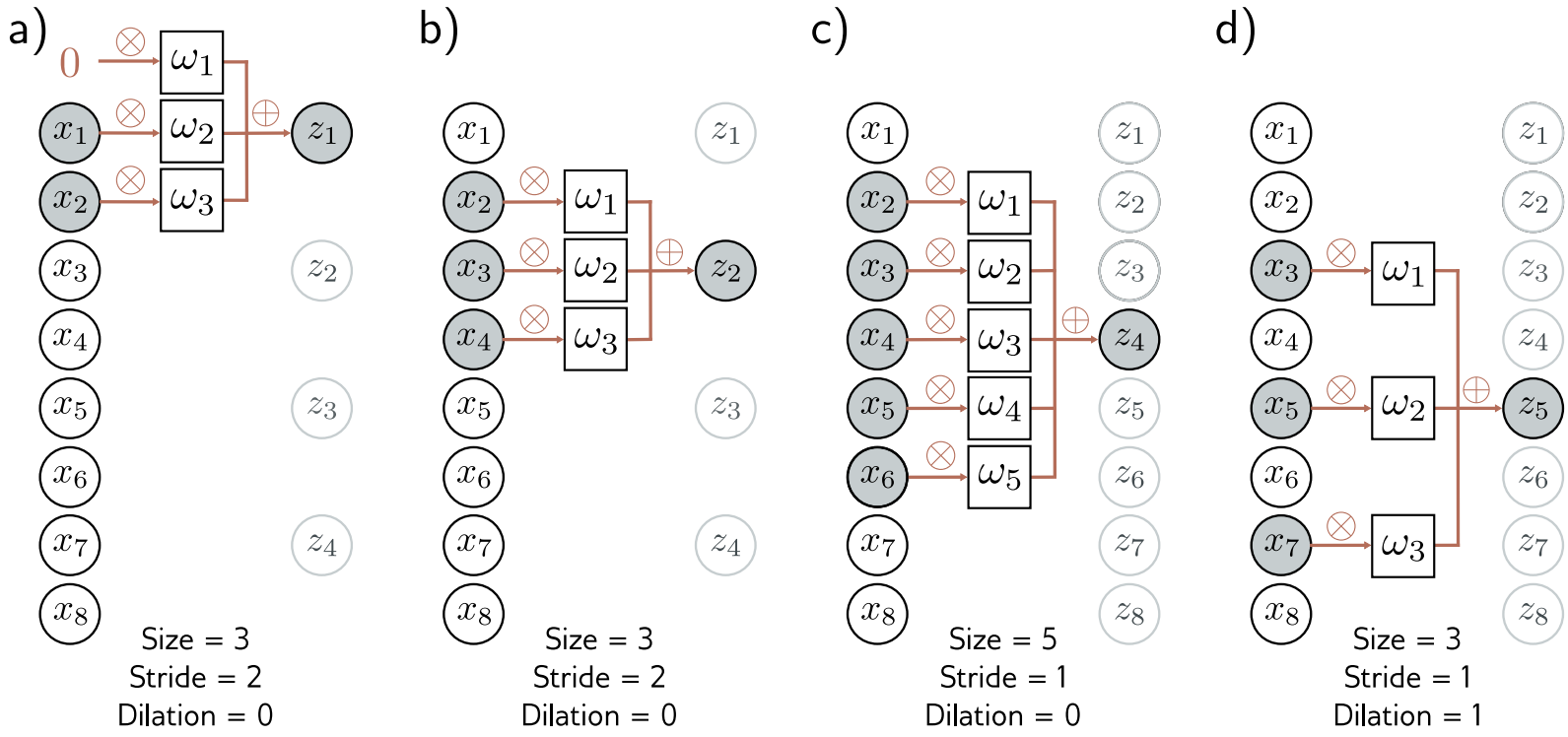


One-dimensional (1D) CNN

$$z_i = \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}$$



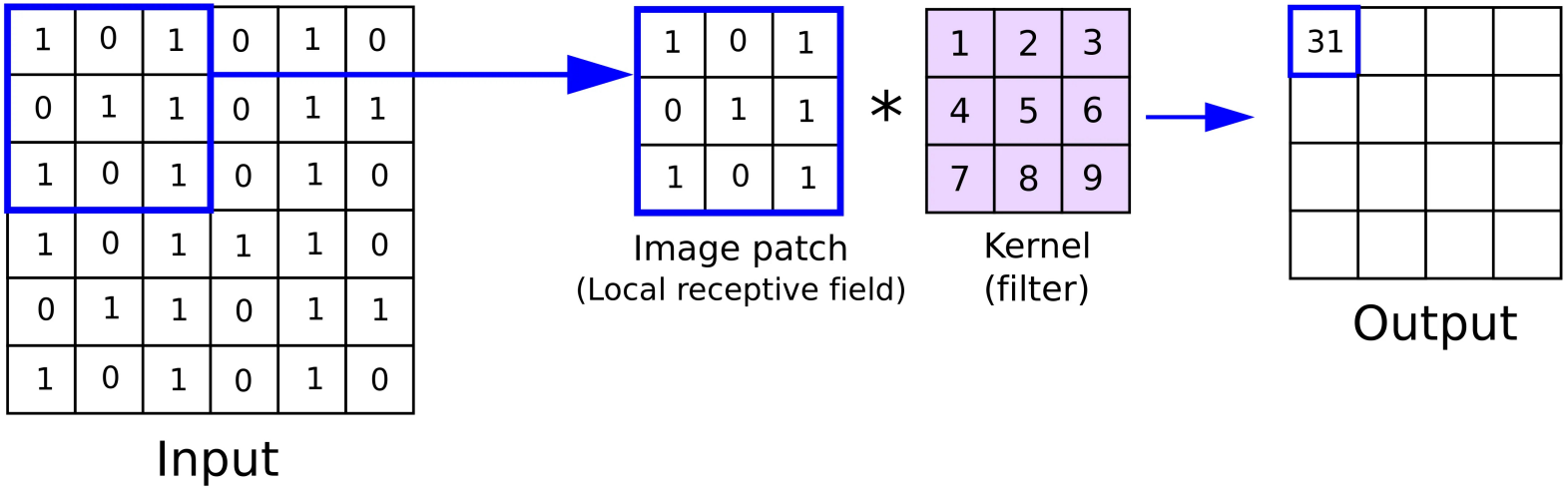
1D CNN

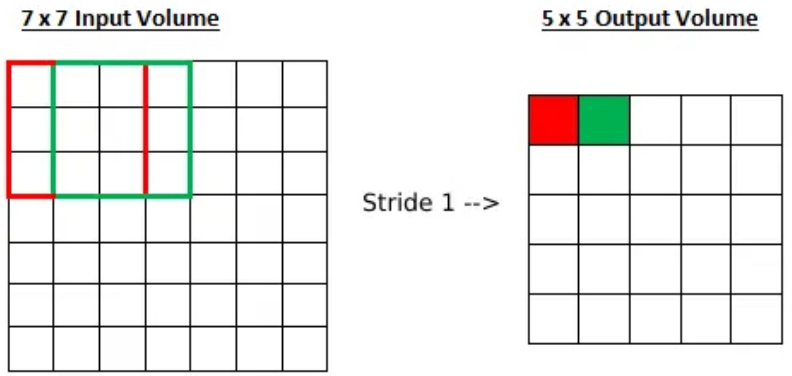

a)
b)
c)
d)
Size = 3
Stride = 2
Dilation = 0
Size = 3
Stride = 2
Dilation = 0
Size = 5
Stride = 1
Dilation = 0
Size = 3
Stride = 1
Dilation = 1

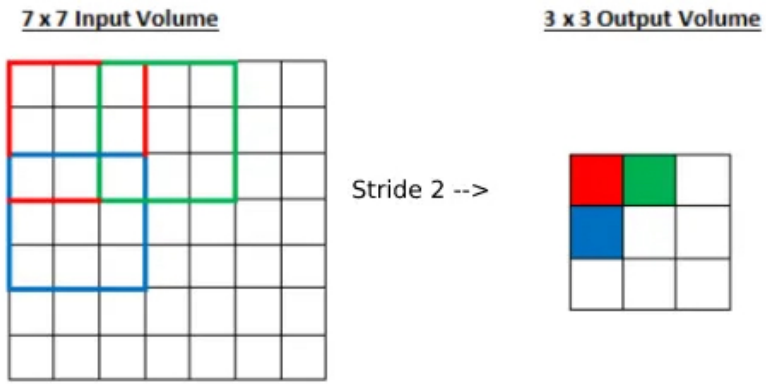
2D CNN

i input size, k kernel size, s stride

Output dimension= $[(i-k)/s]+1$

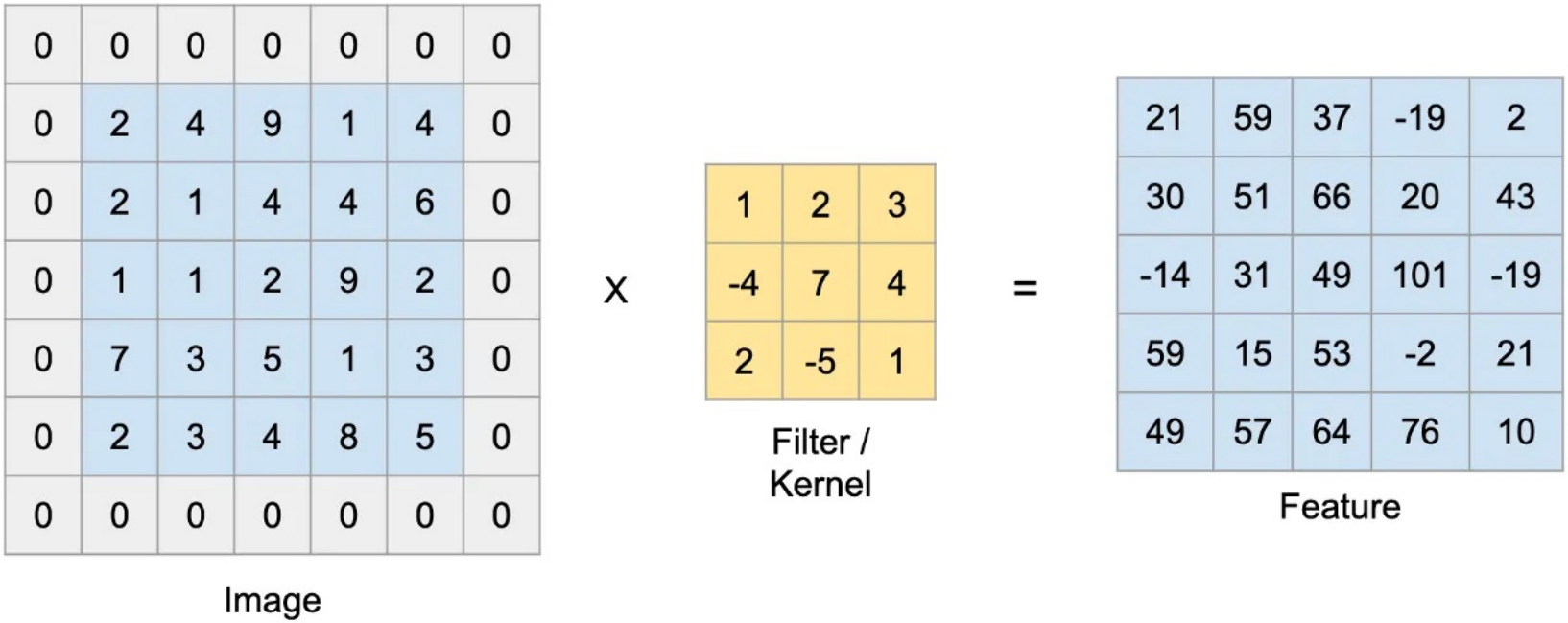




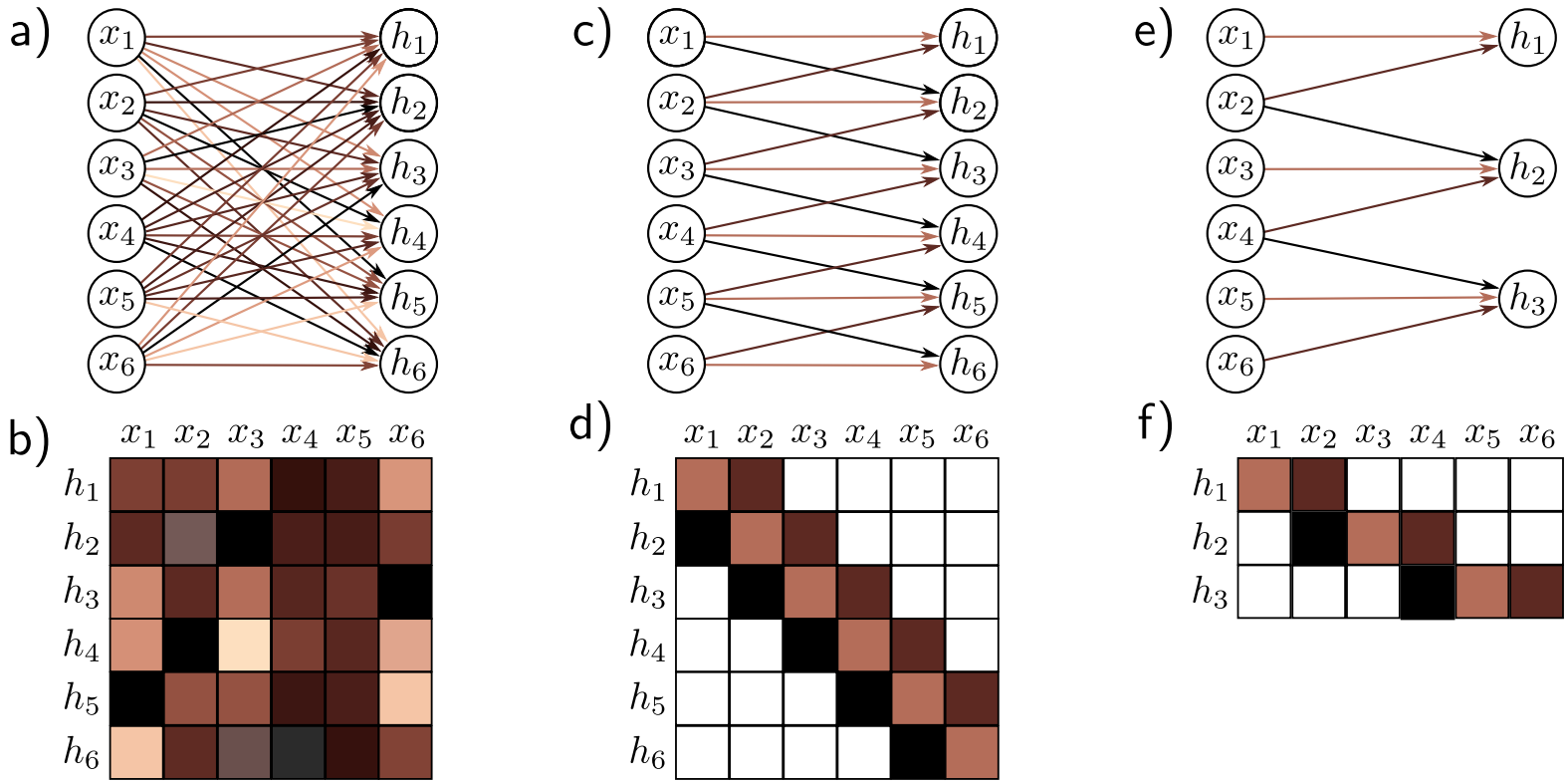


Padding

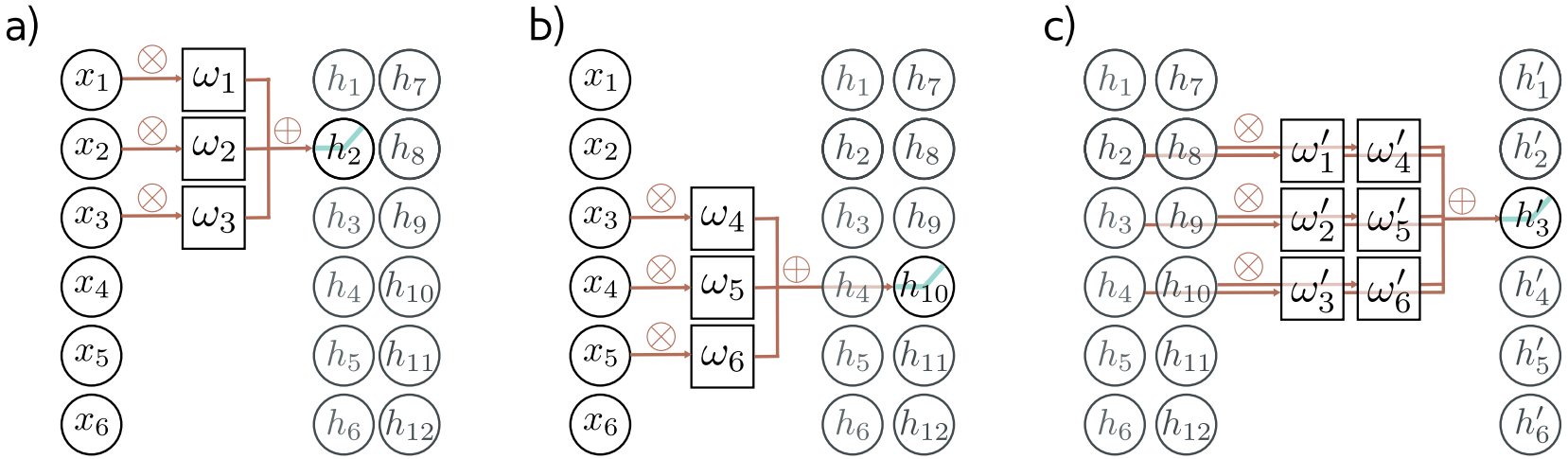
i input size, k kernel size, s stride, p padding
Output dimension= [(i-k + 2p)/s]+1



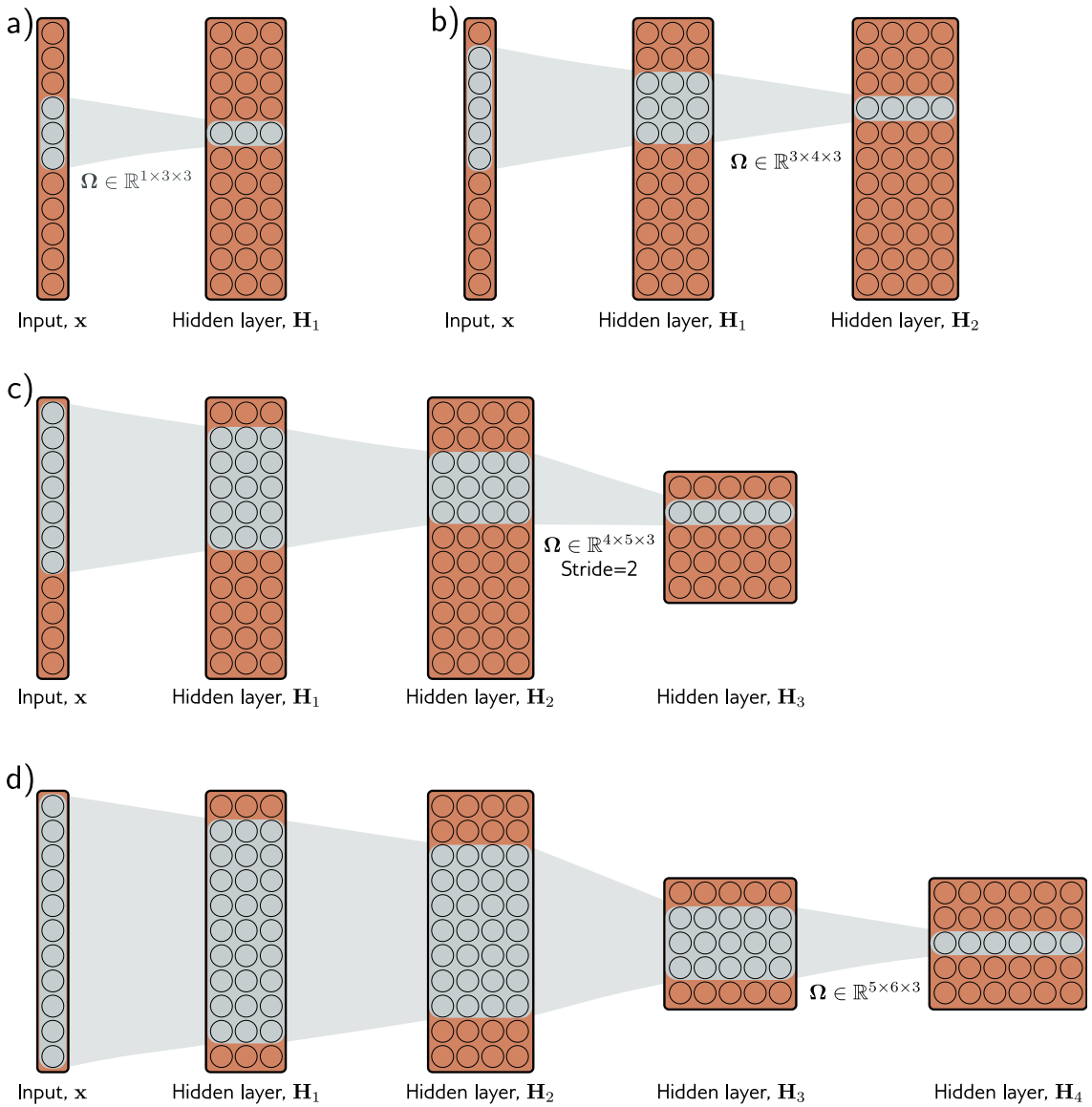
MLP versus CNN



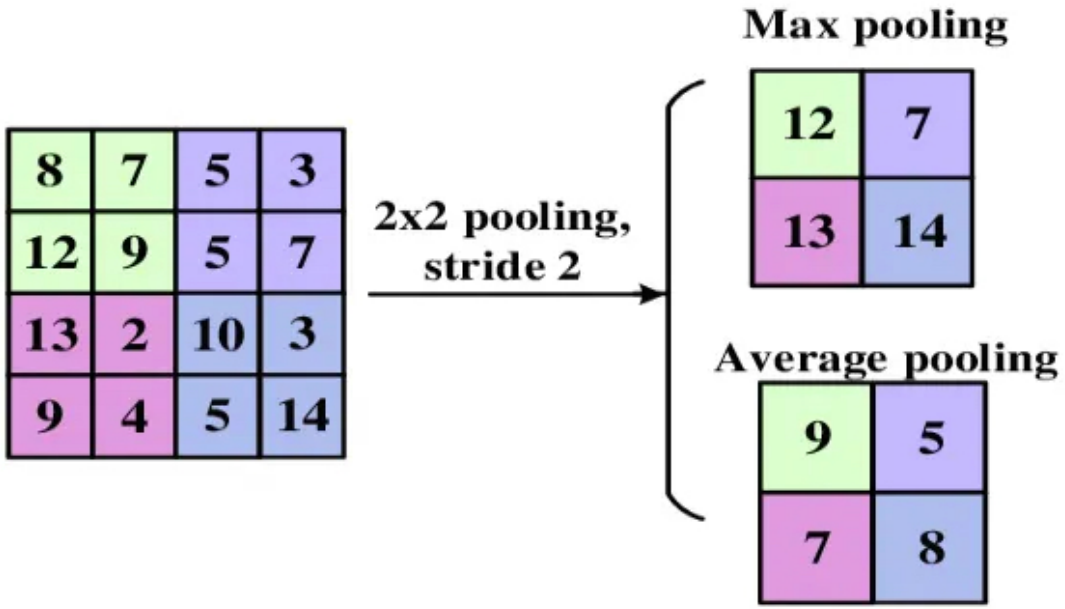
Multiple Channels



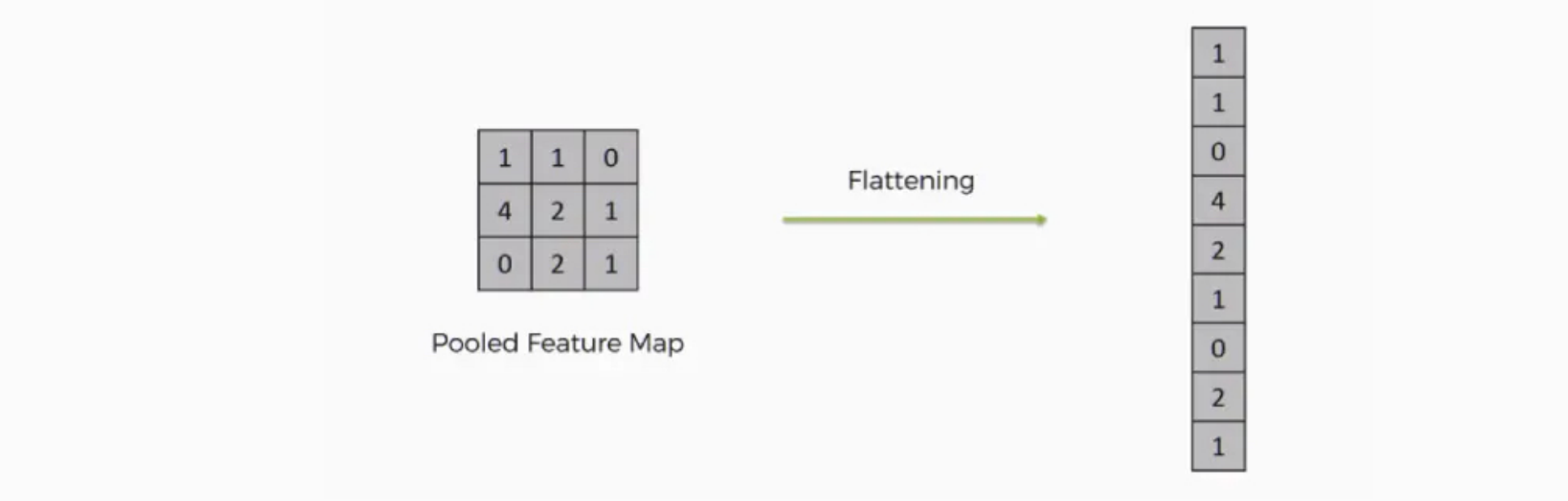
Receptive Fields



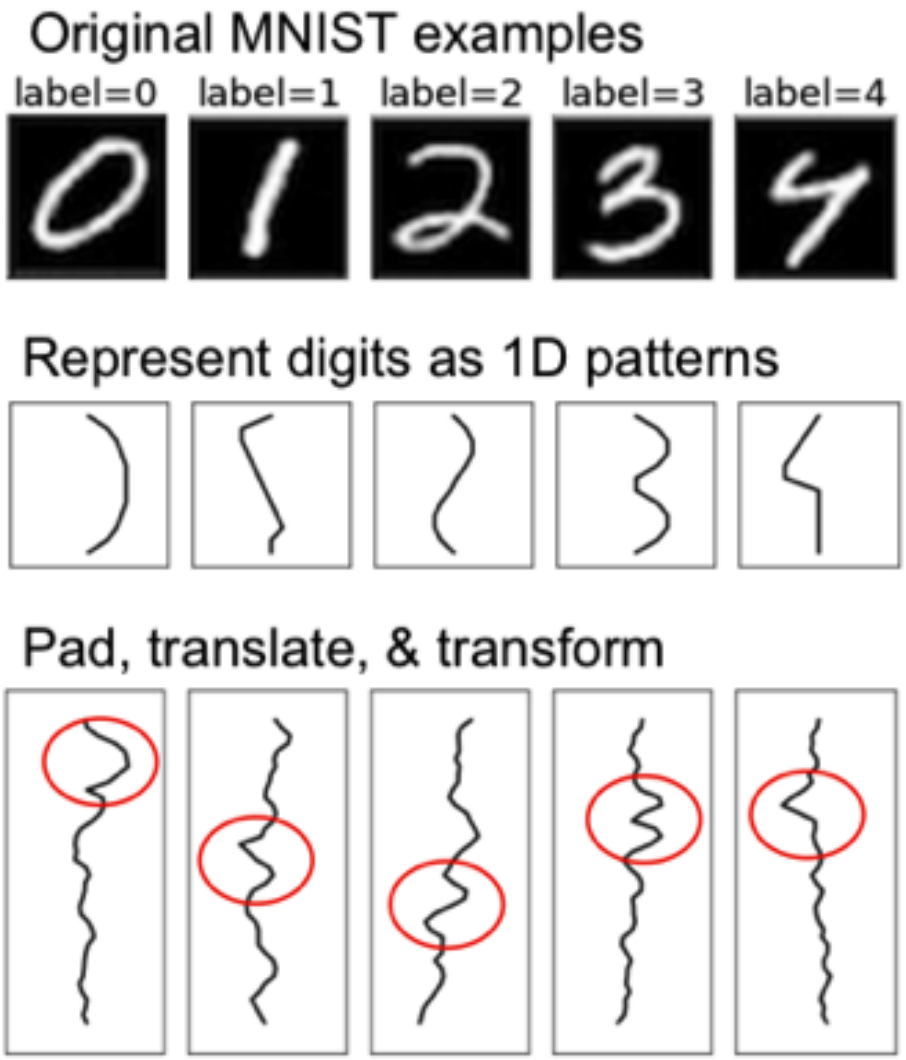
Max Pooling


8 7 5 3
12 9 5 7
13 2 10 3
9 4 5 14
2x2 pooling,
stride 2
Max pooling
12 7
13 14
Average pooling
9 5
7 8

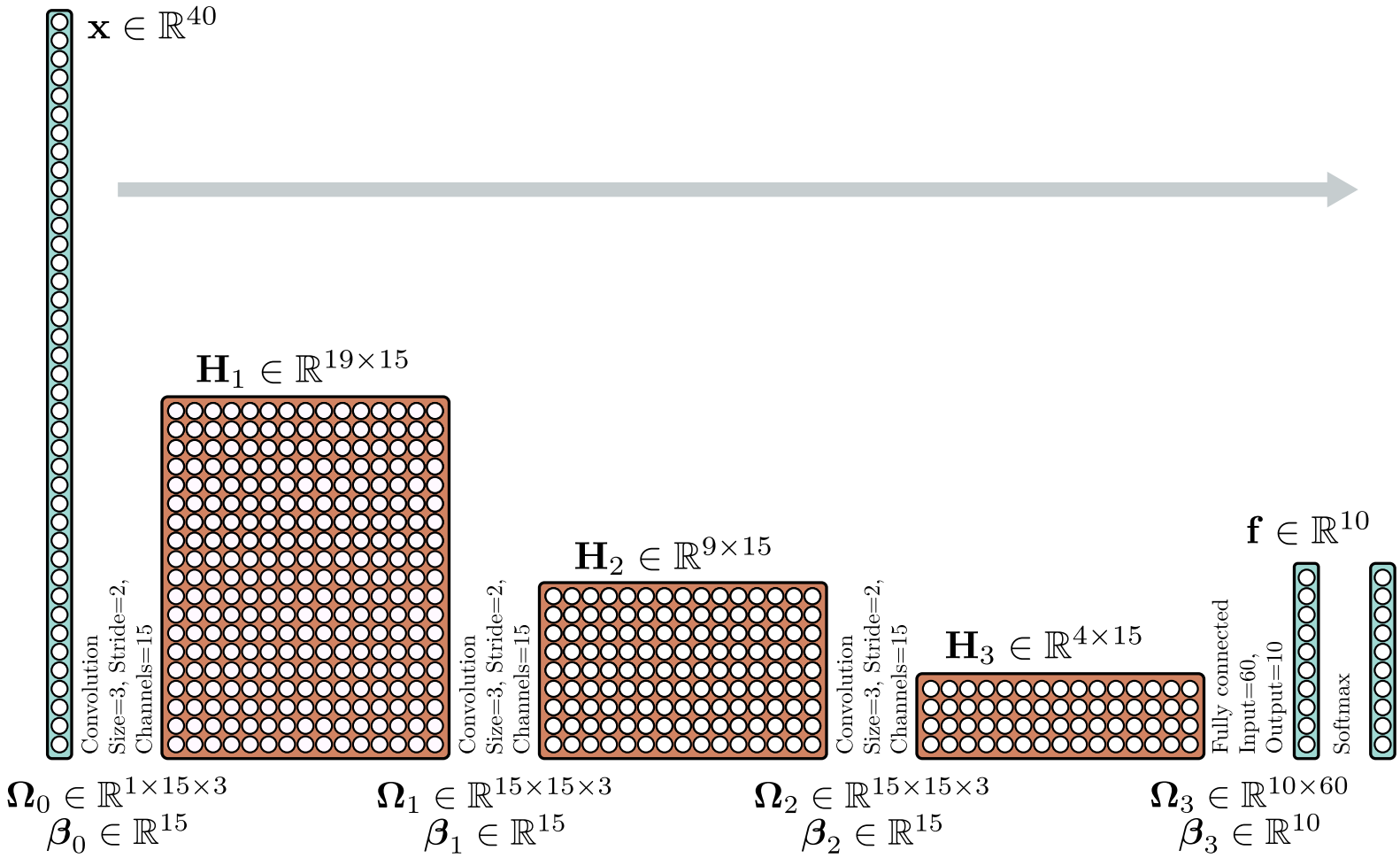
Flattening



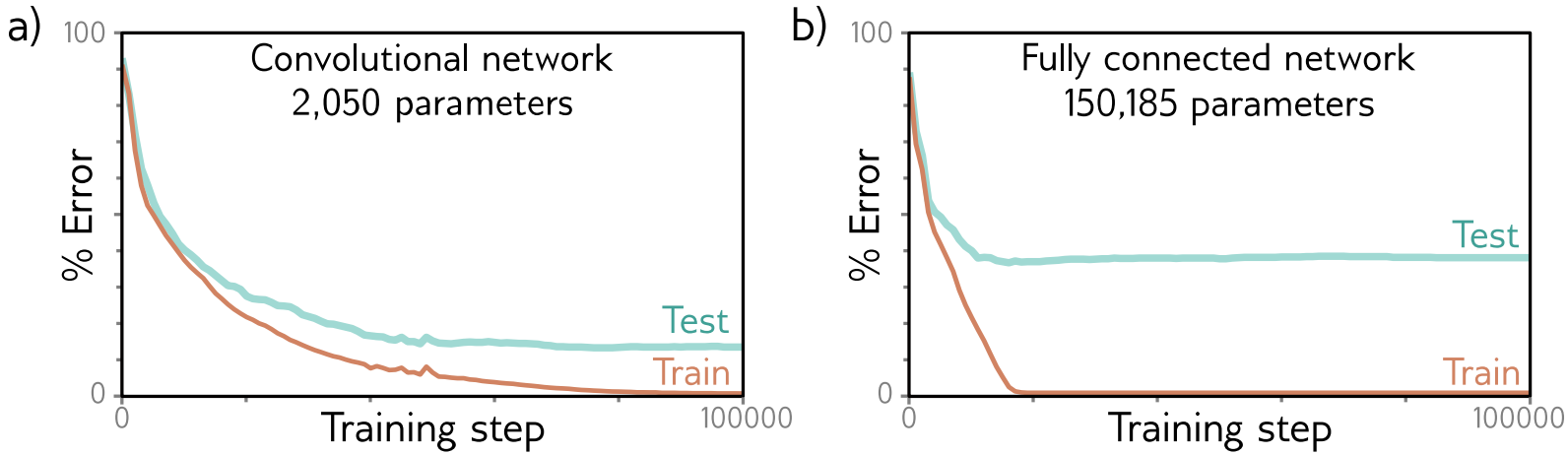
MNIST1D


Original MNIST examples
label=0
label=1
label=2
label=3
label=4
Represent digits as 1D patterns
Pad, translate, & transform

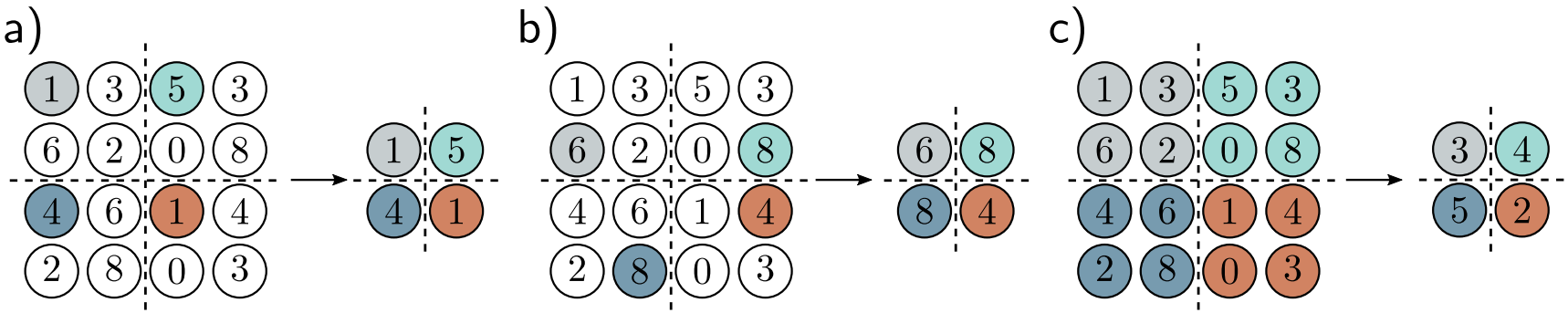
CNN for MNIST1D



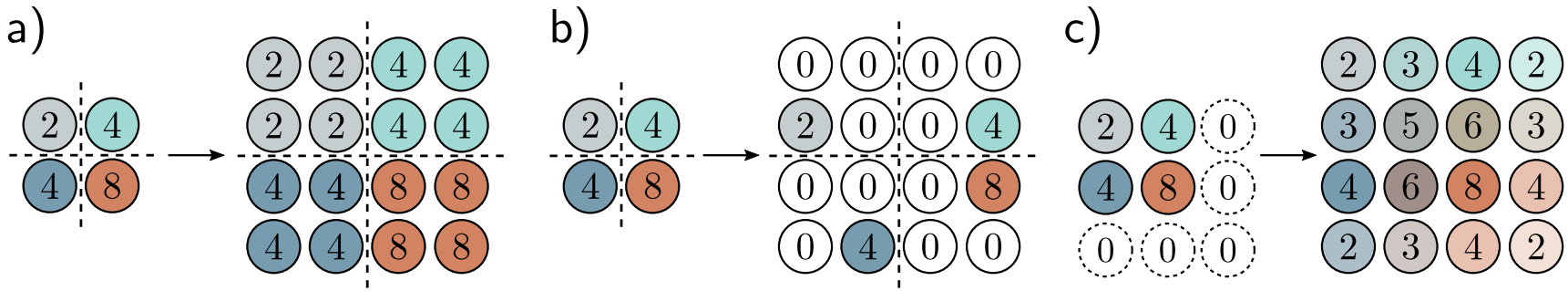
Results for MNIST1D



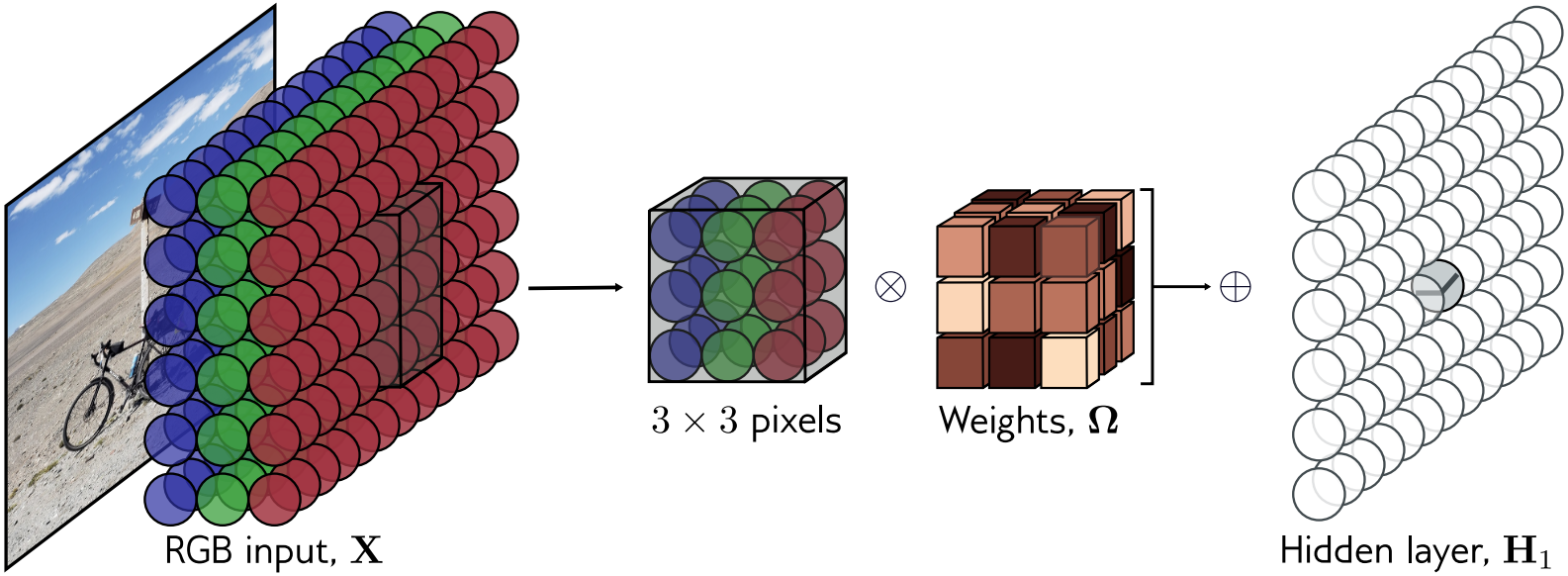
Downsampling



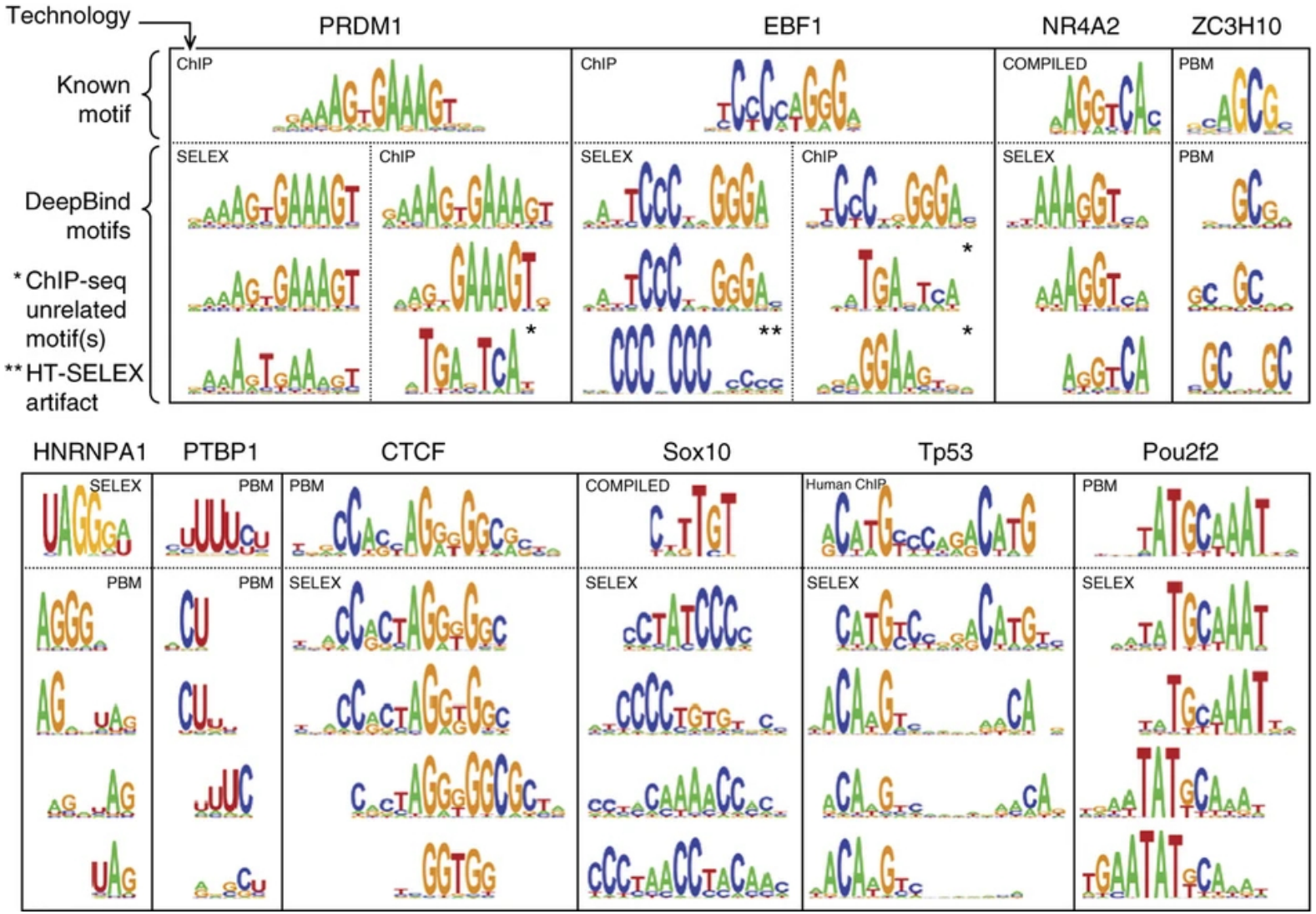
Upsampling


a)
2 4
4 8
2 2 4 4
2 2 4 4
4 4 8 8
4 4 8 8
b)
2 4
4 8
0 0 0 0
2 0 0 4
0 0 0 8
0 4 0 0
c)
2 4 0
4 8 0
0 0 0
2 3 4 2
3 5 6 3
4 6 8 4
2 3 4 2

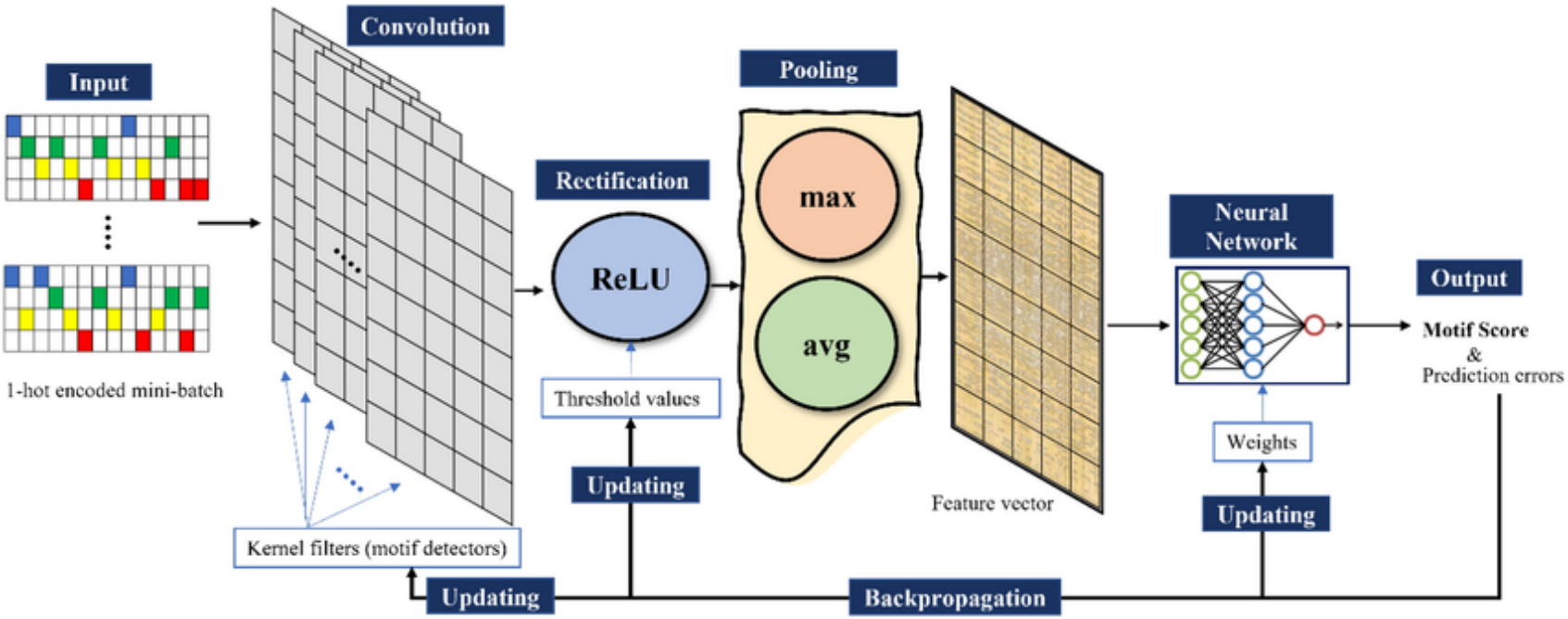
2D CNN


RGB input, $\mathbf{X}$
3×3 pixels
Weights, $\mathbf{\Omega}$
Hidden layer, $\mathbf{H}_1$

DeepBind


Technology
PRDM1
EBF1
NR4A2
ZC3H10
Known motif
DeepBind motifs
*ChIP-seq unrelated motif(s)
**HT-SELEX artifact
ChIP
SELEX
COMPILED
PBM
HNRNPA1
PTBP1
CTCF
Sox10
Tp53
Pou2f2
Human ChIP

DeepBind CNN archetecture



PSSM Matrix (Log-Odds Scores)

Position	A	C	G	T
1	1.65	-1.81	-1.81	-1.81
2	-1.81	-1.81	-1.81	1.65
3	-1.81	-1.81	1.65	-1.81
4	-1.81	1.00	-0.22	-0.22
5	1.36	-0.22	-1.81	-1.81

DeepBind CNN Model

```
# Build DeepBind-style model
from tensorflow.keras import layers, models

def make_deepbind_model(L=101, n_filters=32, filter_len=9):
    inp = layers.Input(shape=(L,4))
    x = layers.Conv1D(filters=n_filters, kernel_size=filter_len, activation='relu', name='conv1')(inp)
    x = layers.GlobalMaxPooling1D(name='gmp')(x)
    x = layers.Dense(32, activation='relu', name='dense1')(x)
    out = layers.Dense(1, activation='sigmoid', name='out')(x)
    model = models.Model(inputs=inp, outputs=out)
    model.compile(optimizer=optimizers.Adam(1e-3), loss='binary_crossentropy', metrics=['accuracy'])
    return model

model = make_deepbind_model(L=101, n_filters=32, filter_len=9)
model.summary()
```

Deep Bind

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 101, 4)	0
conv1 (Conv1D)	(None, 93, 32)	1,184
gmp (GlobalMaxPooling1D)	(None, 32)	0
dense1 (Dense)	(None, 32)	1,056
out (Dense)	(None, 1)	33

Total params: 2,273 (8.88 KB)

Trainable params: 2,273 (8.88 KB)

Non-trainable params: 0 (0.00 B)