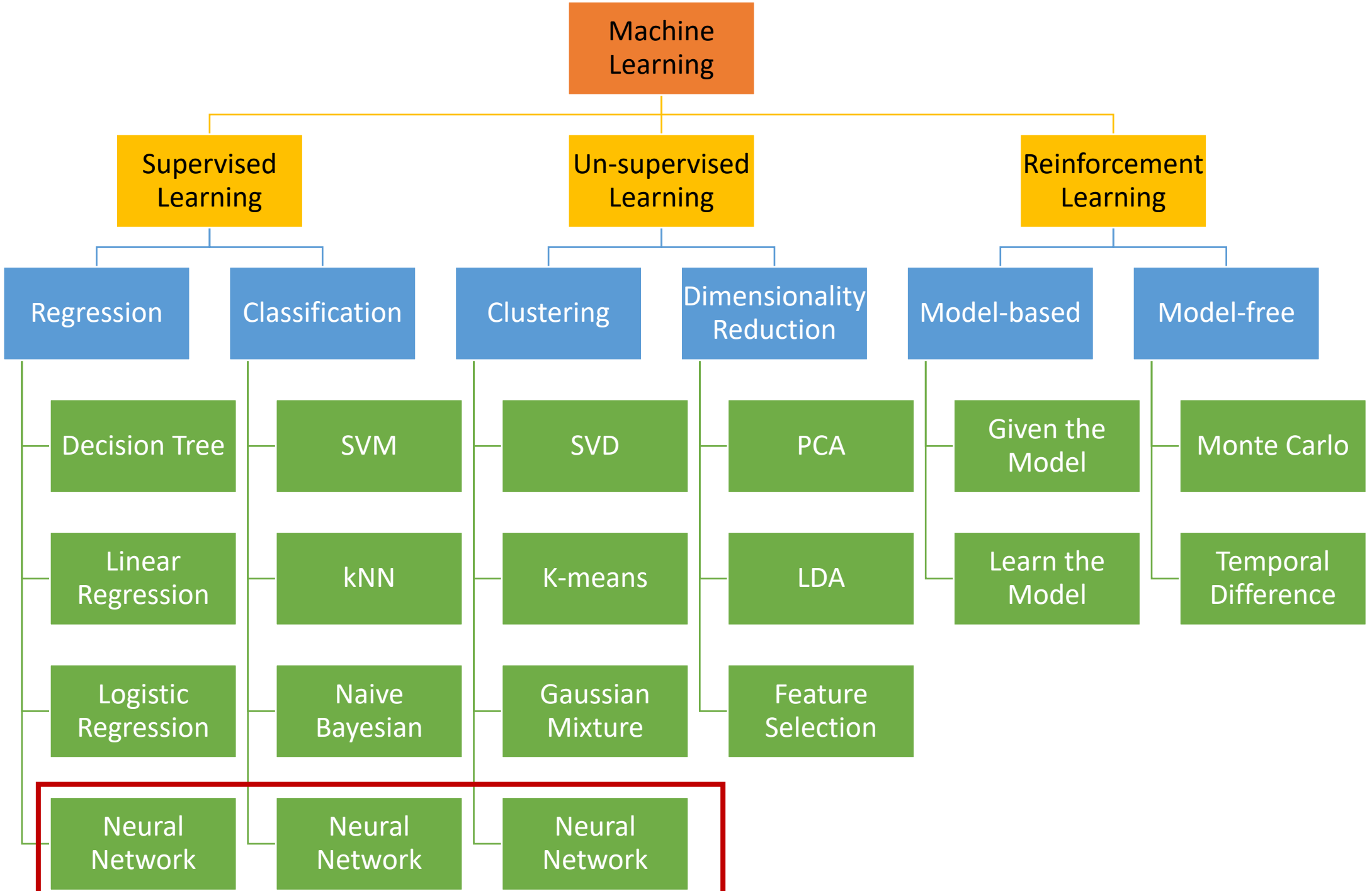
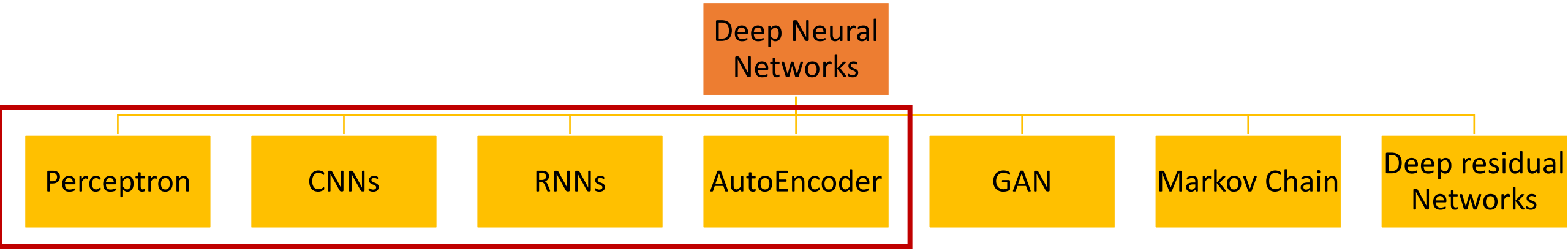
Summer Bootcamp 2021

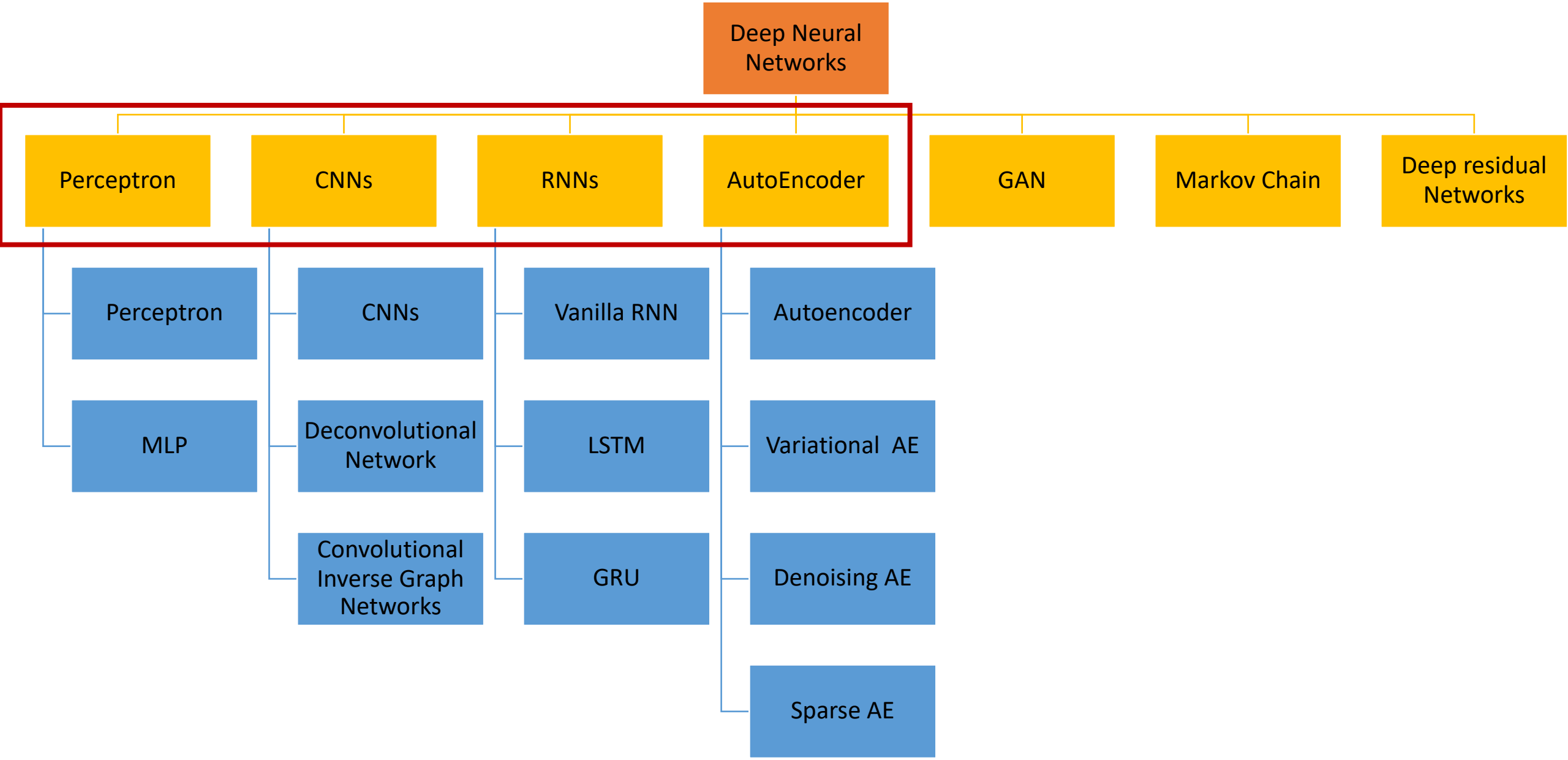
Applied Machine Learning
Deep Neural Networks: RECAP

Ngan Le

thile@uark.edu

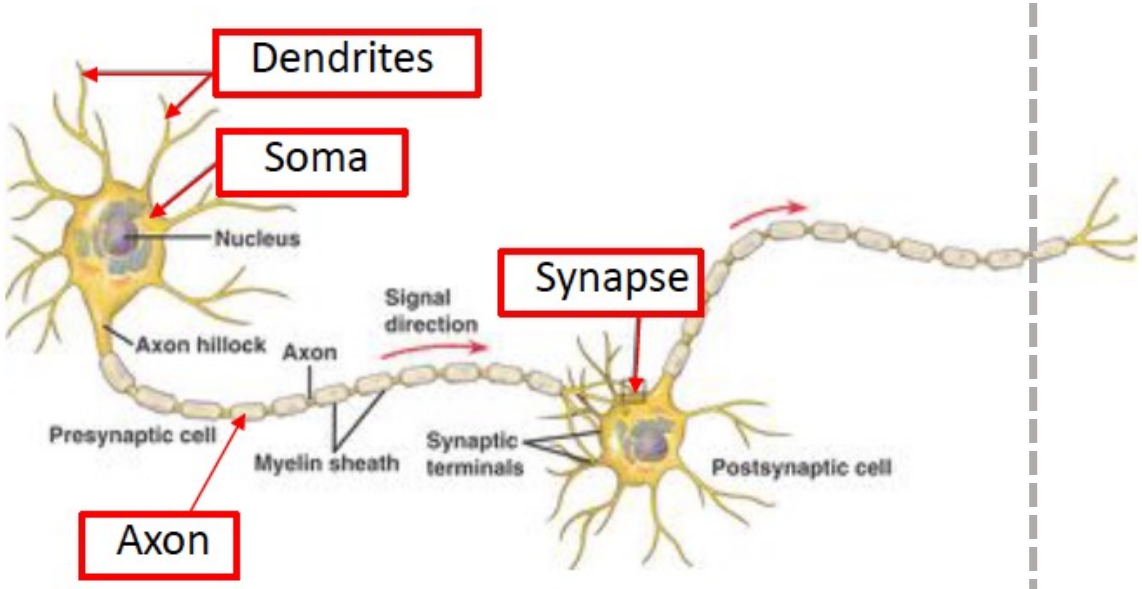

Machine Learning
Supervised Learning
Un-supervised Learning
Reinforcement Learning
Regression
Classification
Clustering
Dimensionality Reduction
Model-based
Model-free
Decision Tree
Linear Regression
Logistic Regression
SVM
kNN
Naive Bayesian
SVD
K-means
Gaussian Mixture
PCA
LDA
Feature Selection
Given the Model
Learn the Model
Monte Carlo
Temporal Difference
Neural Network
Neural Network
Neural Network


Deep Neural Networks
Perceptron
CNNs
RNNs
AutoEncoder
GAN
Markov Chain
Deep residual Networks

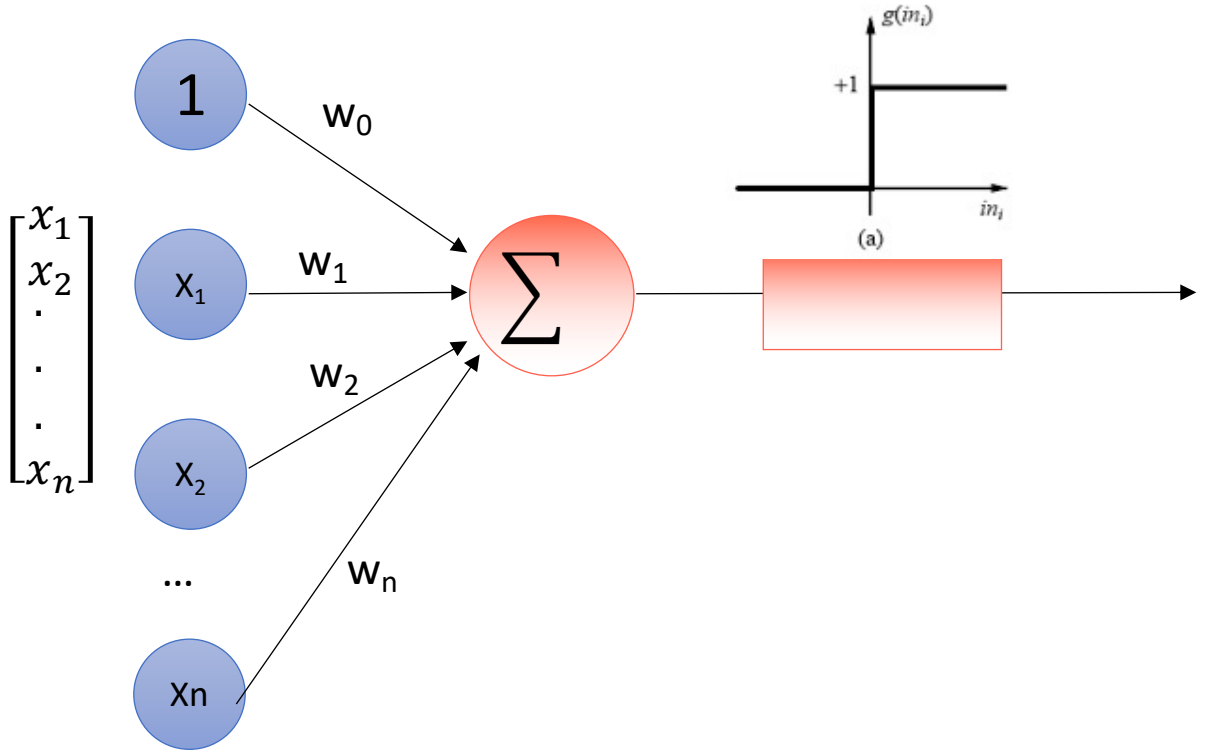

Deep Neural Networks
Perceptron
CNNs
RNNs
AutoEncoder
GAN
Markov Chain
Deep residual Networks
Perceptron
MLP
CNNs
Deconvolutional Network
Convolutional Inverse Graph Networks
Vanilla RNN
LSTM
GRU
Autoencoder
Variational AE
Denoising AE
Sparse AE

Neural Network

Perceptron

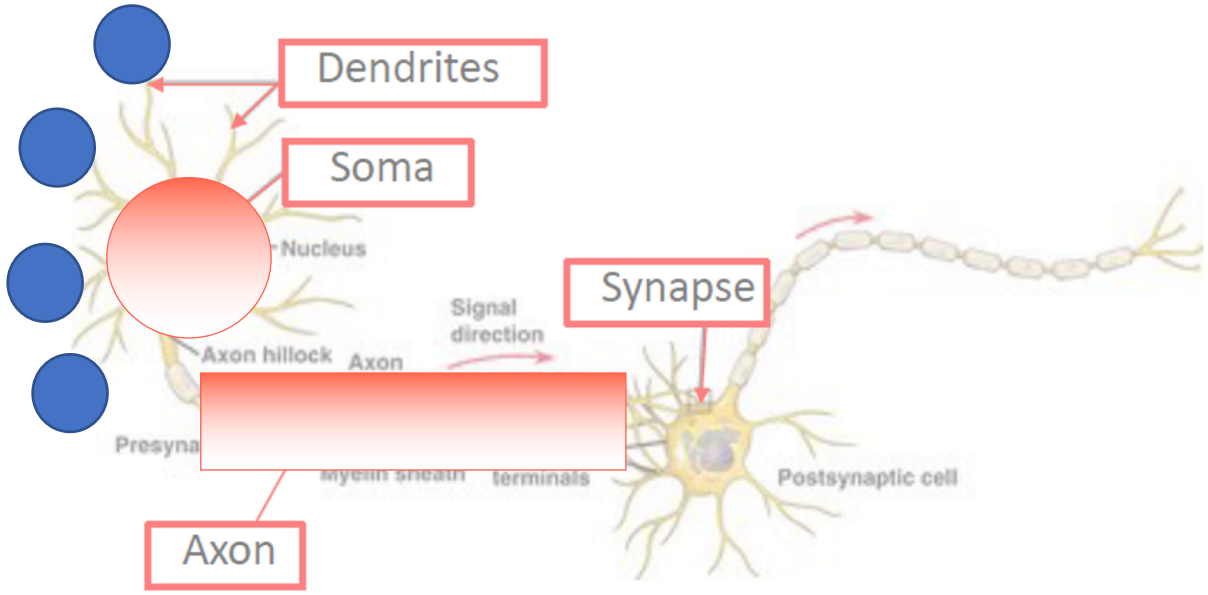


Cells that fire together, wire together

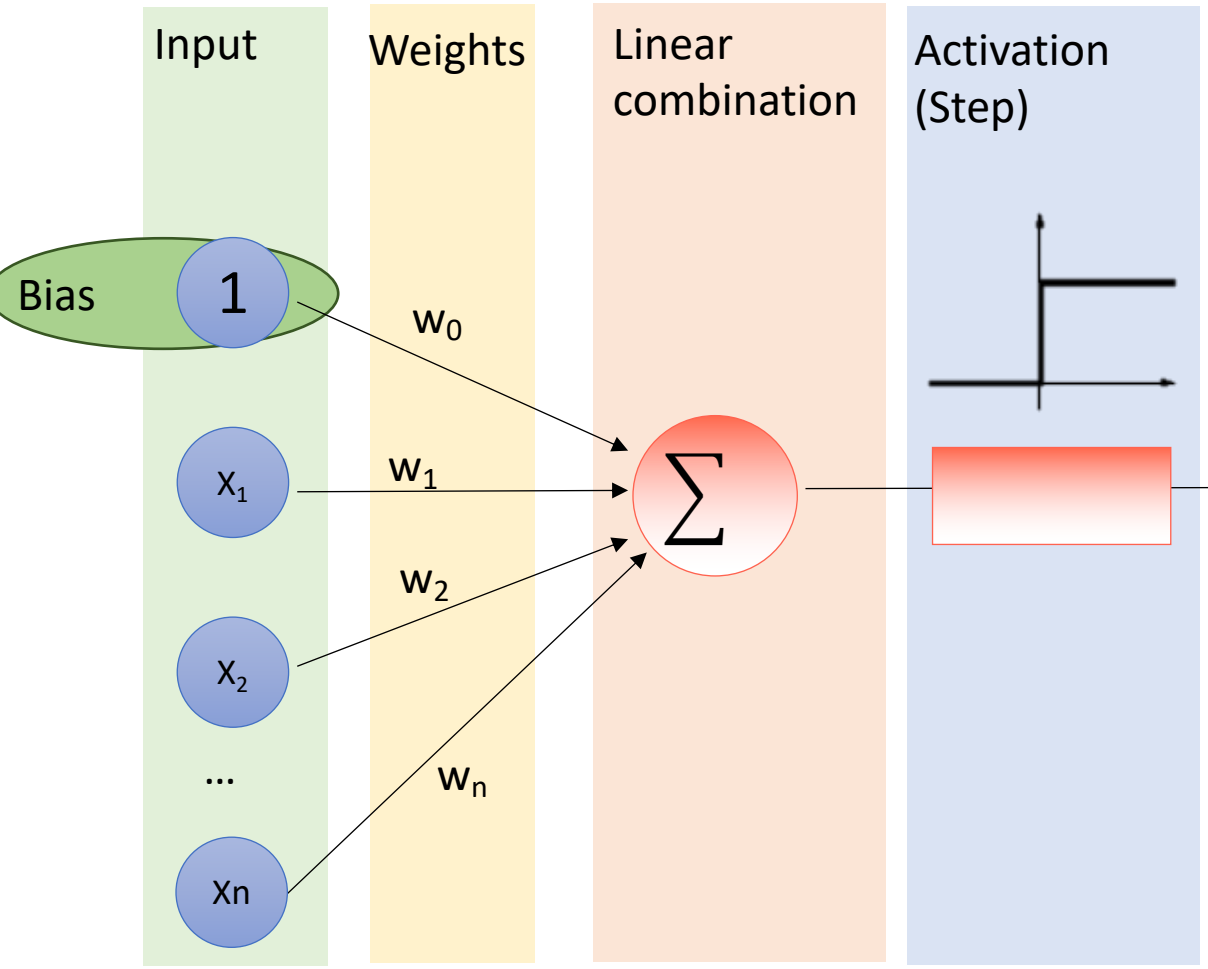


$$\begin{cases} y = 1 \quad if \quad \sum w_0 1 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > T \\ y = 0 \qquad\qquad otherwise \end{cases}$$

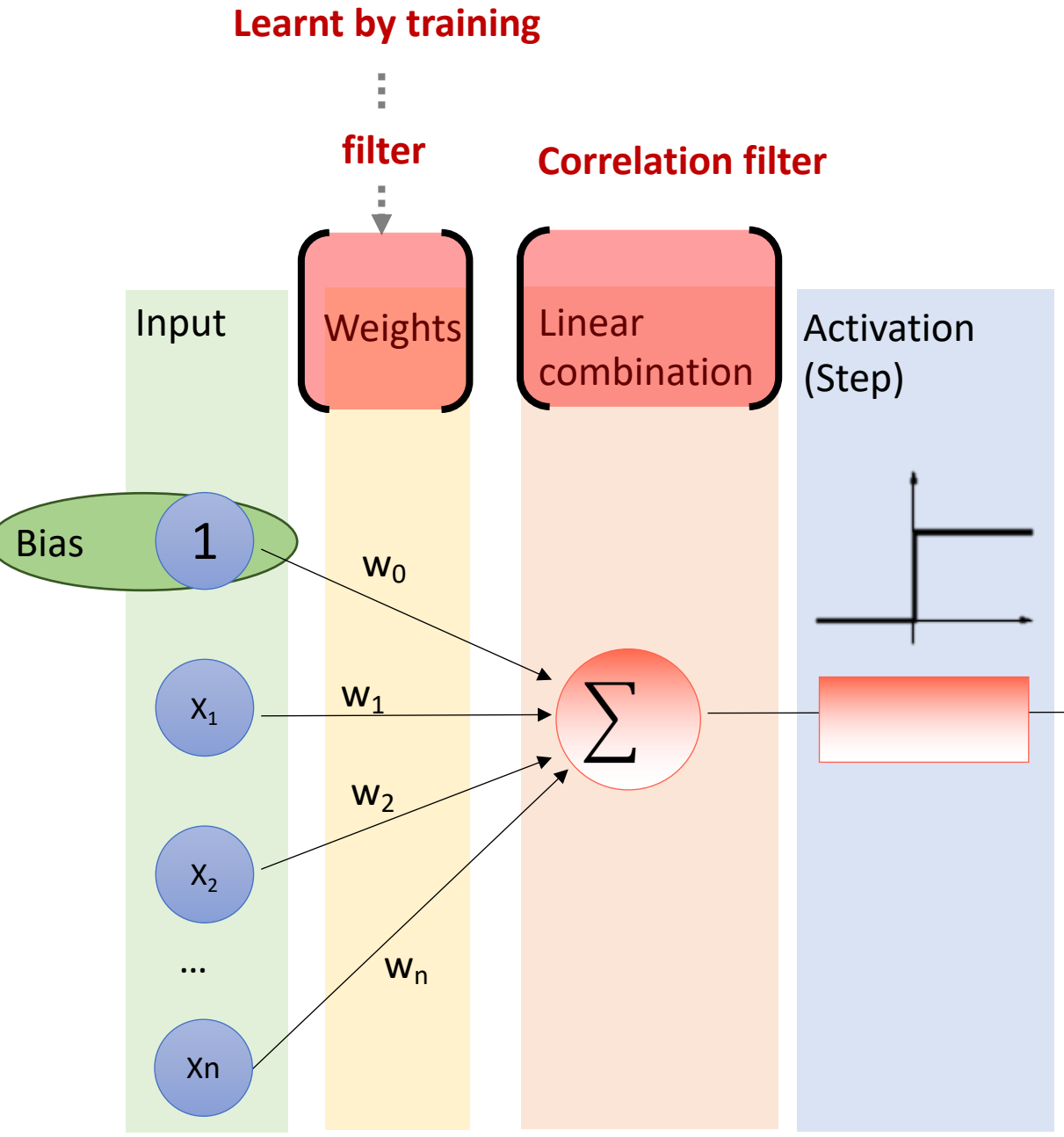
Perceptron



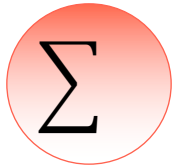
Cells that fire together, wire together



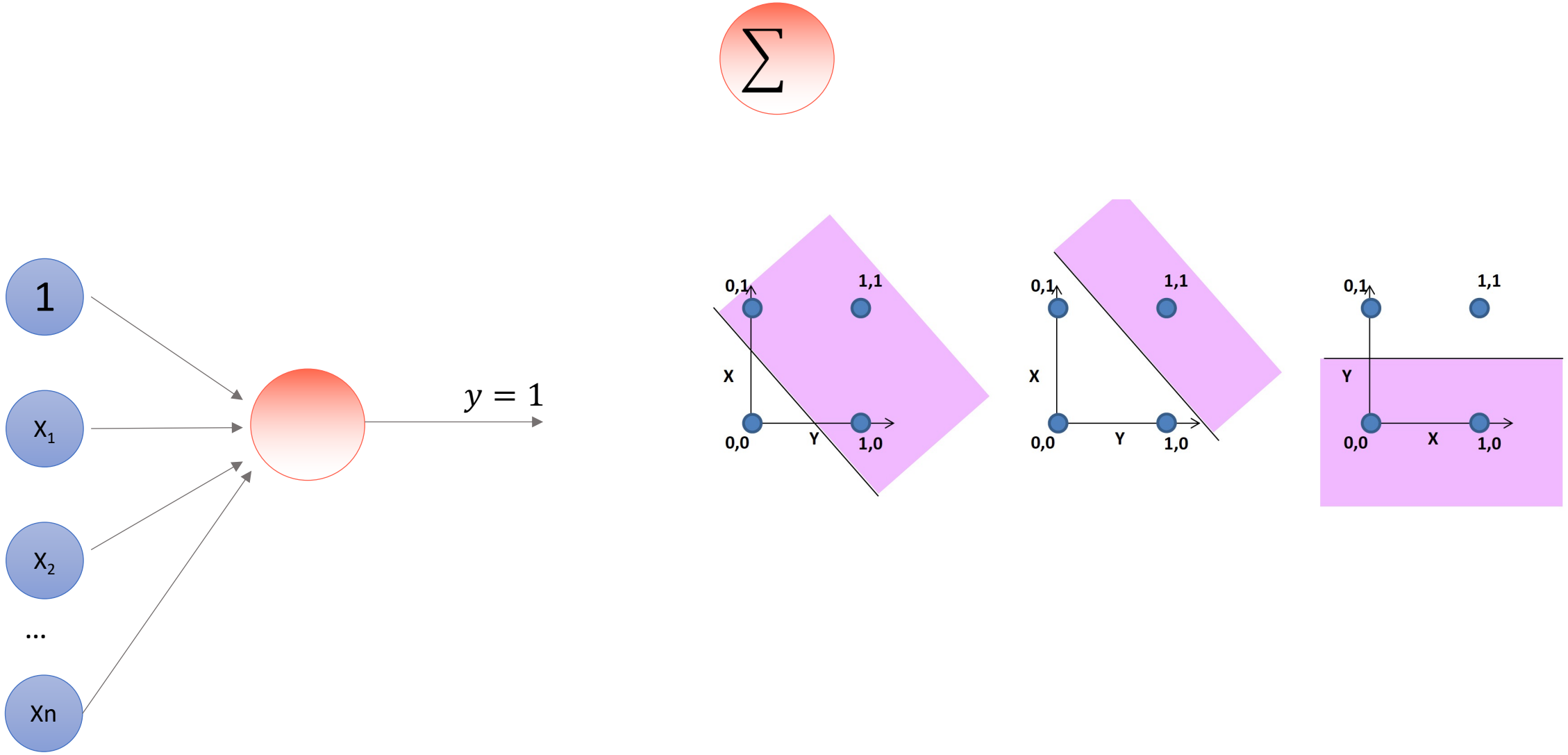
$$\begin{cases} y = 1 & if \quad \sum w_0 1 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > T \\ y = 0 & otherwise \end{cases}$$

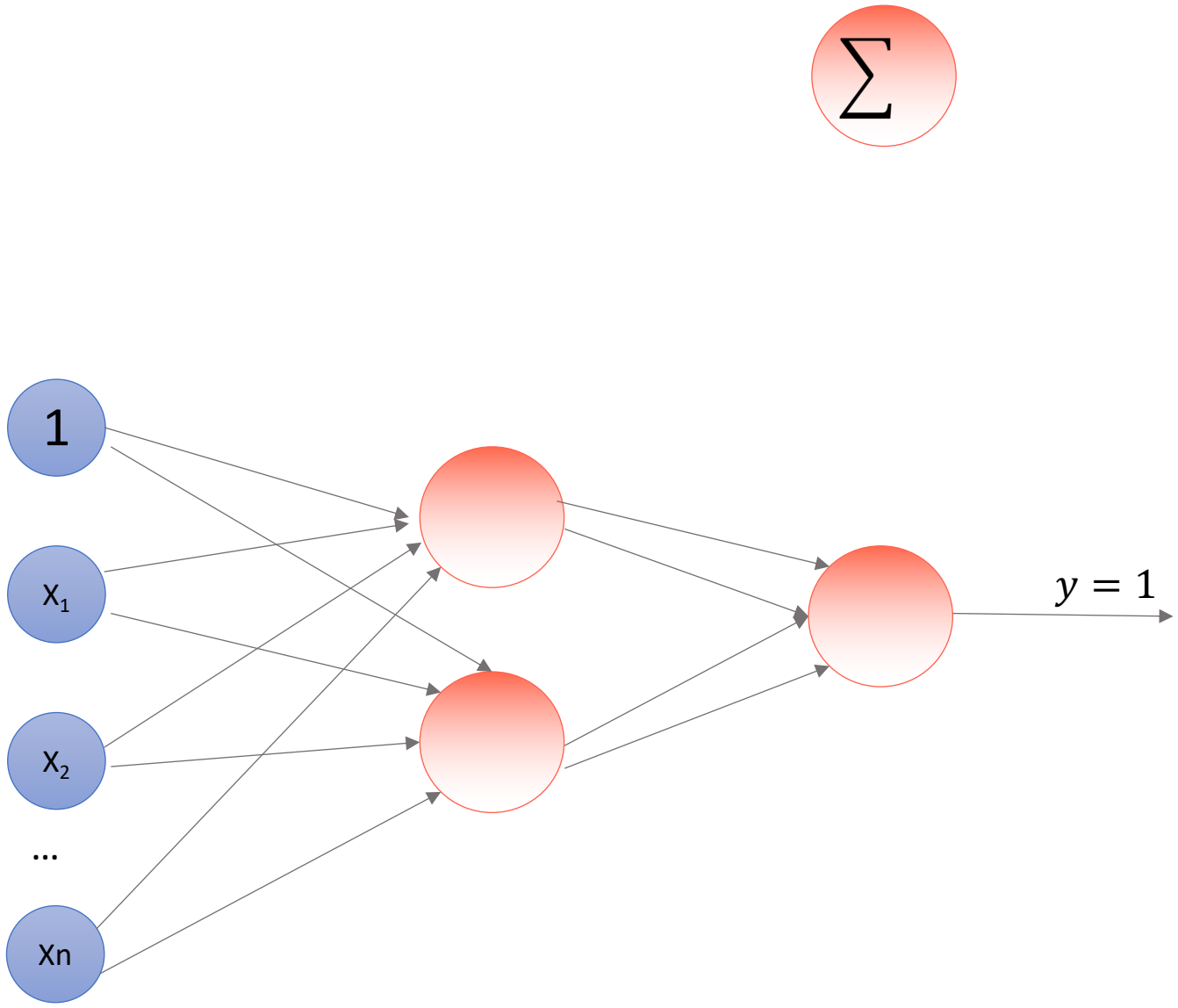


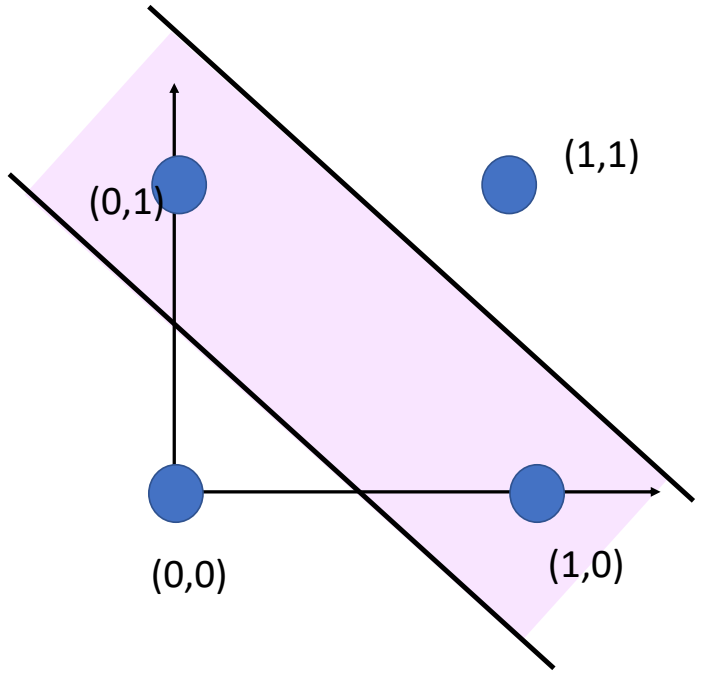
$$\begin{cases} y = 1 \quad if \quad \sum w_0 1 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > T \\ y = 0 \qquad\qquad otherwise \end{cases}$$

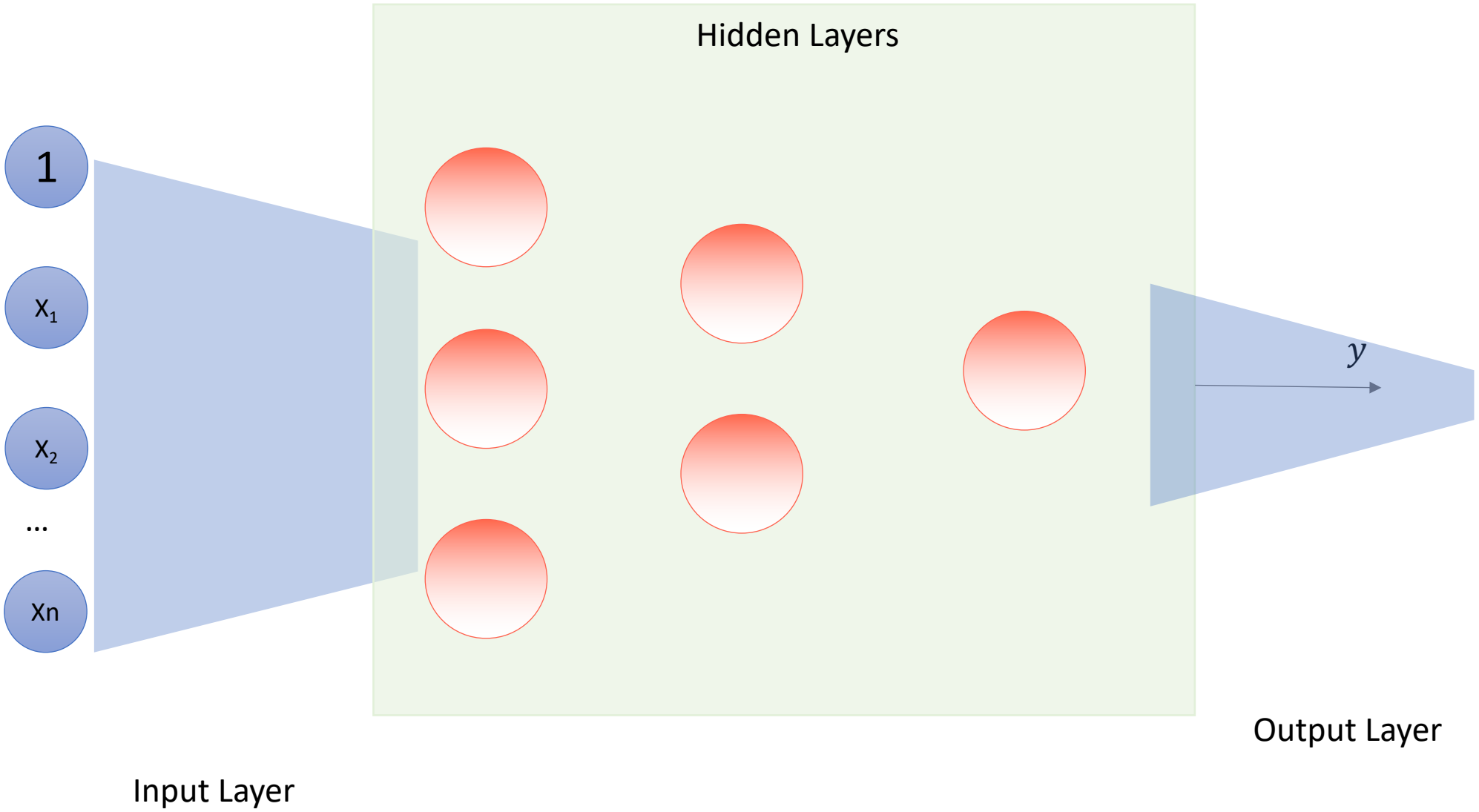


- How many perceptrons?
- How is perceptron’s arrange?

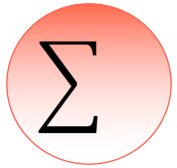

∑
1
X1
X2
...
Xn
y = 1
0,1
1,1
X
0,0
Y
1,0
0,1
1,1
X
0,0
Y
1,0
0,1
1,1
Y
0,0
X
1,0


∑
1
x_1
x_2
...
xn
$y = 1$

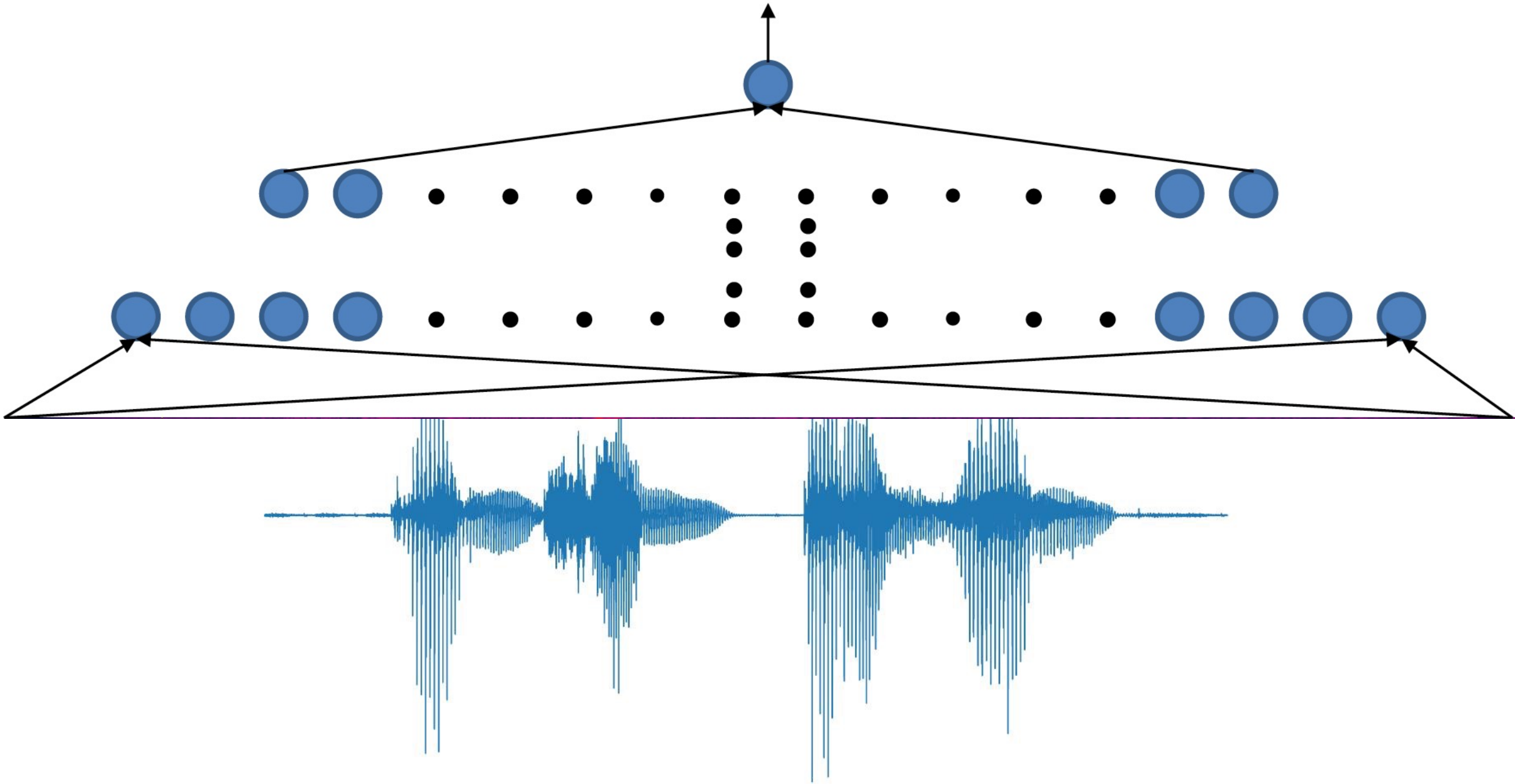

(0,1)
(1,1)
(0,0)
(1,0)



Multi Layer Perceptron (MLP)

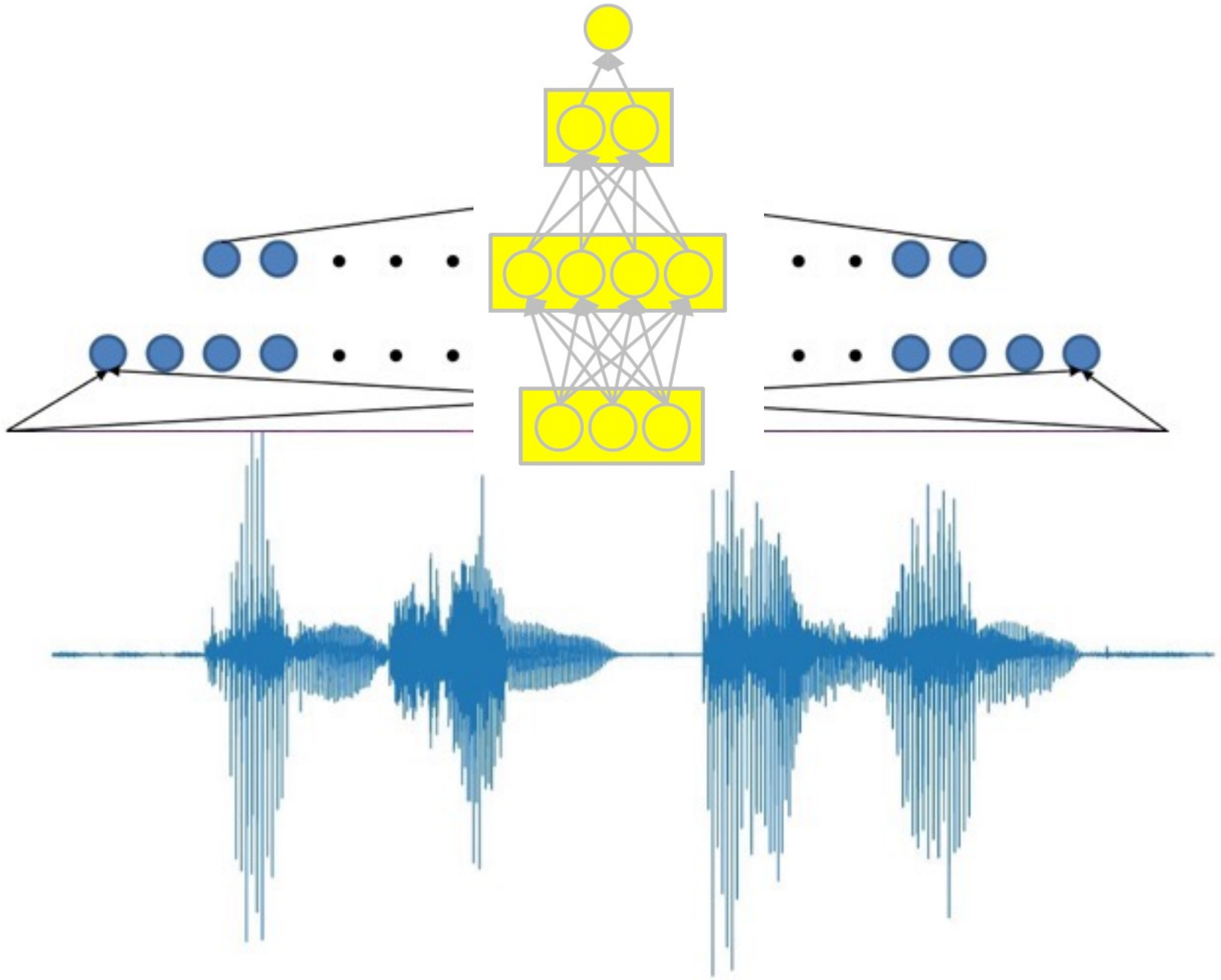


- Which information that one perceptron cover?

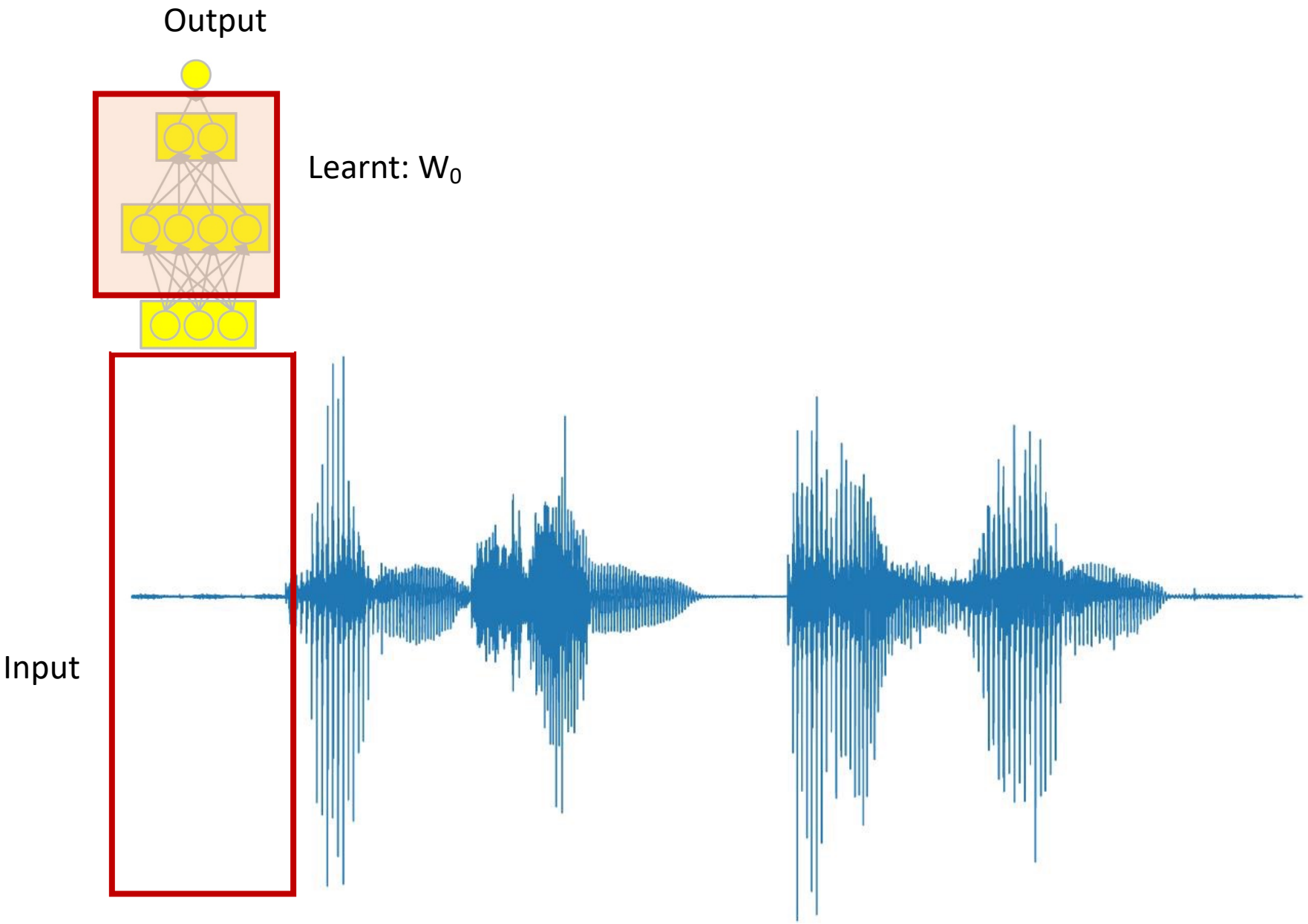


Solution for?

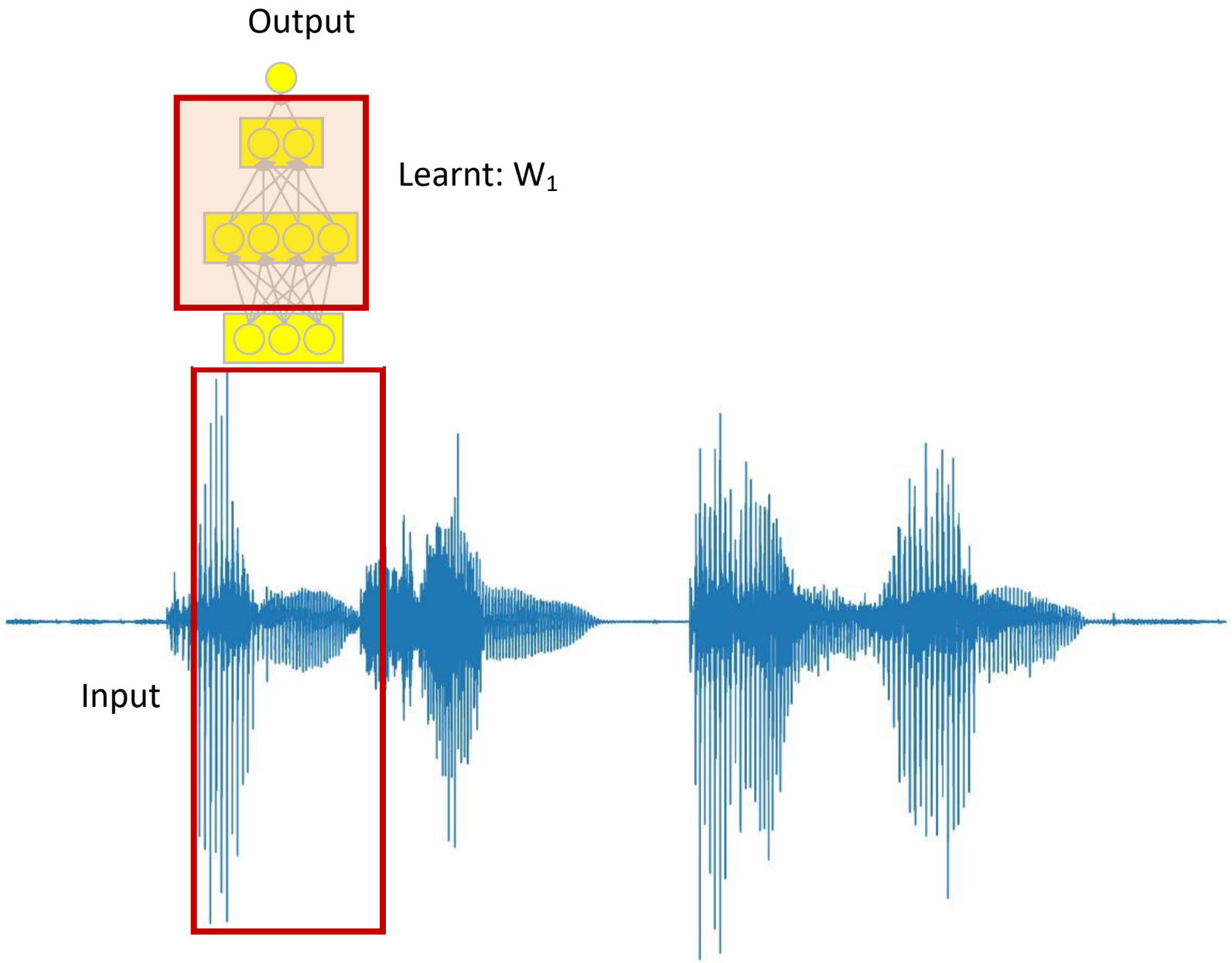
Limitations?



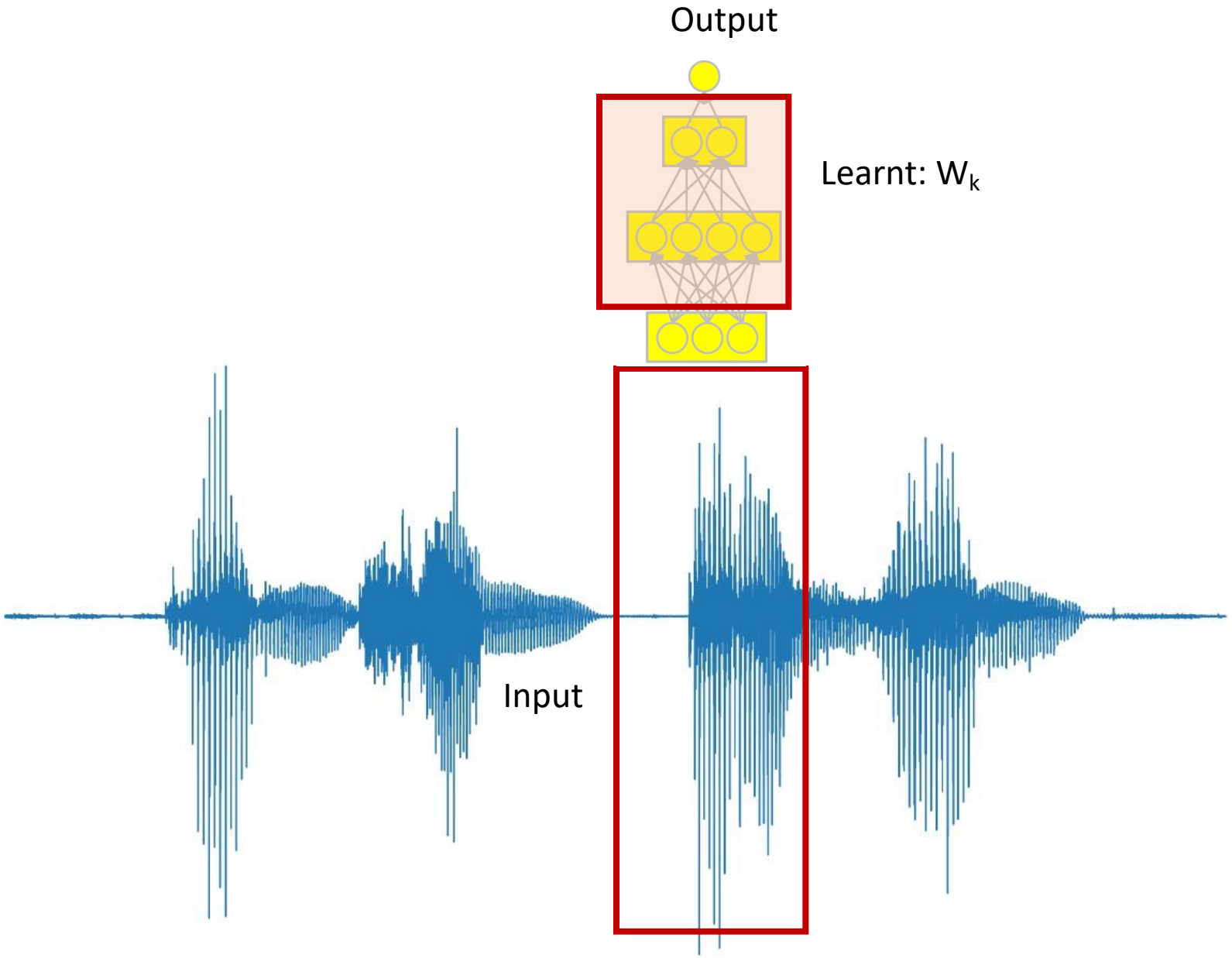
Search Pattern: Scanning



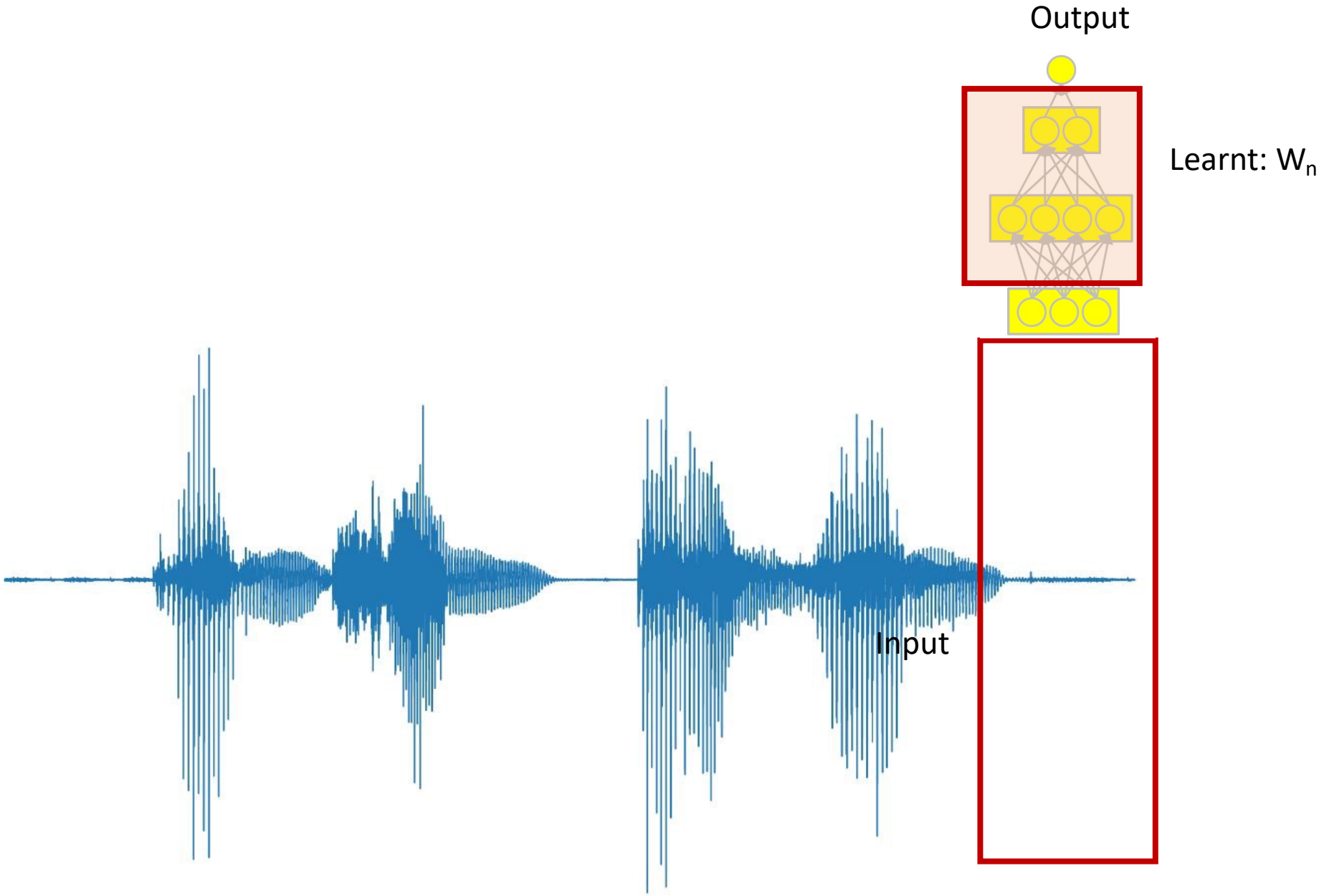
Search Pattern: Scanning



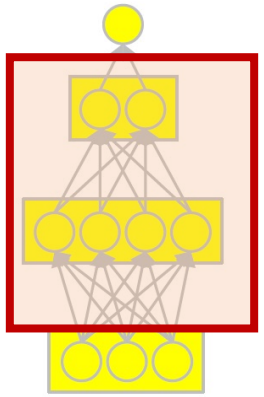
Search Pattern: Scanning



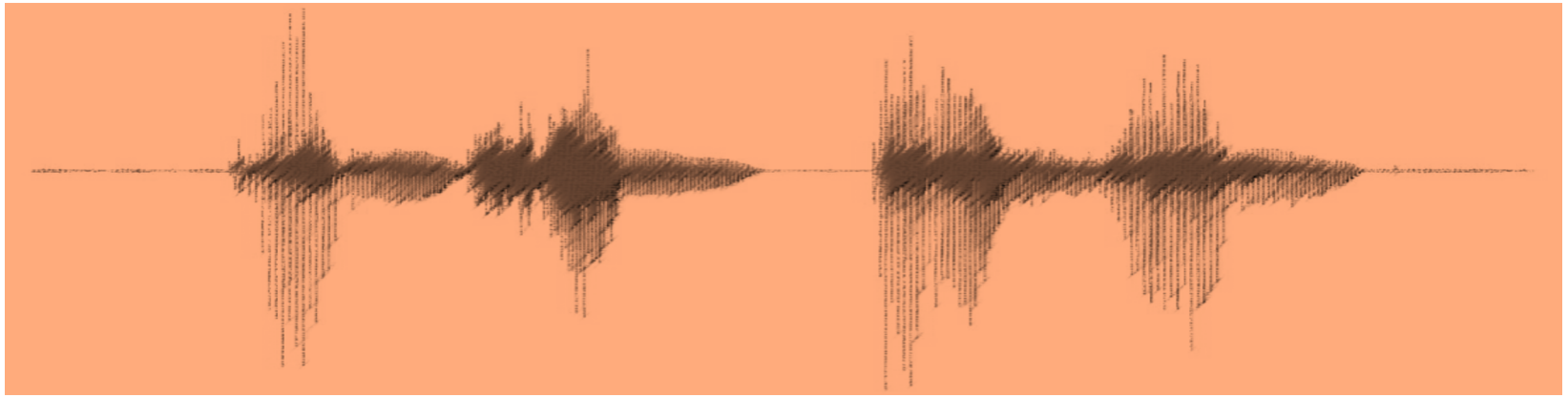
Search Pattern: Scanning

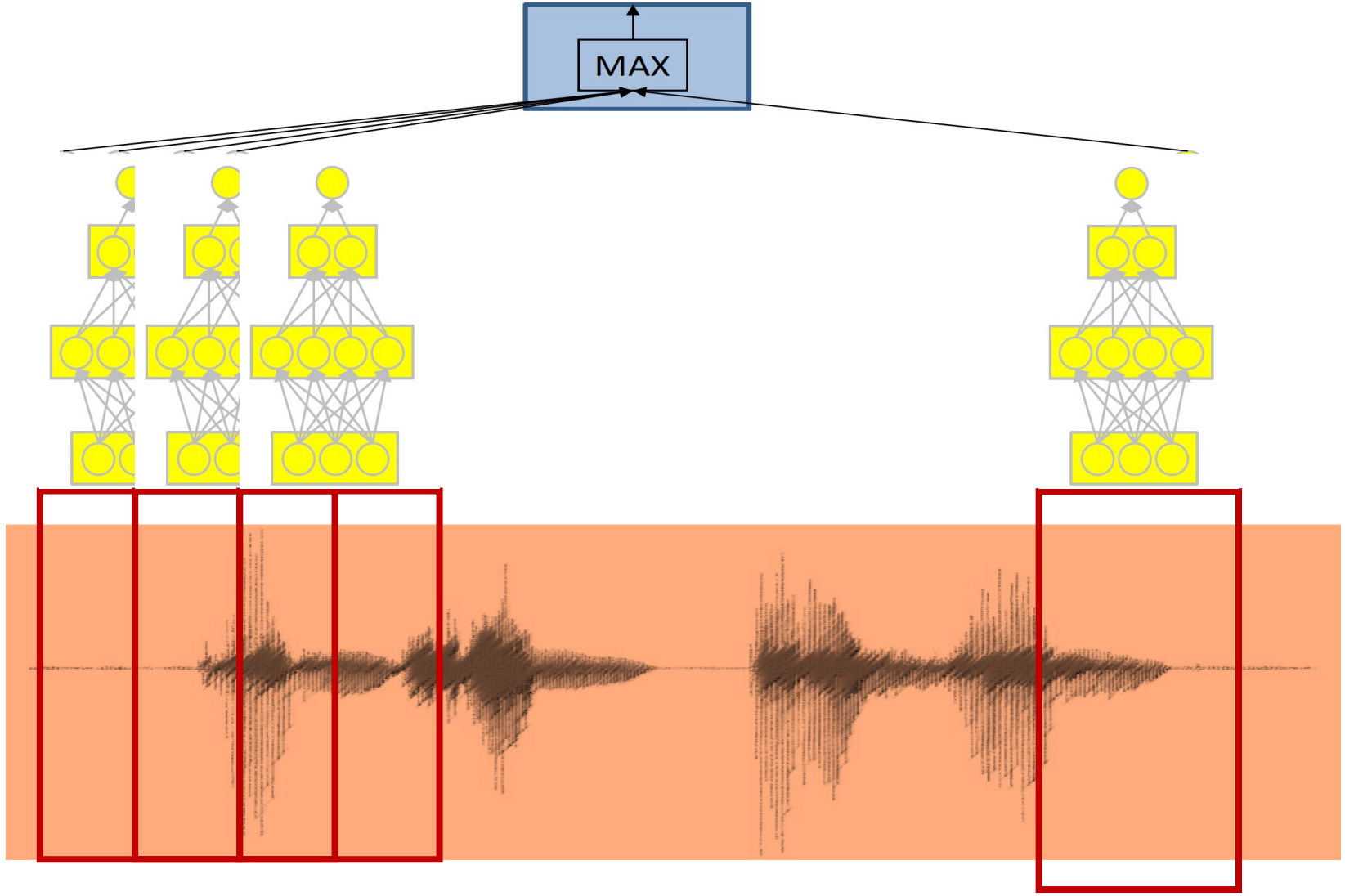


Search Pattern: Scanning

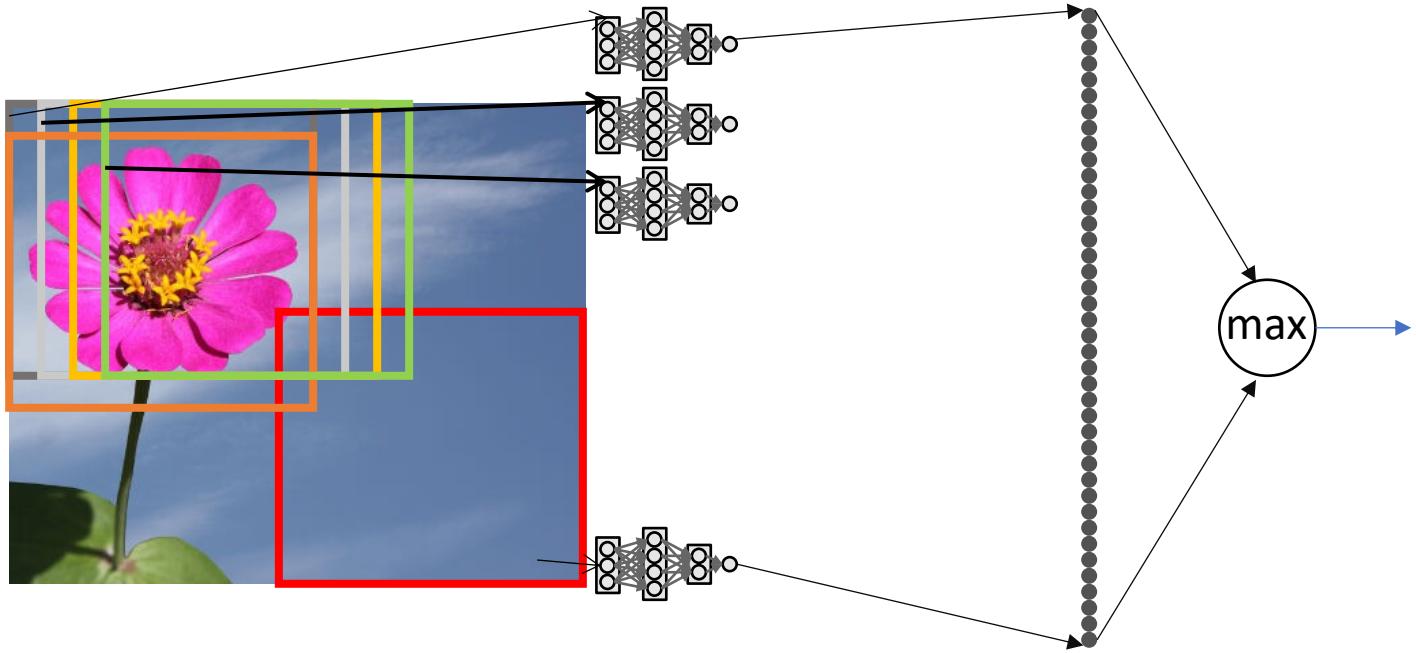


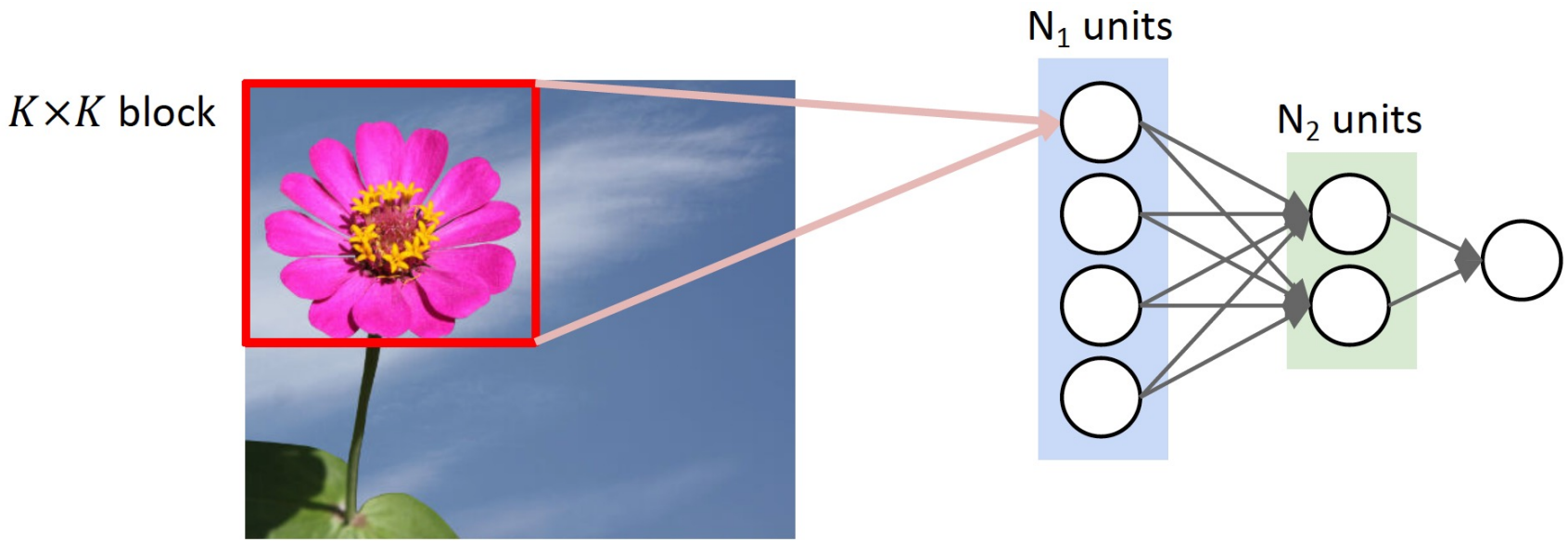
Learnt: W




MAX

“Look” for the target object at each position




$K \times K$ block
N_1 units
N_2 units

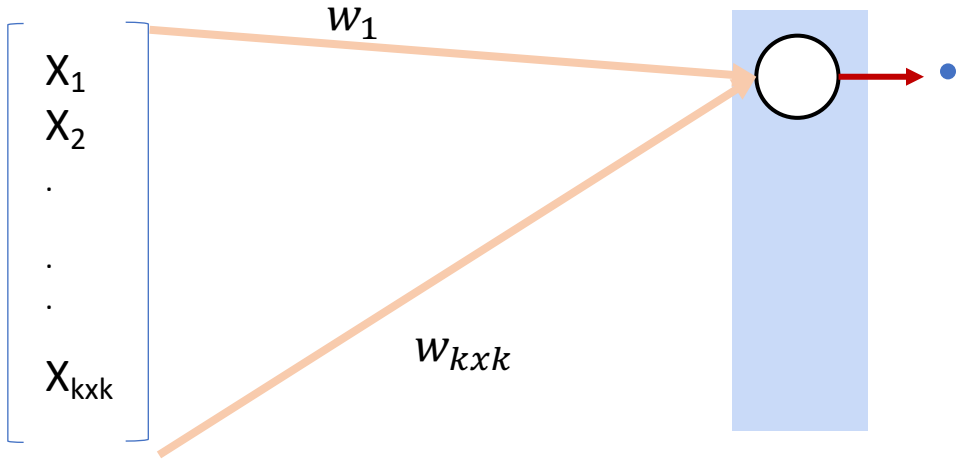
A set of pixels → Concatenate all pixels

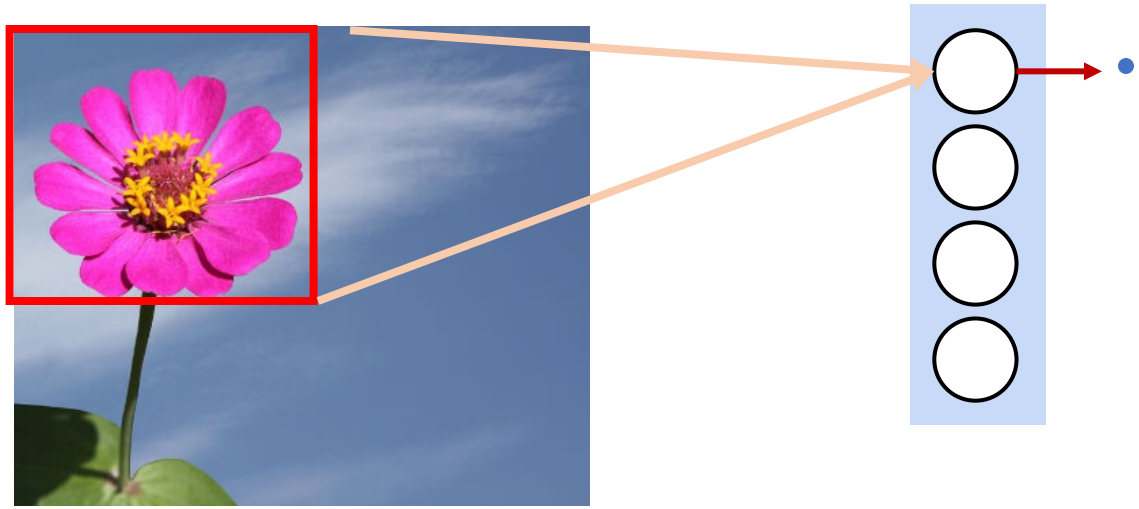
KxK block

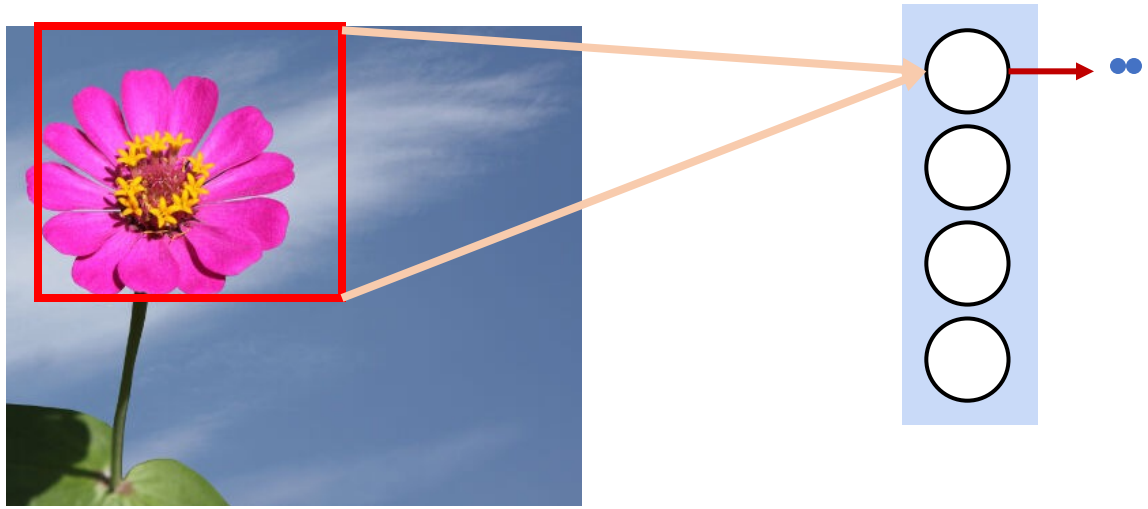
Vector

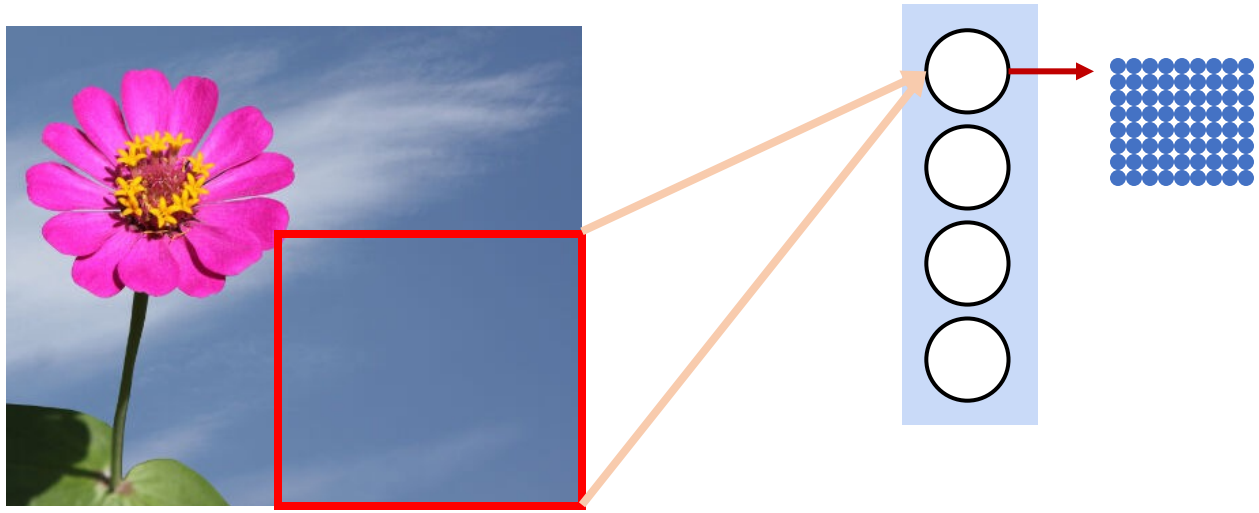
$$reLU\left(\sum w_1x_1 + w_2x_2 + \cdots + w_{kxk}x_{kxk}\right)$$

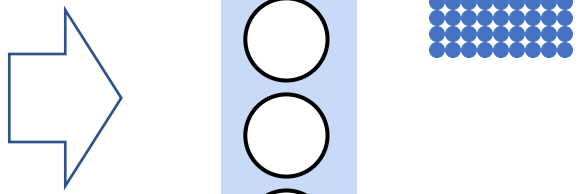
$$\sum w_1x_1 + w_2x_2 + \cdots + w_{kxk}x_{kxk} > T$$

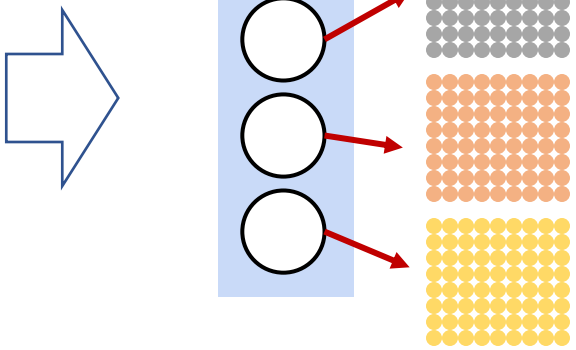


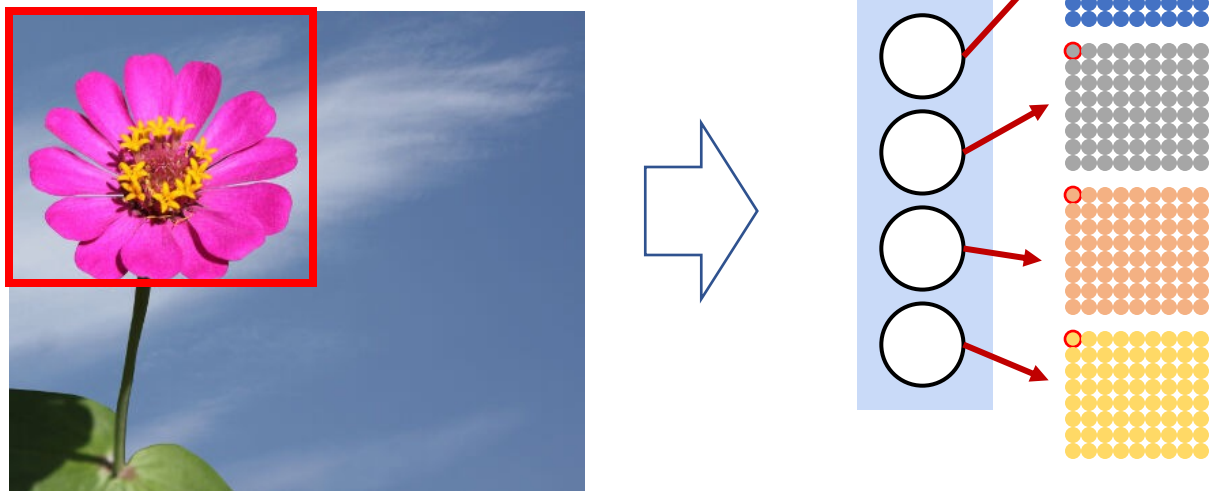


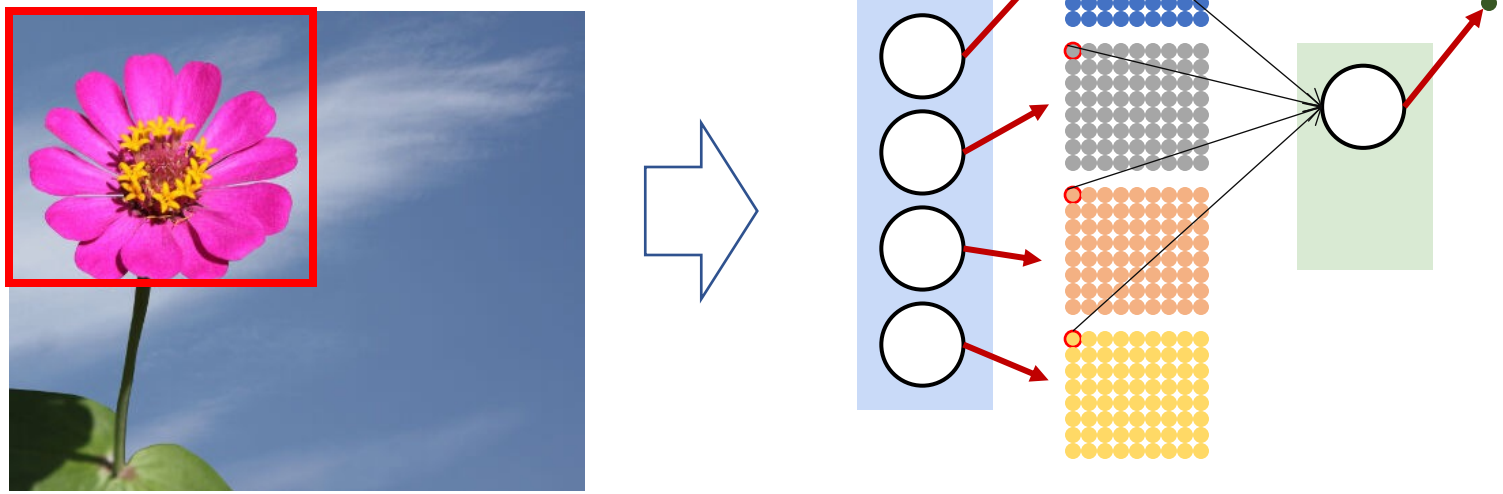


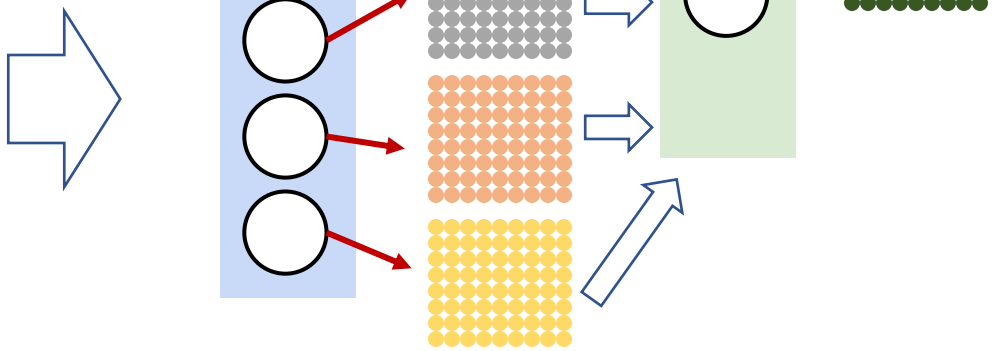


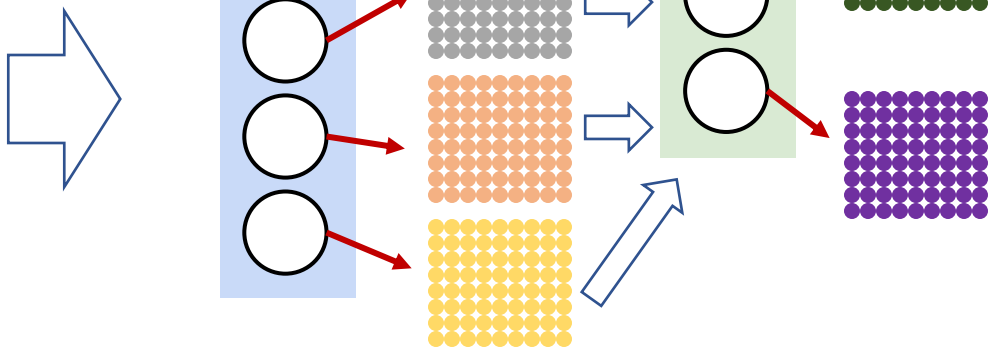


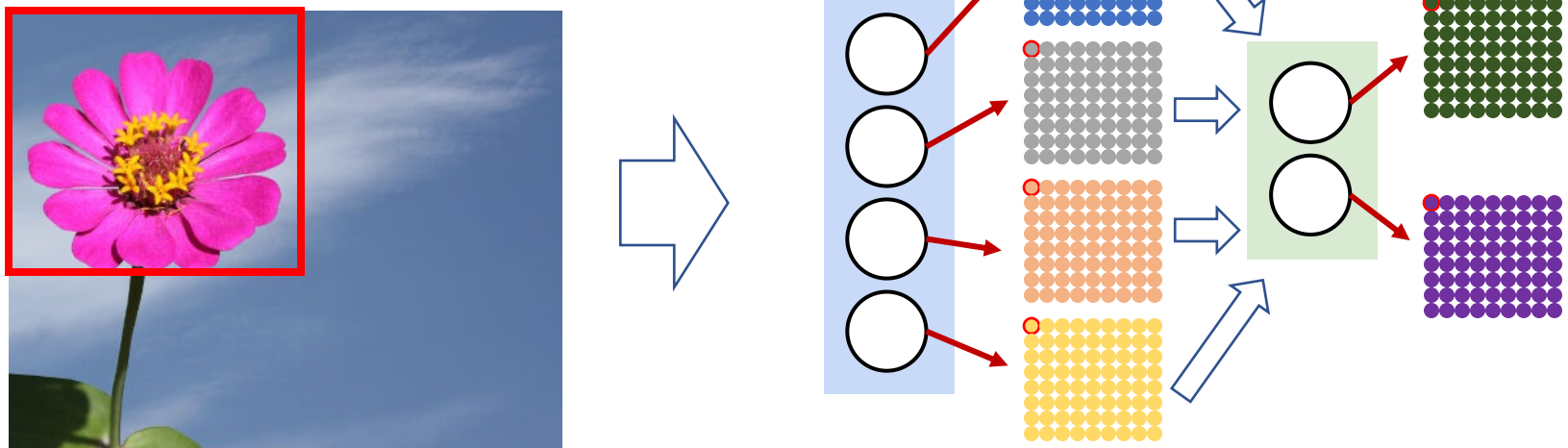


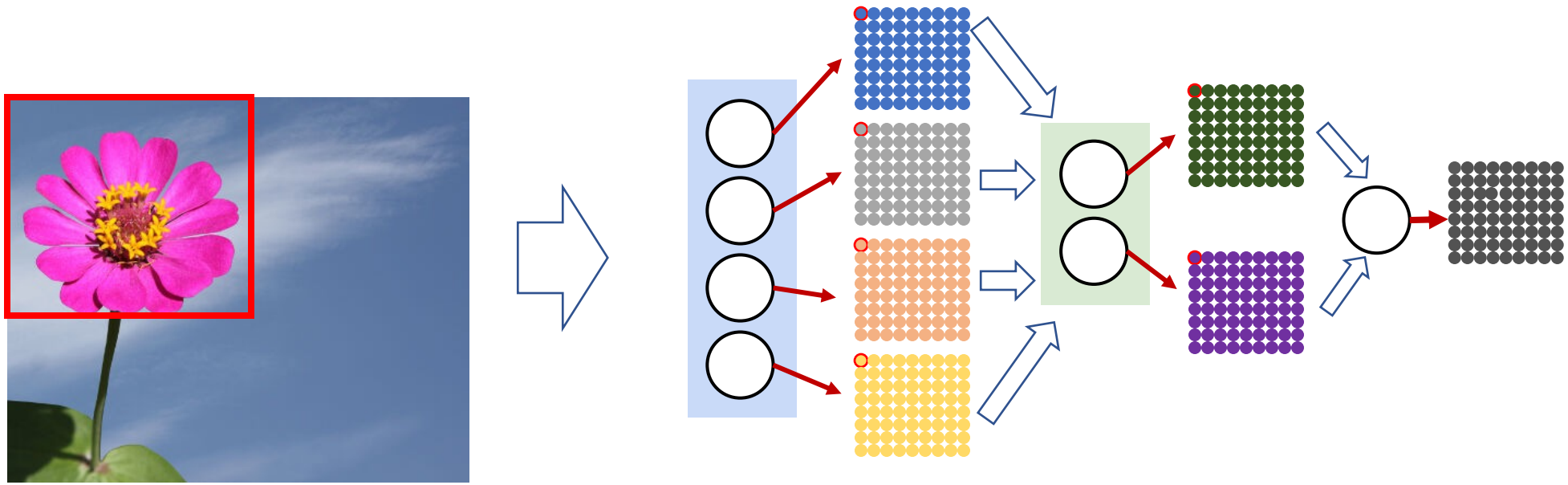


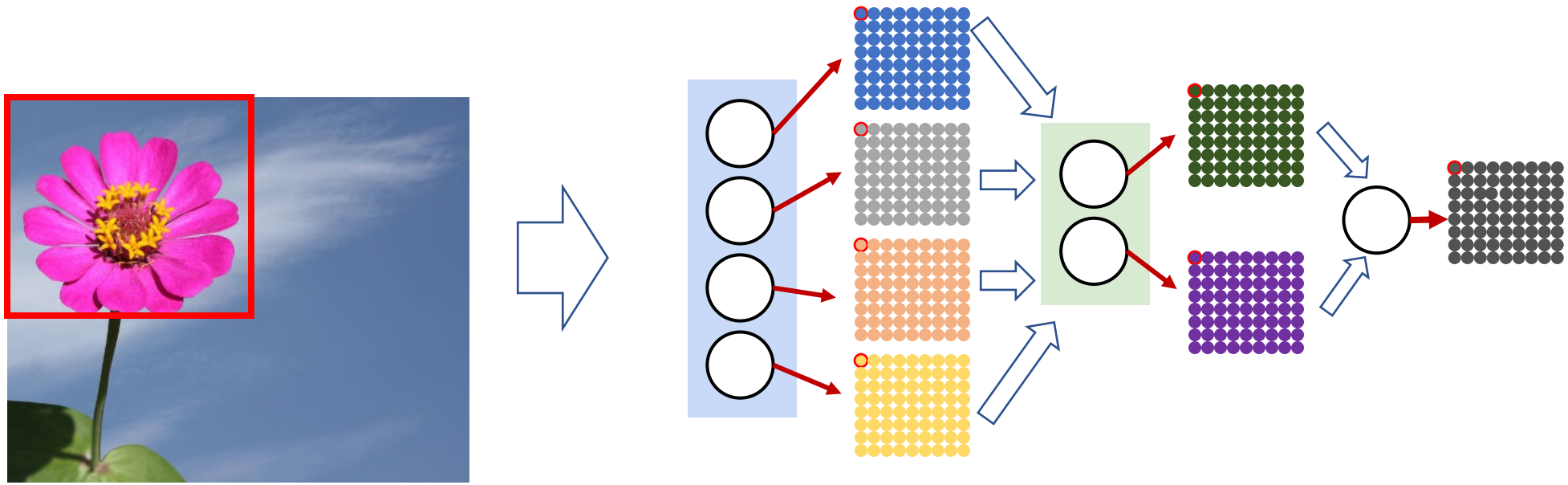


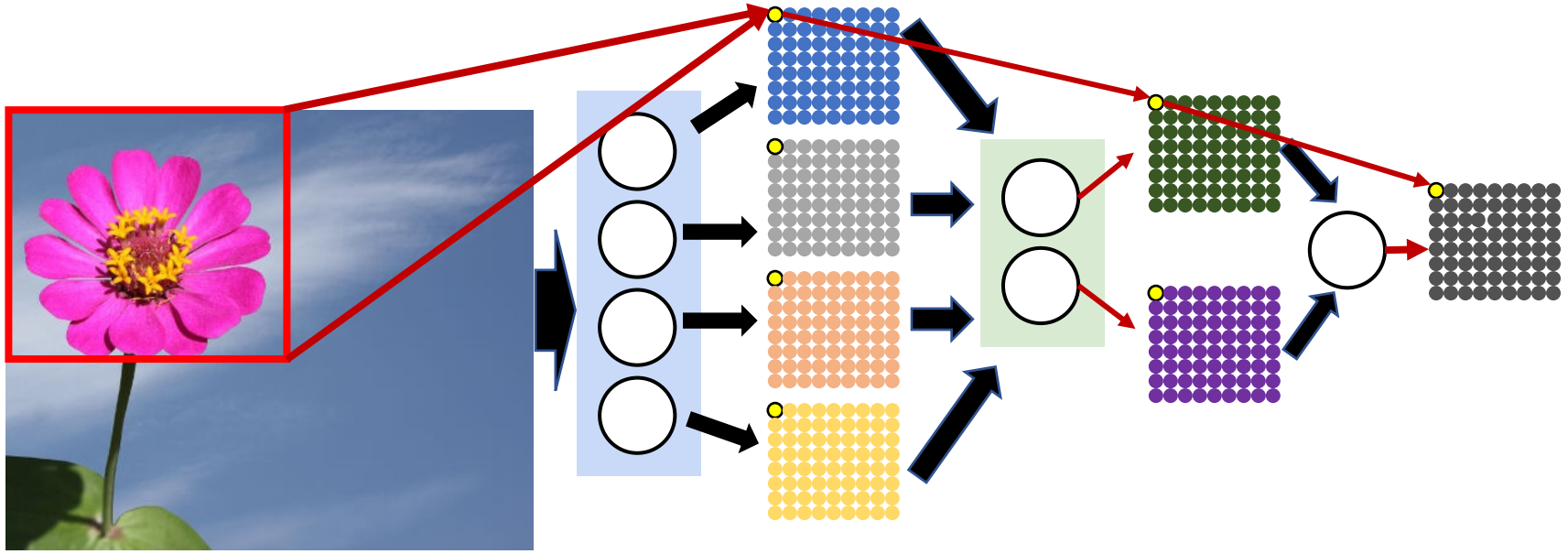




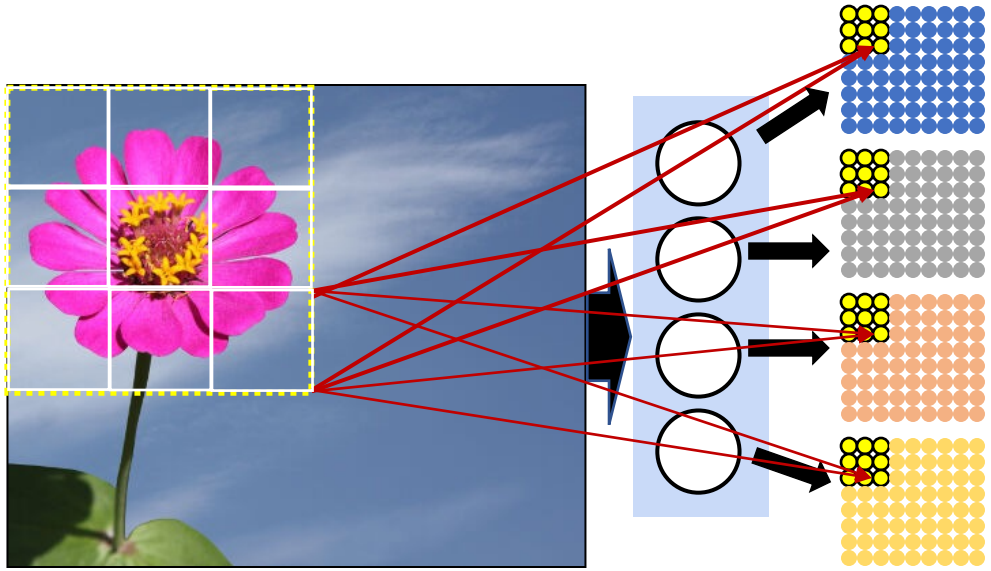






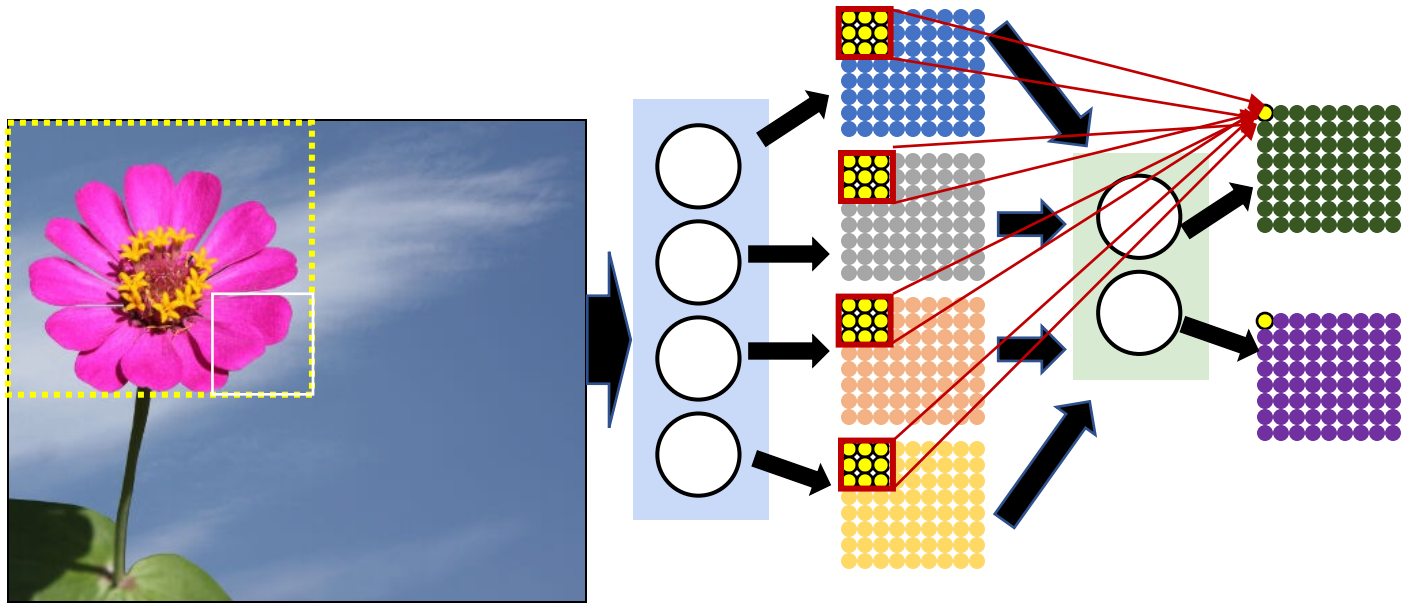


Flower Detection: Distributing the scan



The first layer evaluates smaller blocks of pixels

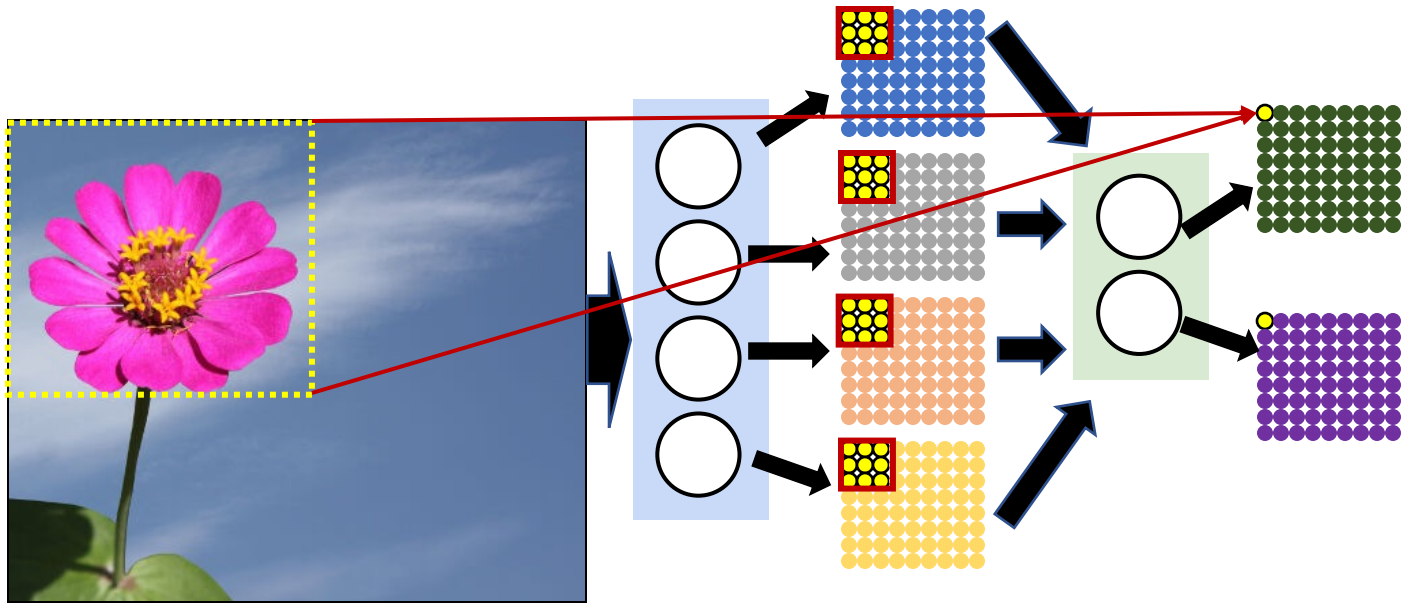
Flower Detection: Distributing the scan



The first layer evaluates smaller blocks of pixels

The next layer evaluates blocks of outputs from the first layer

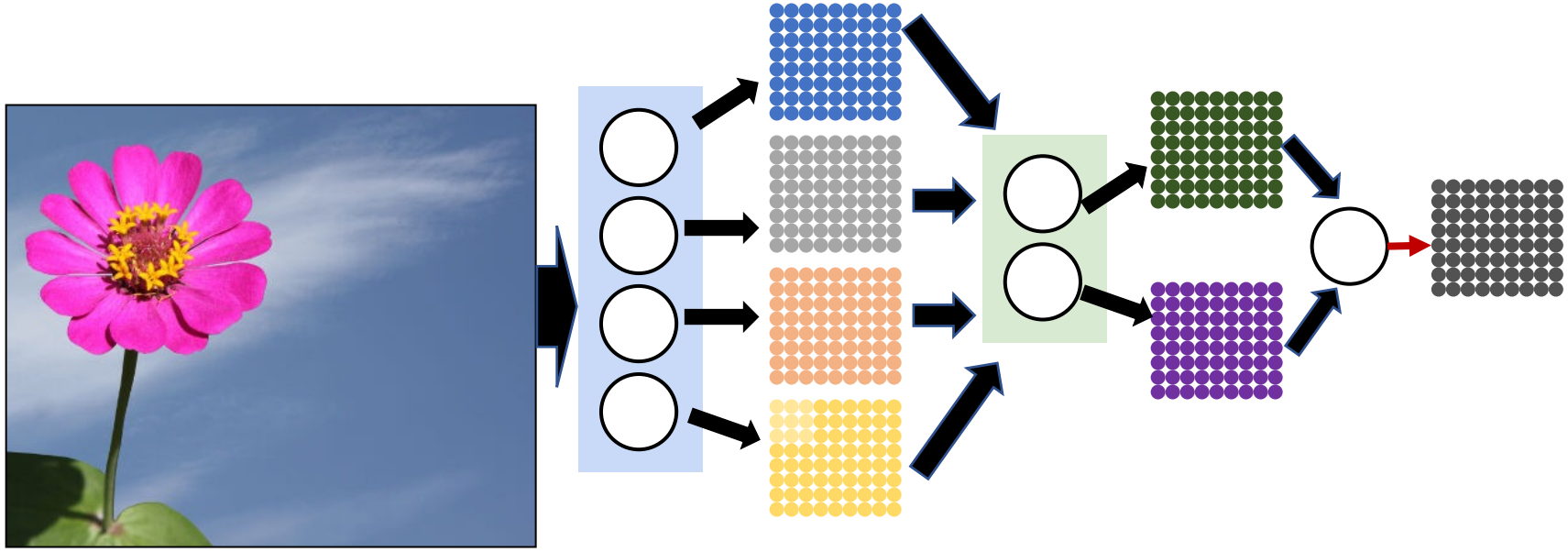
Flower Detection: Distributing the scan



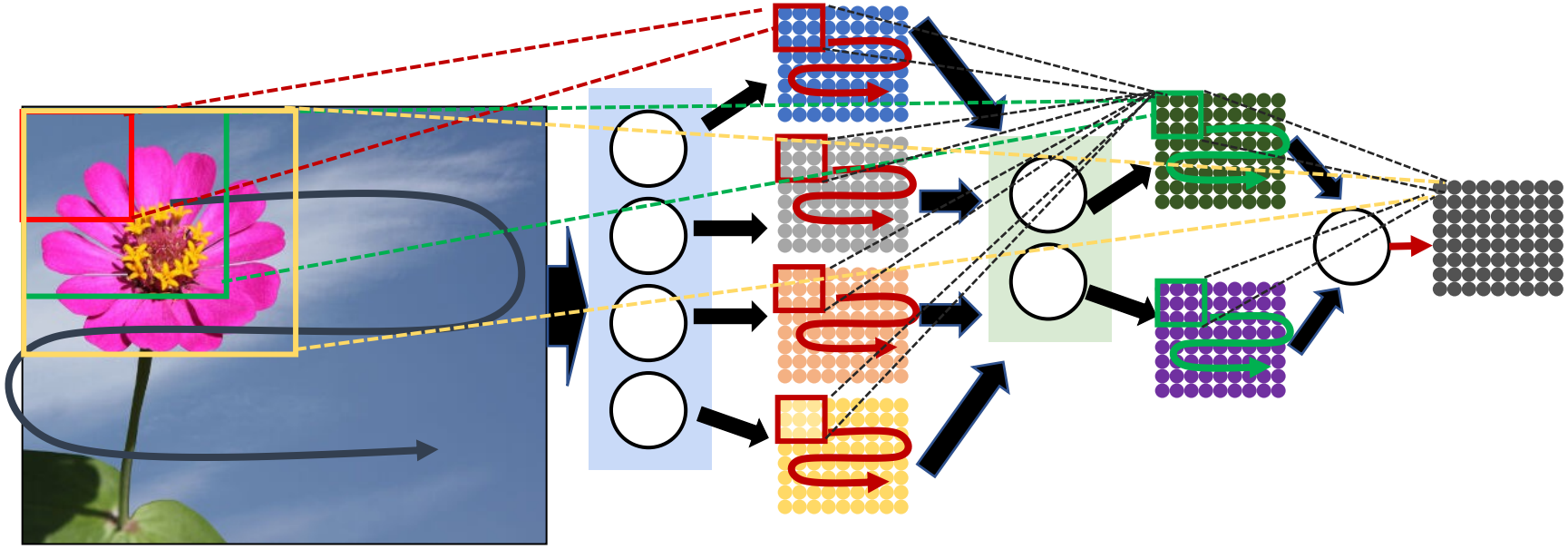
The first layer evaluates smaller blocks of pixels

The next layer evaluates blocks of outputs from the first layer

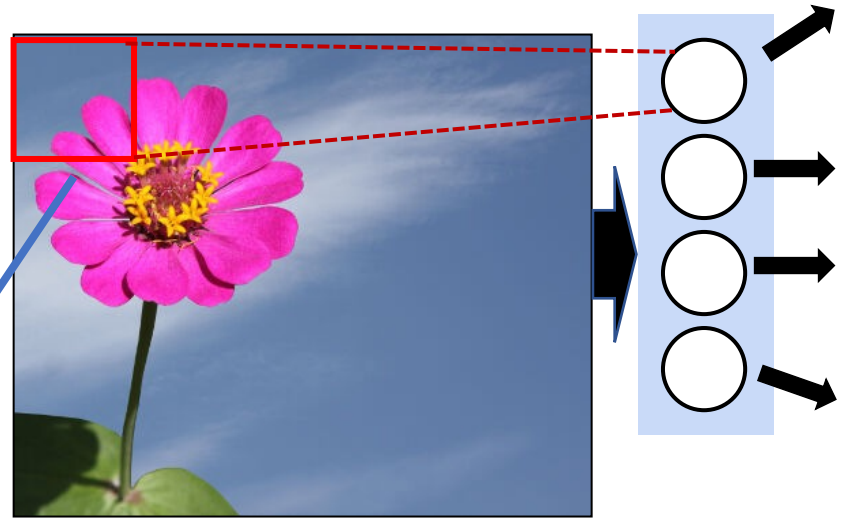
Distribute over 2 layers



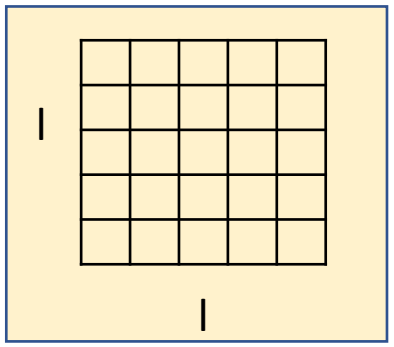
Distribute over 3 layers



Convolutional Neural Networks (CNNs)



A set of pixels → Concatenate all pixels → Vector



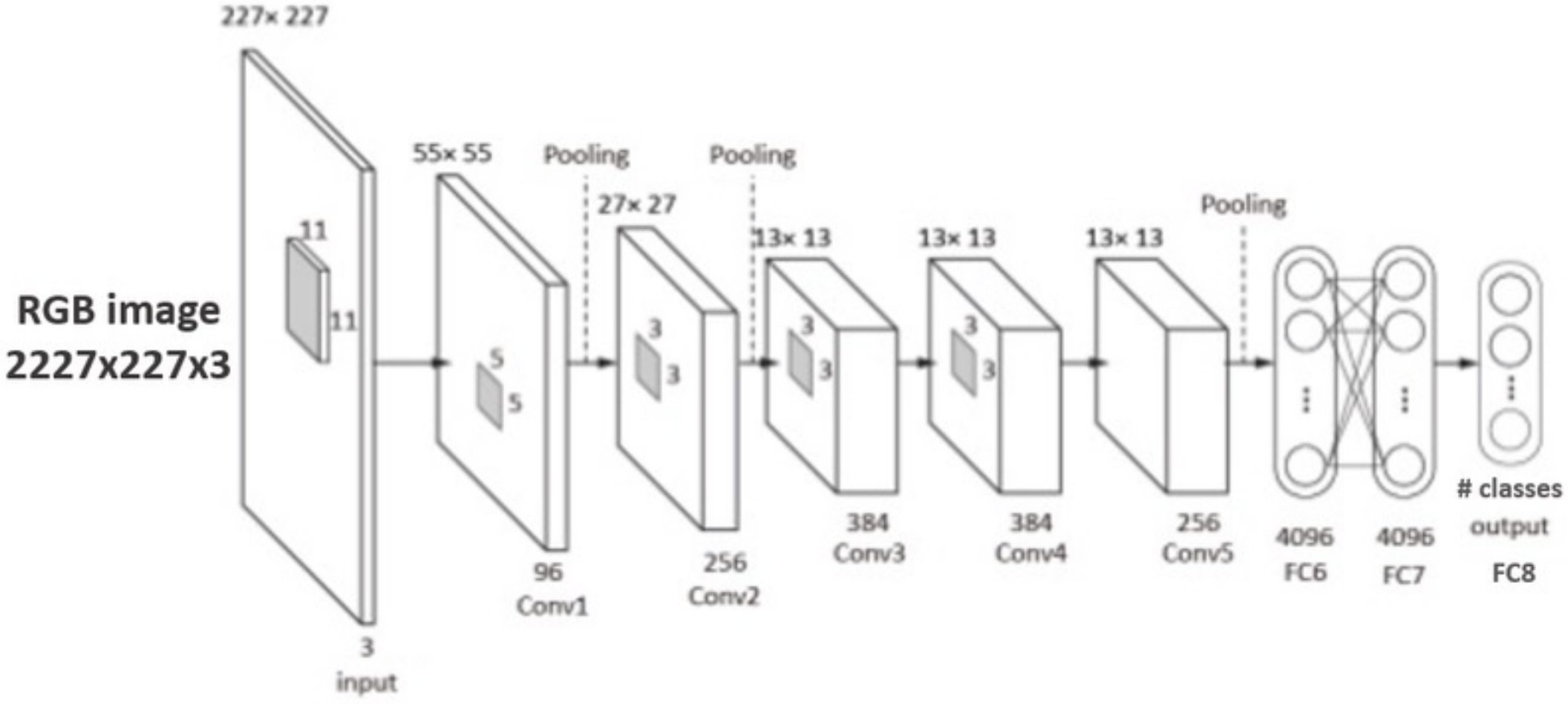
Filter

$$\begin{bmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_{lxl} \end{bmatrix}$$

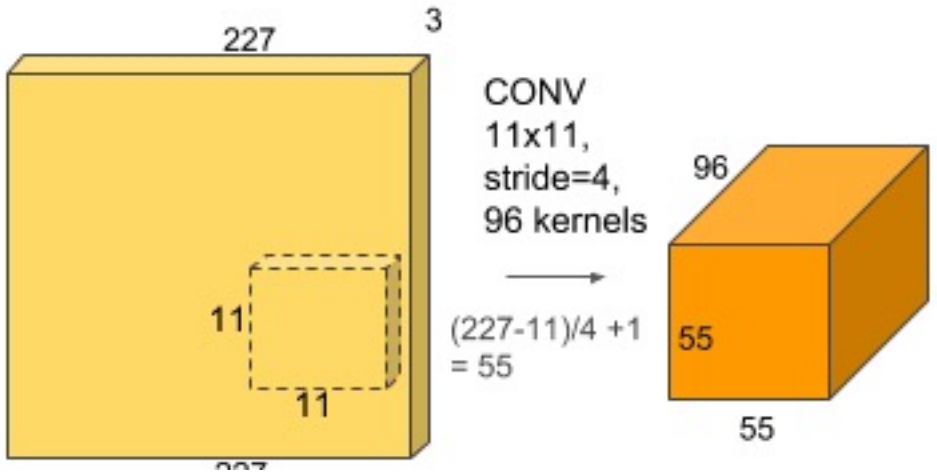
w_1 ... w_{lxl}

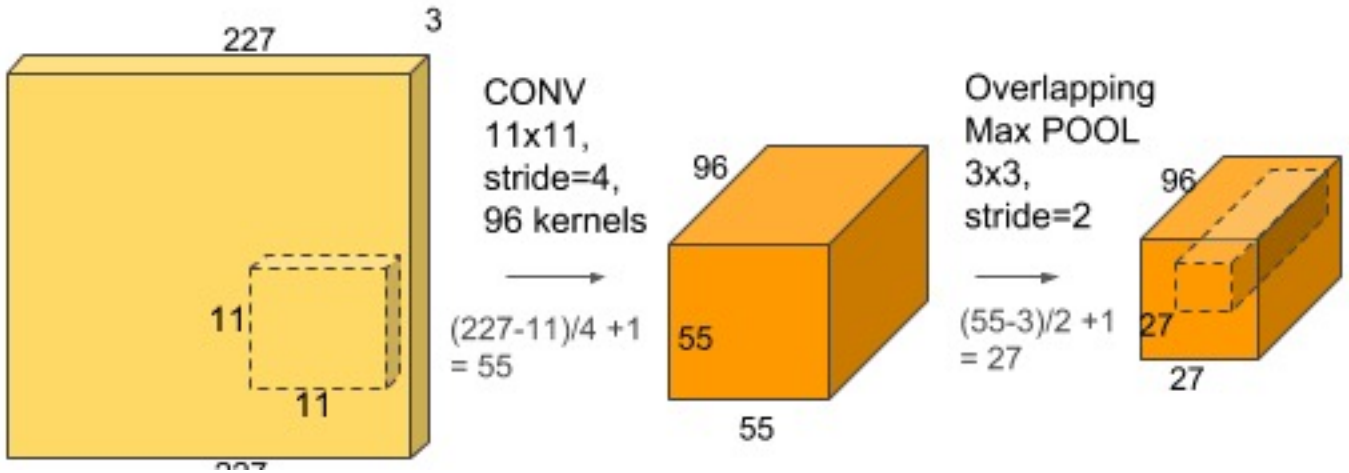
$$reLU\left(\sum w_1 x_1 + w_2 x_2 + \cdots + w_{kxk} x_{kxk}\right)$$

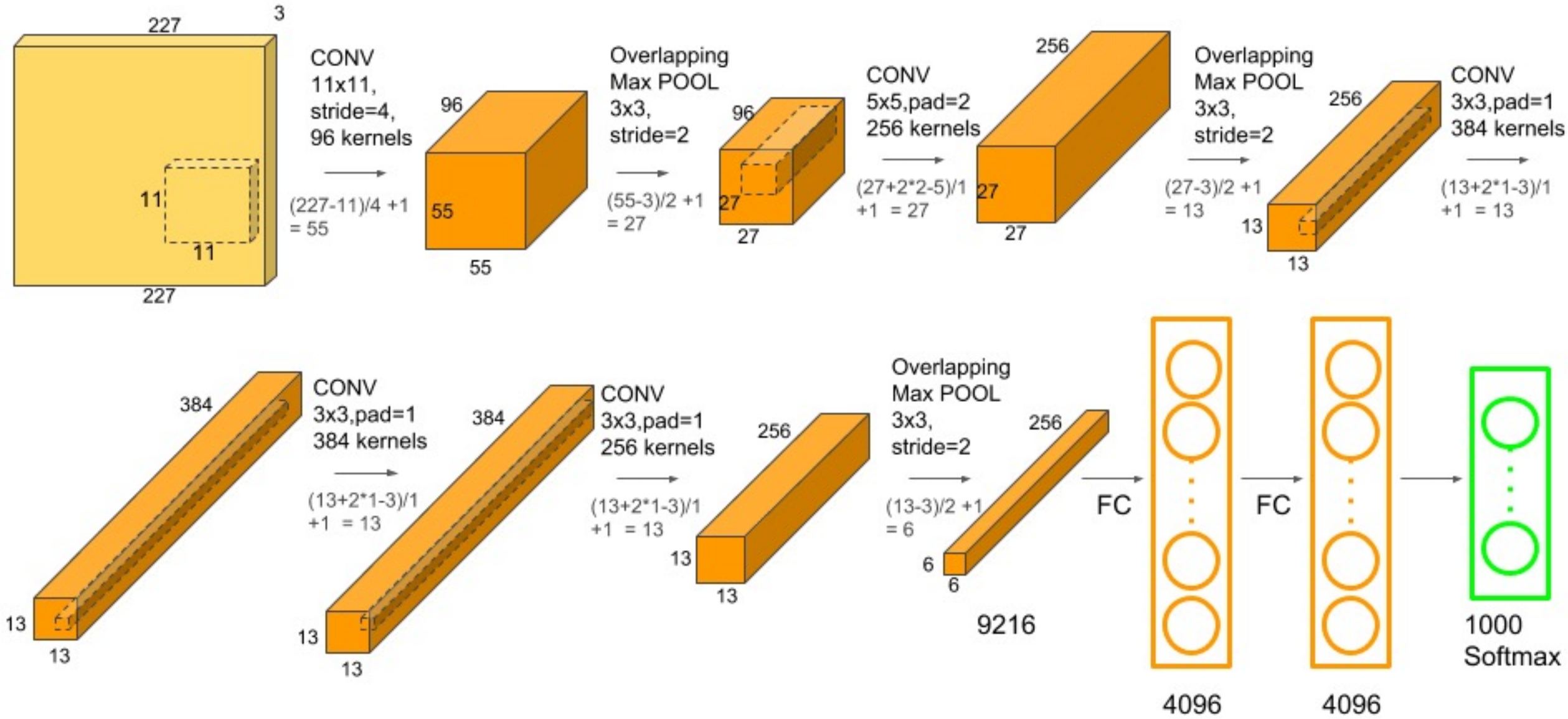
AlexNet



Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In NIPS, 2012.

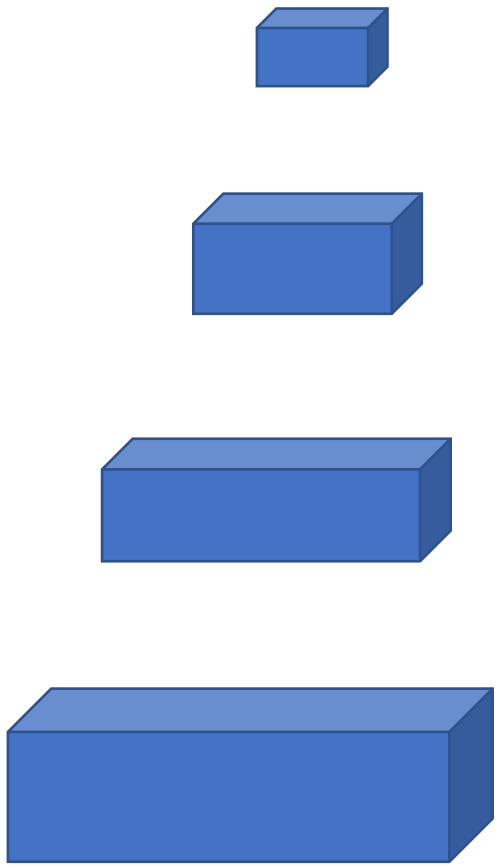

227
3
11
11
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55


227
3
11
11
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27


227
3
227
11
11
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

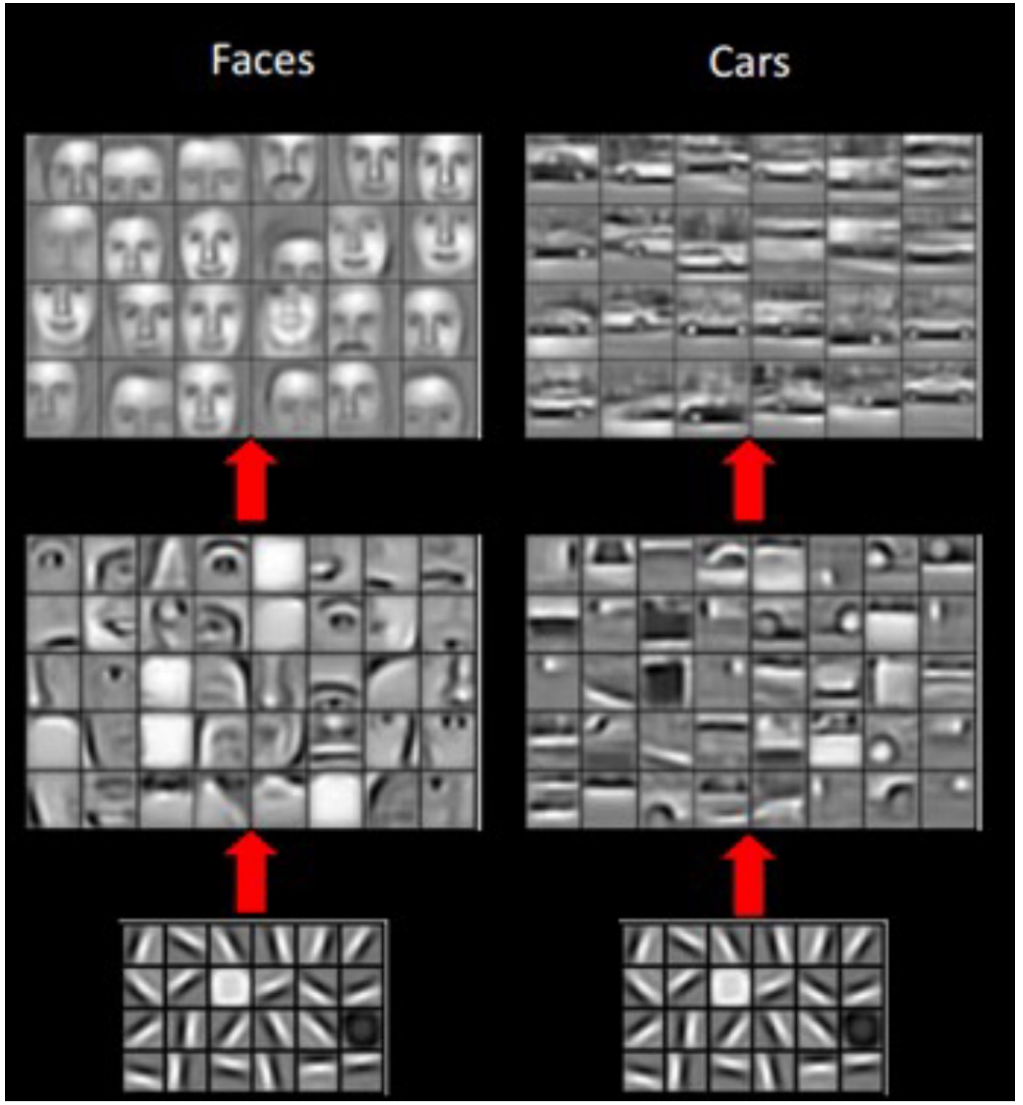
Output

High level feature



Low level feature

Input



Recap

- Perceptron
- Multi-layer Perceptron
- Convolutional Neural Network
  - Detection ?
  - Classification ?
  - Prediction ?
  - Language Translation?

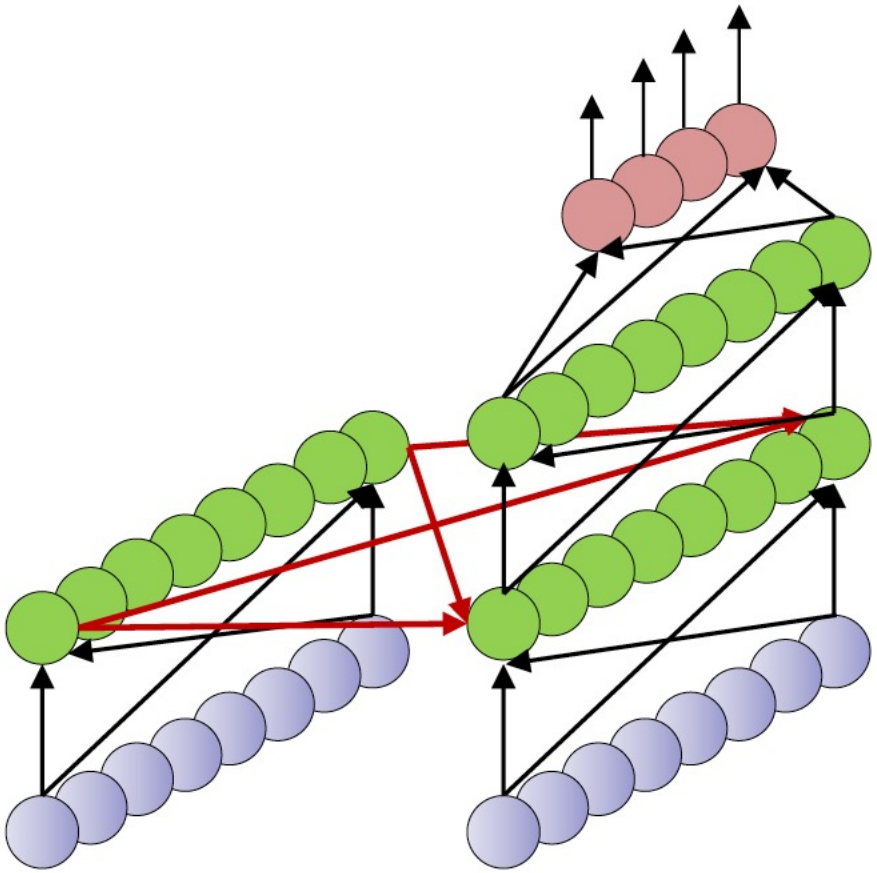
Recurrent Neural Networks

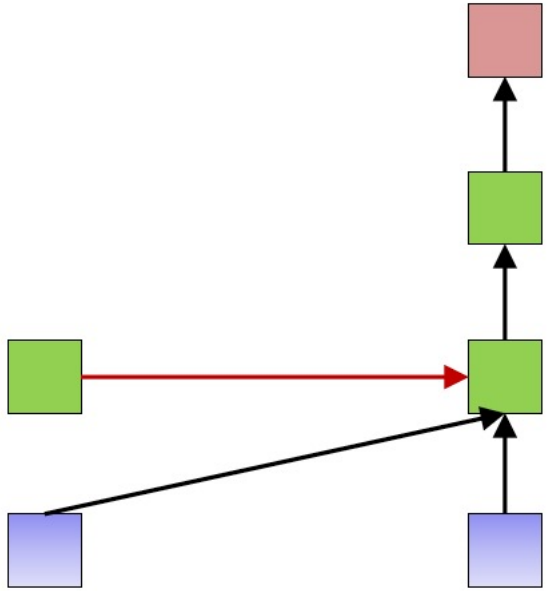
Stock Prediction:

Inputs: are vectors.

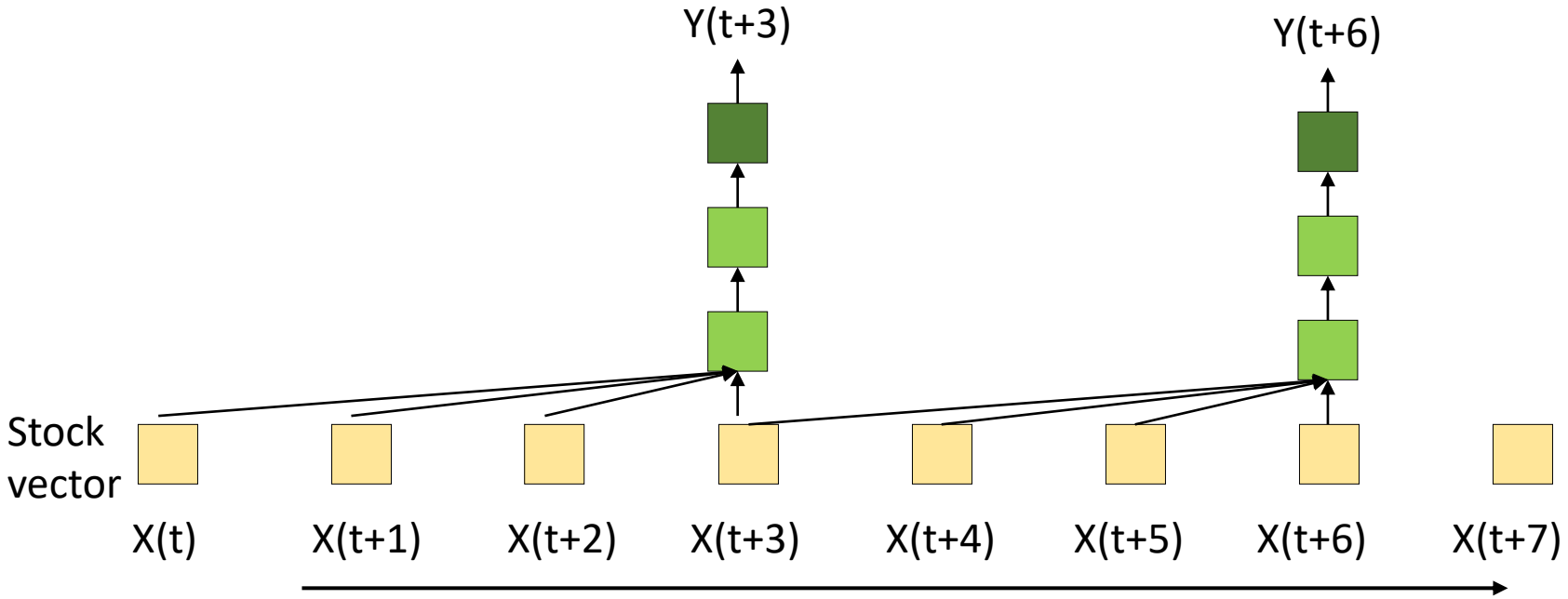
Output: may be scalar or vector

Representation Shortcut



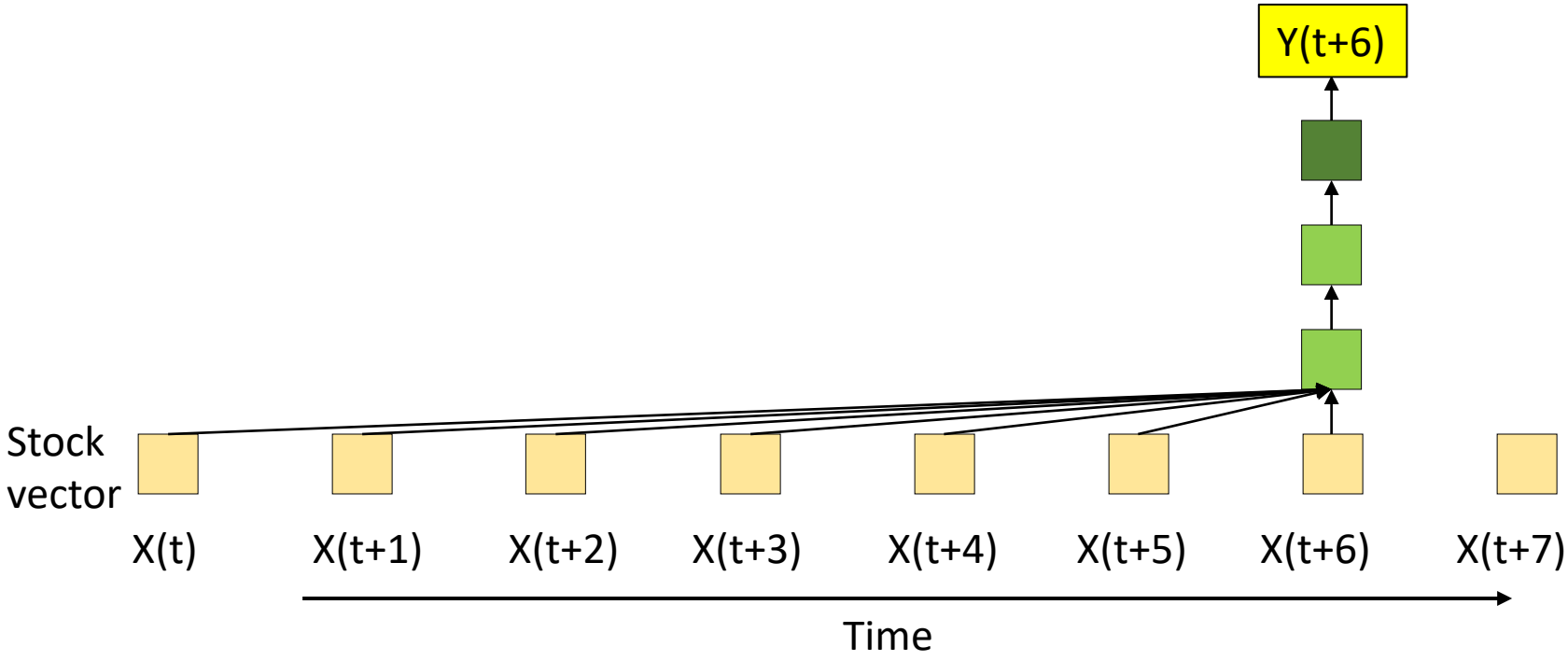


The stock predictor



$$Y_t = f(X_t, X_{t-1}, \ldots, X_{t-N})$$

The stock predictor

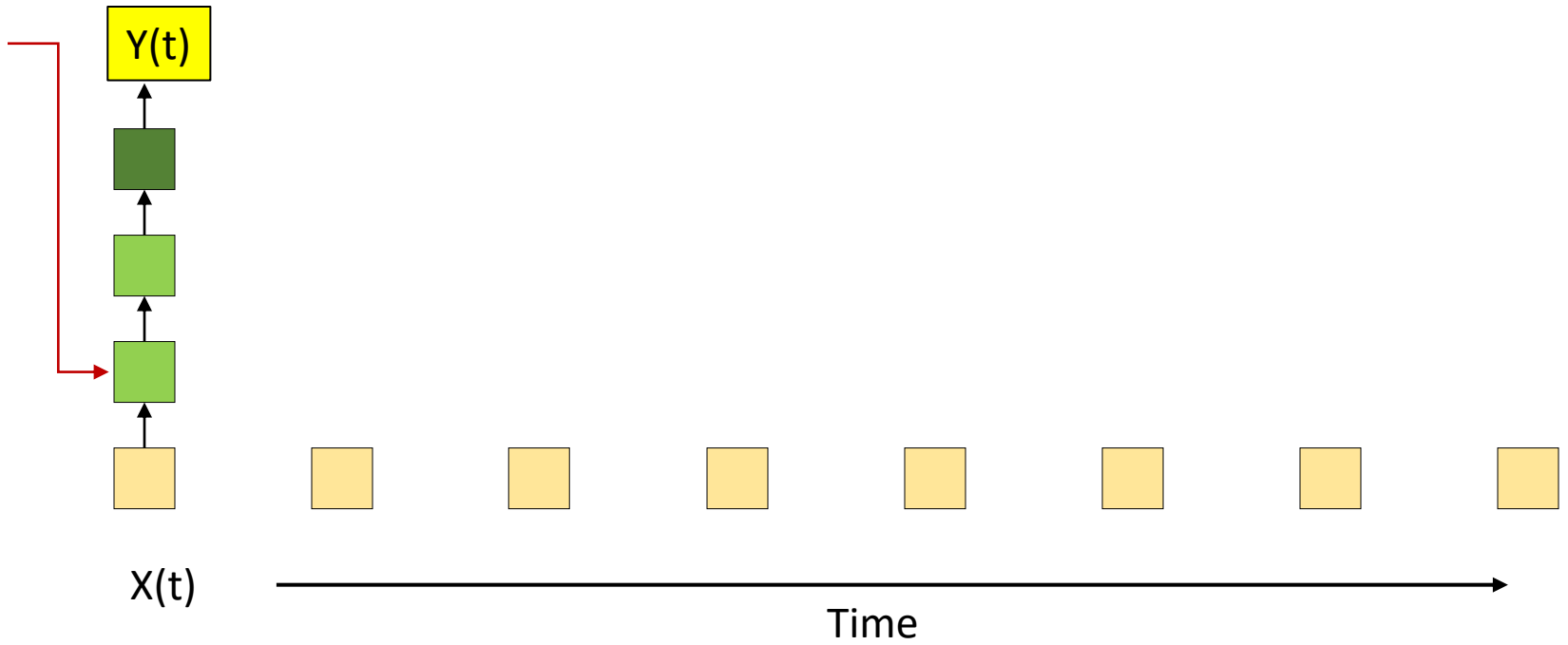


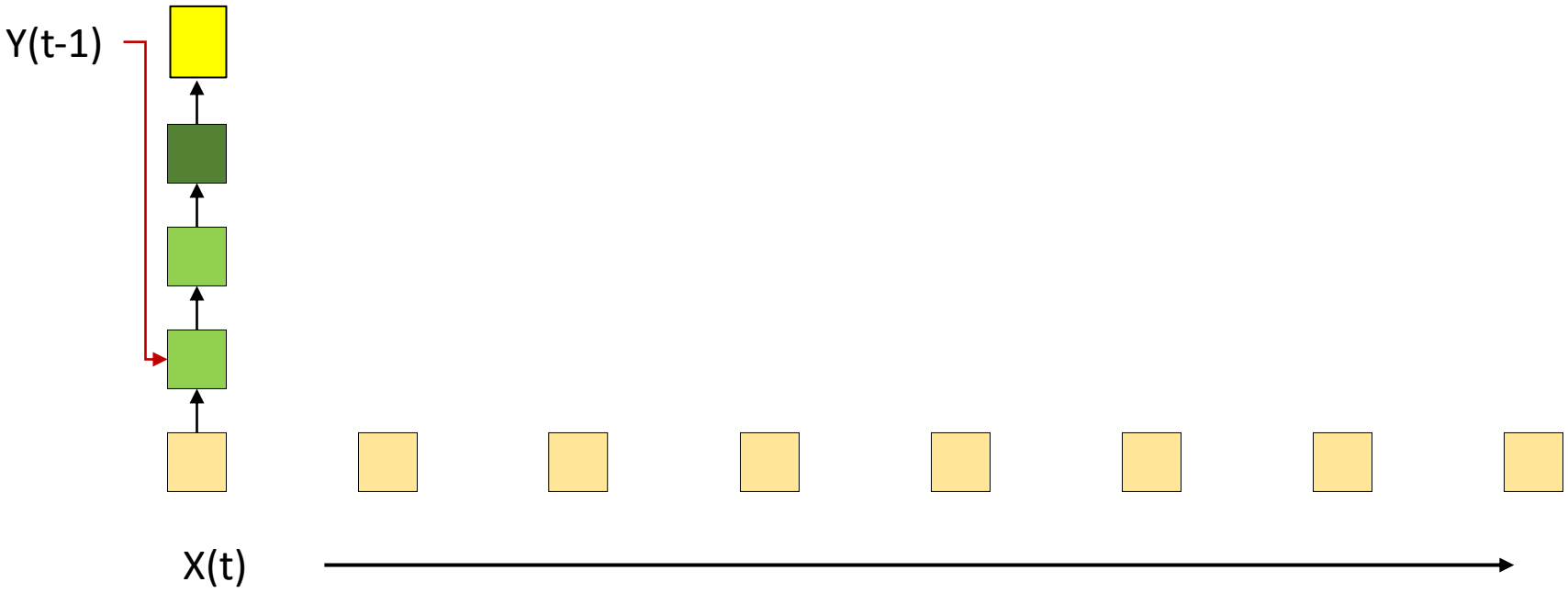
$$Y_t = f(X_t, X_{t-1}, \dots, X_{t-\infty})$$

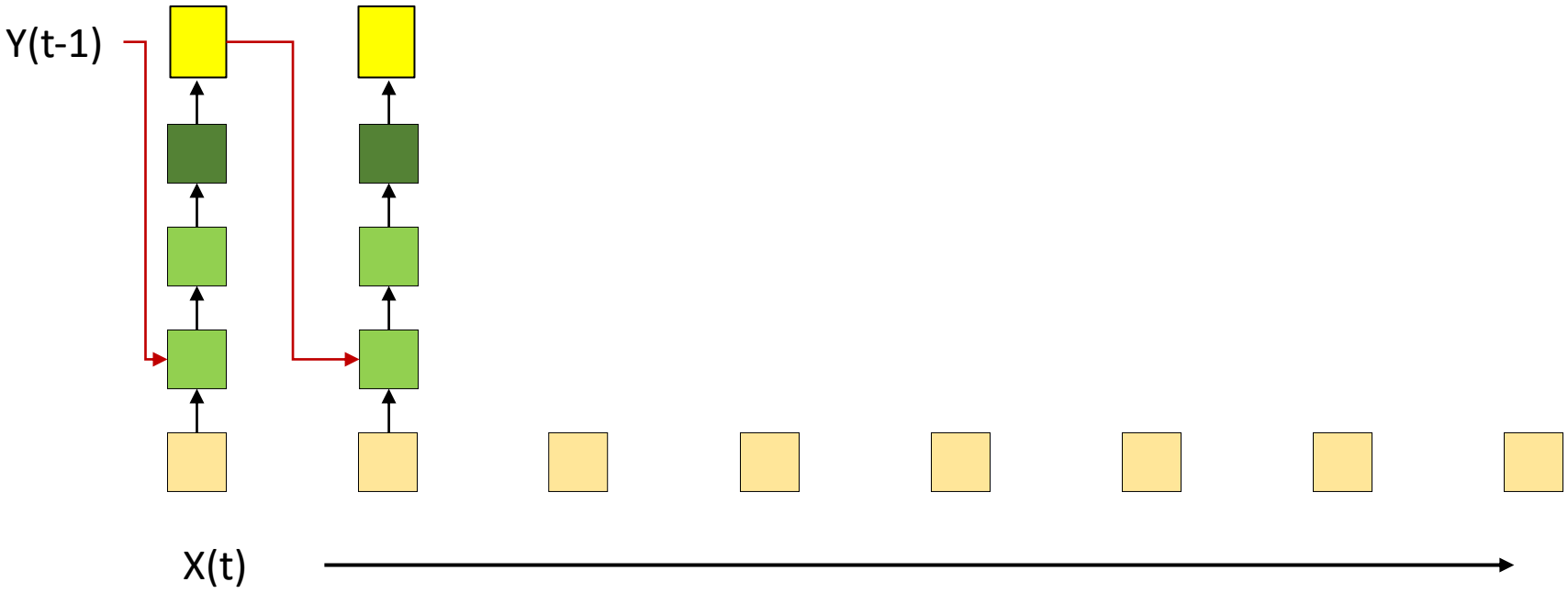
$$Y_t = f(X_t, X_{t-1}, \dots, X_{t-\infty})$$

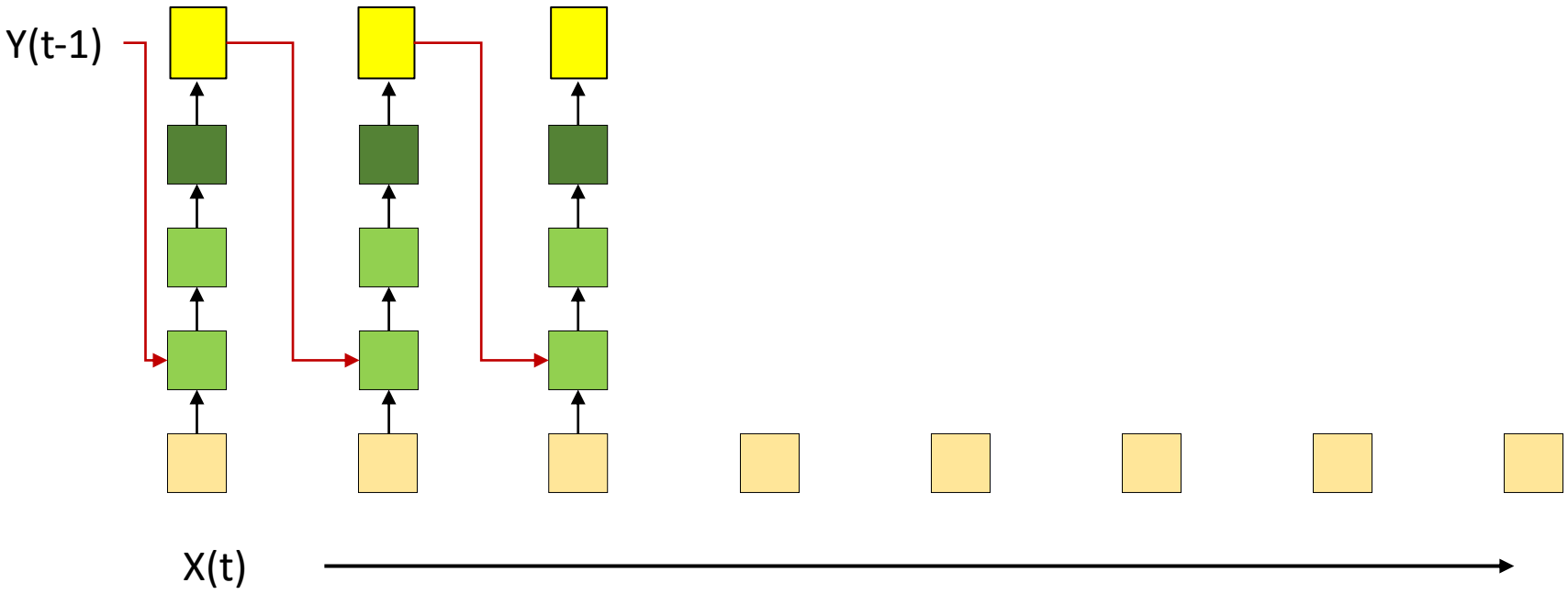
An input at X_0 at $t = 0$ produces Y_0

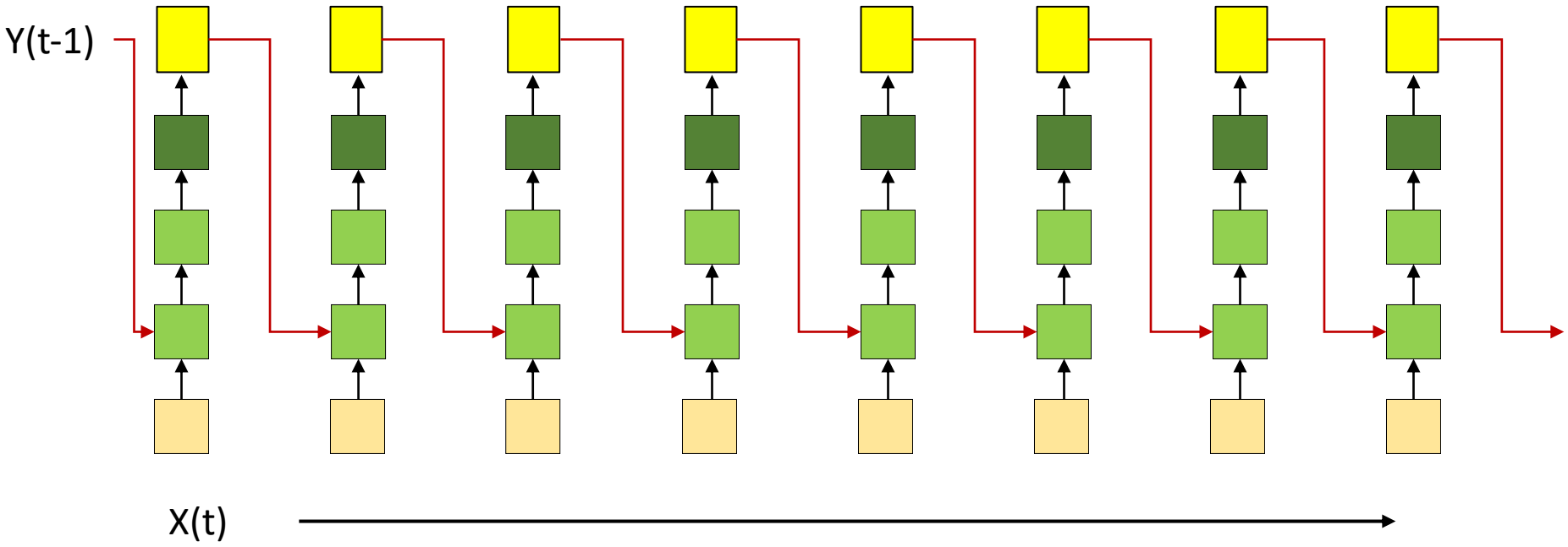
Y_0 produces Y_1 which produces Y_2 and so on until Y_∞

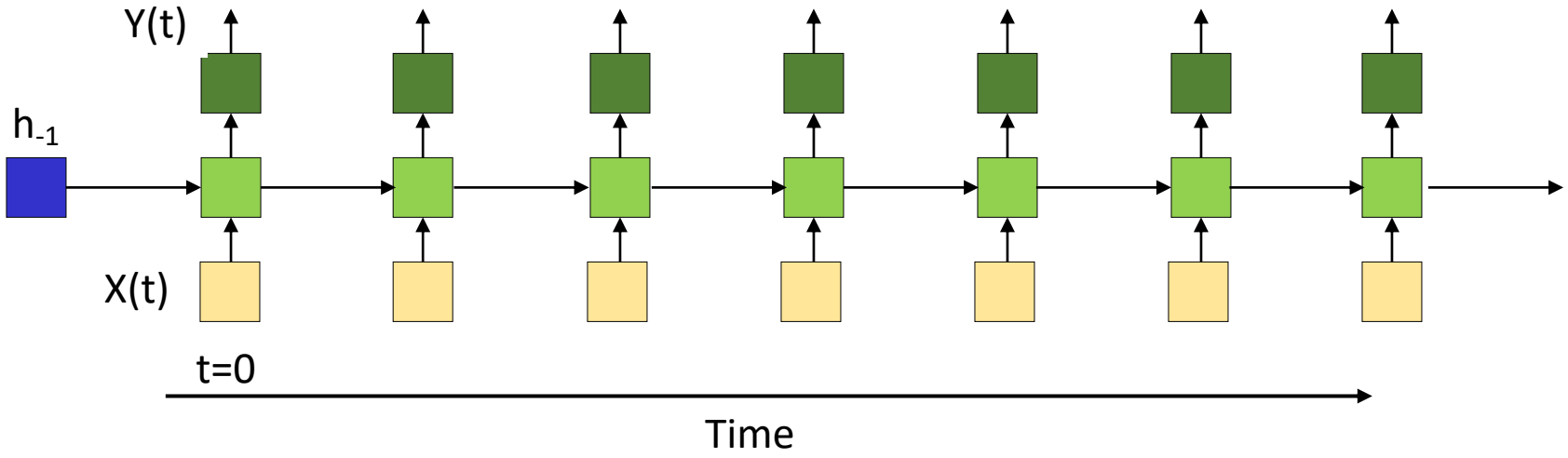

Y(t)
X(t)
Time


Y(t-1)
X(t)


Y(t-1)
X(t)

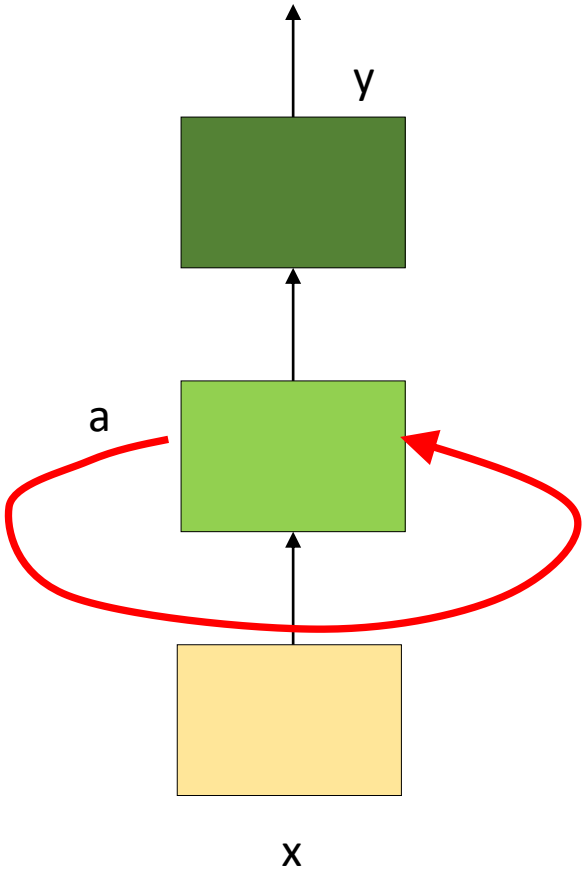

Y(t-1)
X(t)

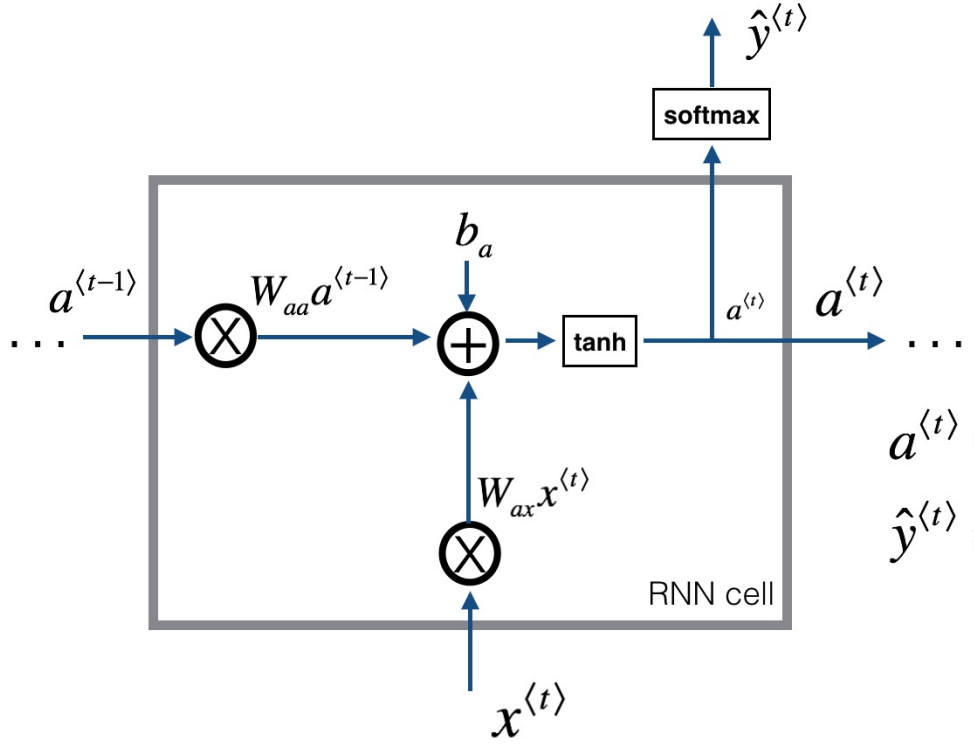

Y(t-1)
X(t)



- Recurrent neural network
- All columns are identical
- *An input at t=0 affects outputs forever*

Recurrent Neural Networks



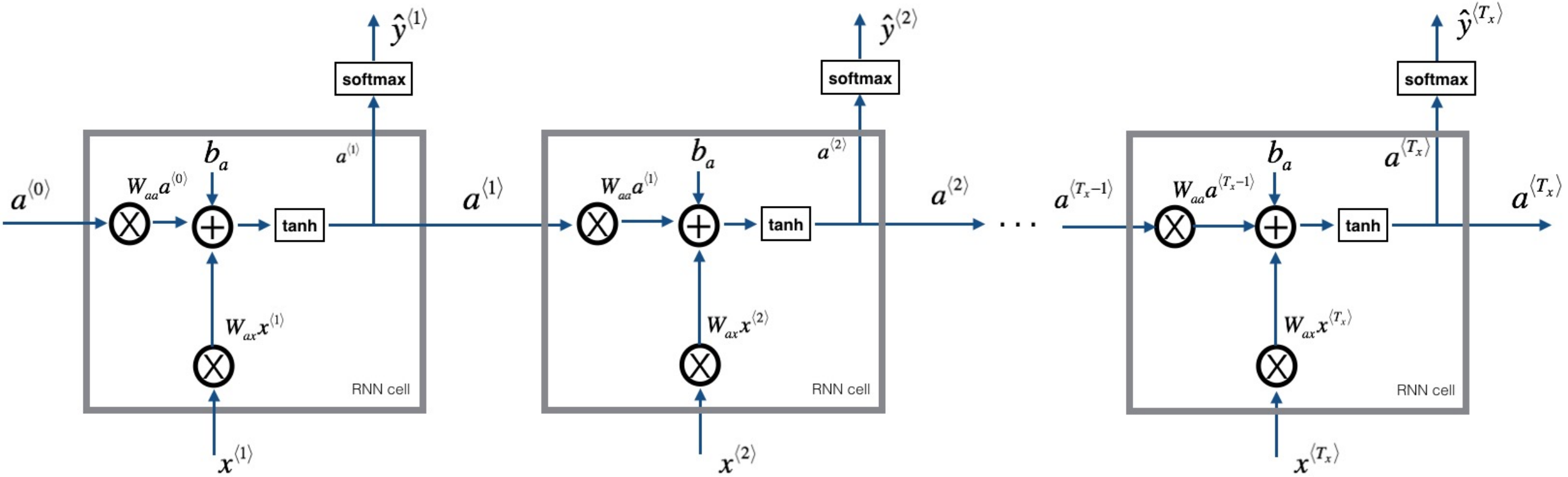


$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

Image source: https://datascience-enthusiast.com/DL/Building_a_Recurrent_Neural_Network-Step_by_Step_v1.html

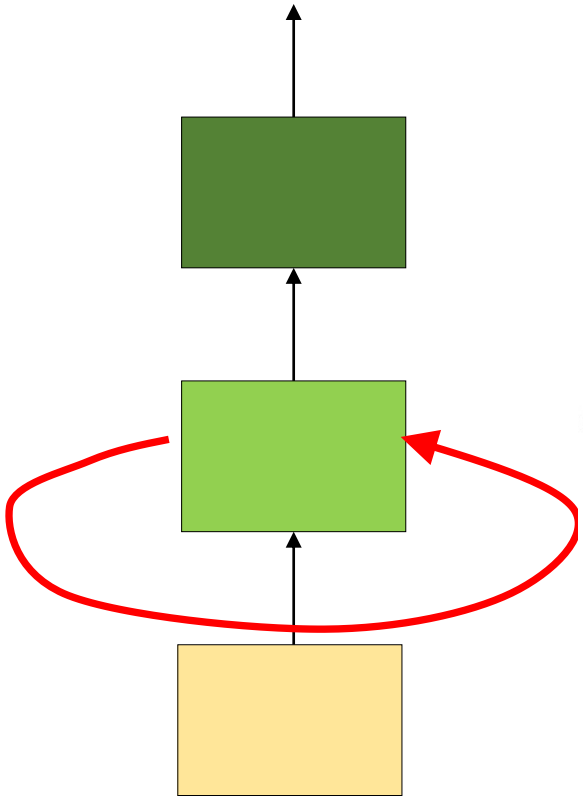
Recurrent Neural Networks

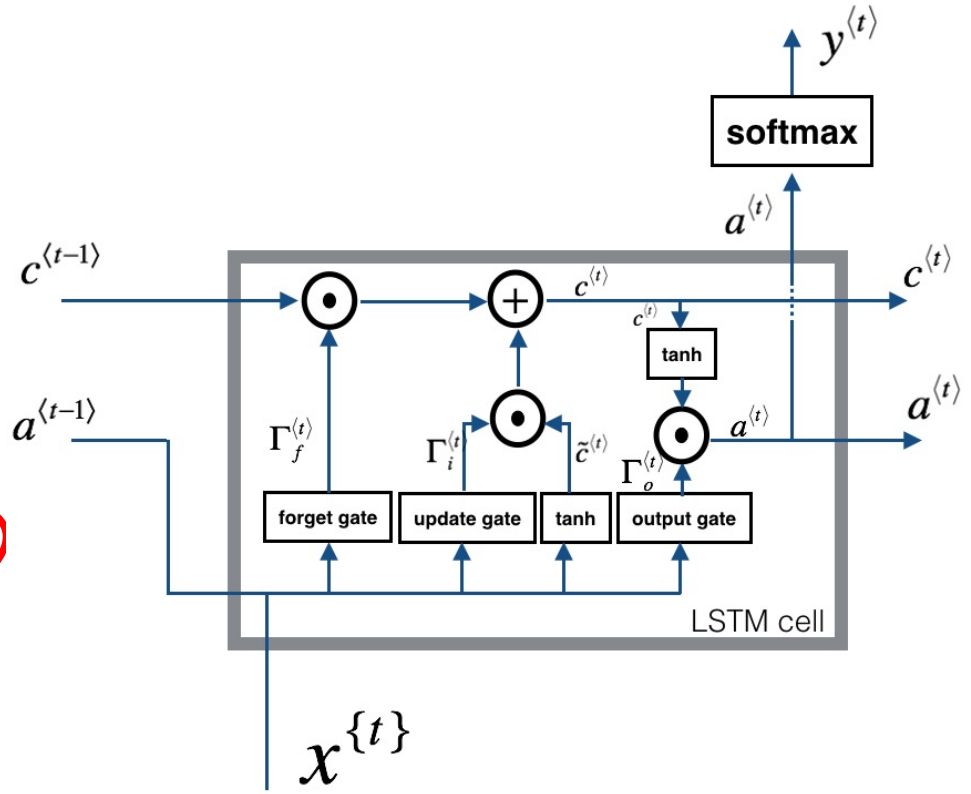


$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

LSTM





$$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$

$$\Gamma_u^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$

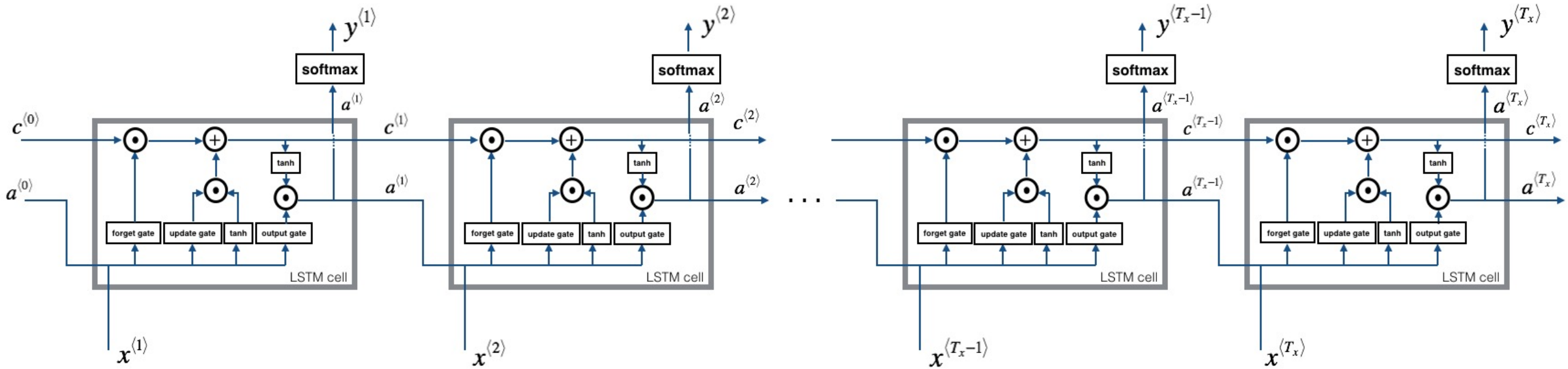
$$\tilde{c}^{\{t\}} = \tanh(W_C[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_C)$$

$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle}$$

$$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$$

$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh(c^{\langle t \rangle})$$

LSTM


$y^{\langle 1 \rangle}$
softmax
$a^{\langle 1 \rangle}$
$c^{\langle 0 \rangle}$
$a^{\langle 0 \rangle}$
tanh
forget gate
update gate
tanh
output gate
LSTM cell
$c^{\langle 1 \rangle}$
$a^{\langle 1 \rangle}$
$x^{\langle 1 \rangle}$
$y^{\langle 2 \rangle}$
softmax
$a^{\langle 2 \rangle}$
$c^{\langle 2 \rangle}$
tanh
forget gate
update gate
tanh
output gate
LSTM cell
$a^{\langle 2 \rangle}$
$x^{\langle 2 \rangle}$
. . .
$y^{\langle T_x - 1 \rangle}$
softmax
$a^{\langle T_x - 1 \rangle}$
$c^{\langle T_x - 1 \rangle}$
tanh
forget gate
update gate
tanh
output gate
LSTM cell
$a^{\langle T_x - 1 \rangle}$
$x^{\langle T_x - 1 \rangle}$
$y^{\langle T_x \rangle}$
softmax
$a^{\langle T_x \rangle}$
$c^{\langle T_x \rangle}$
tanh
forget gate
update gate
tanh
output gate
LSTM cell
$a^{\langle T_x \rangle}$
$x^{\langle T_x \rangle}$

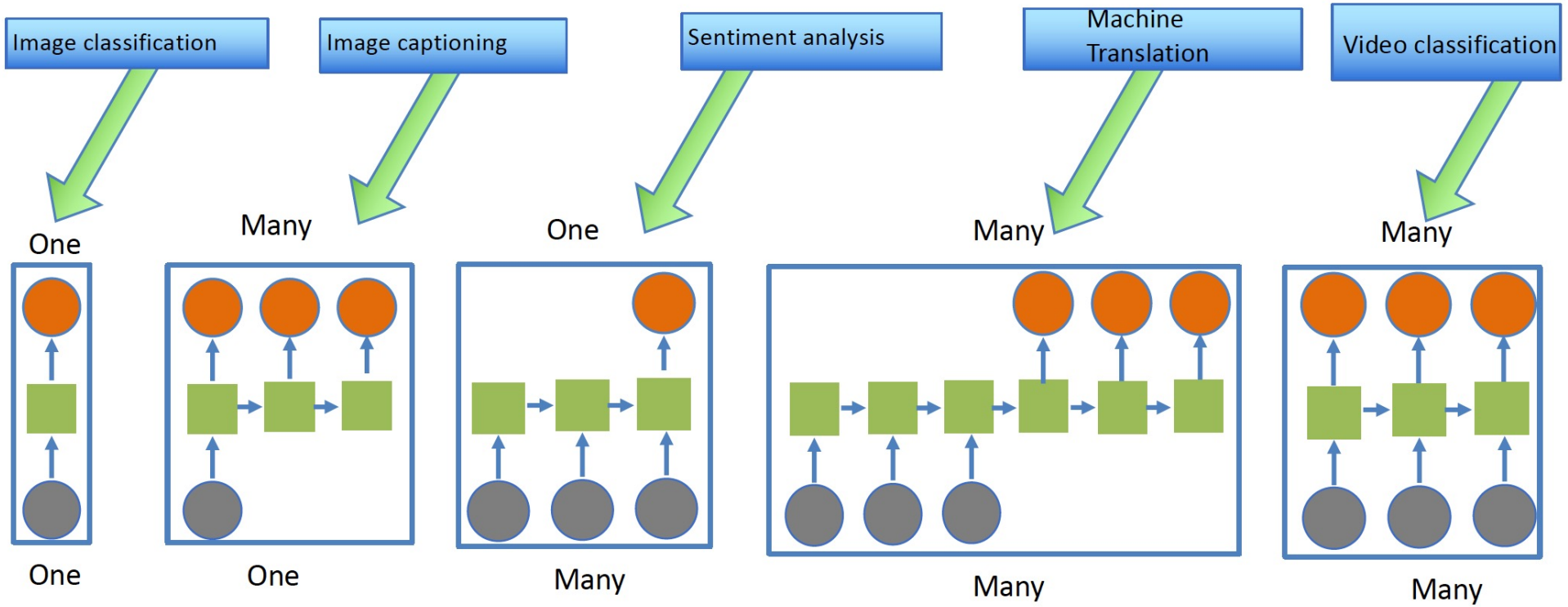
RNN VS. LSTM VS. GRU

https://jhui.github.io/2017/03/15/RNN-LSTM-GRU/

https://datascience-enthusiast.com/DL/Building_a_Recurrent_Neural_Network-Step_by_Step_v1.html

https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/

Variants on recurrent nets



Input

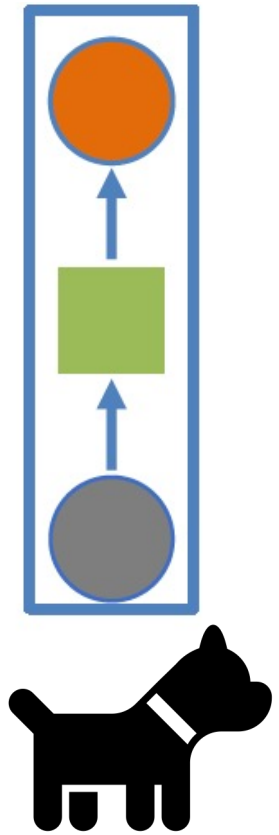
Recurrent core (Hidden unit)

Output

One-One

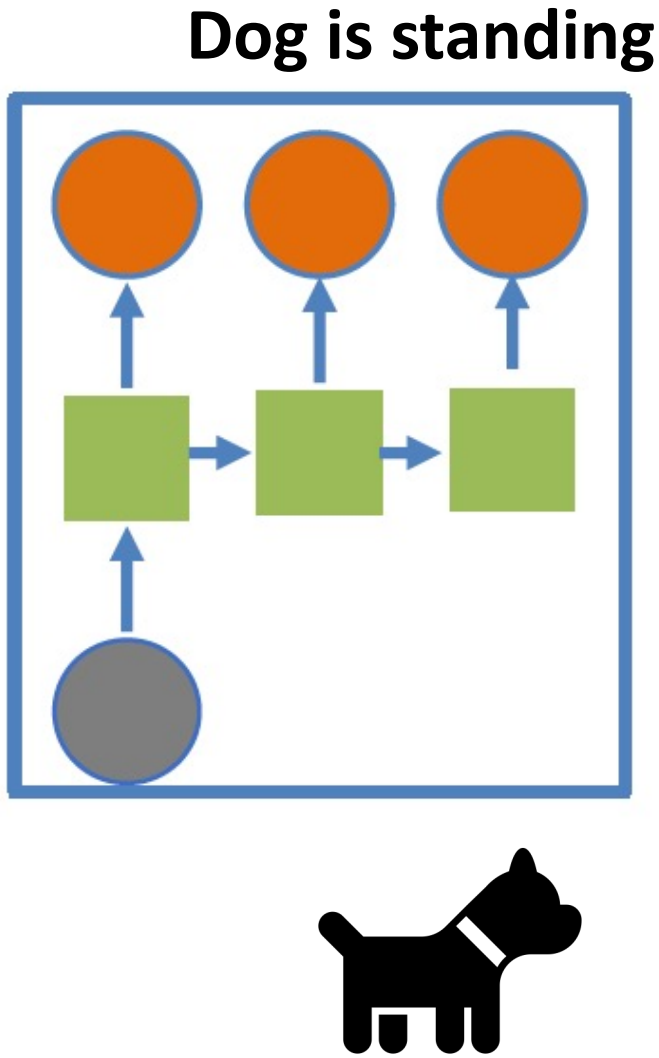
Image Classification

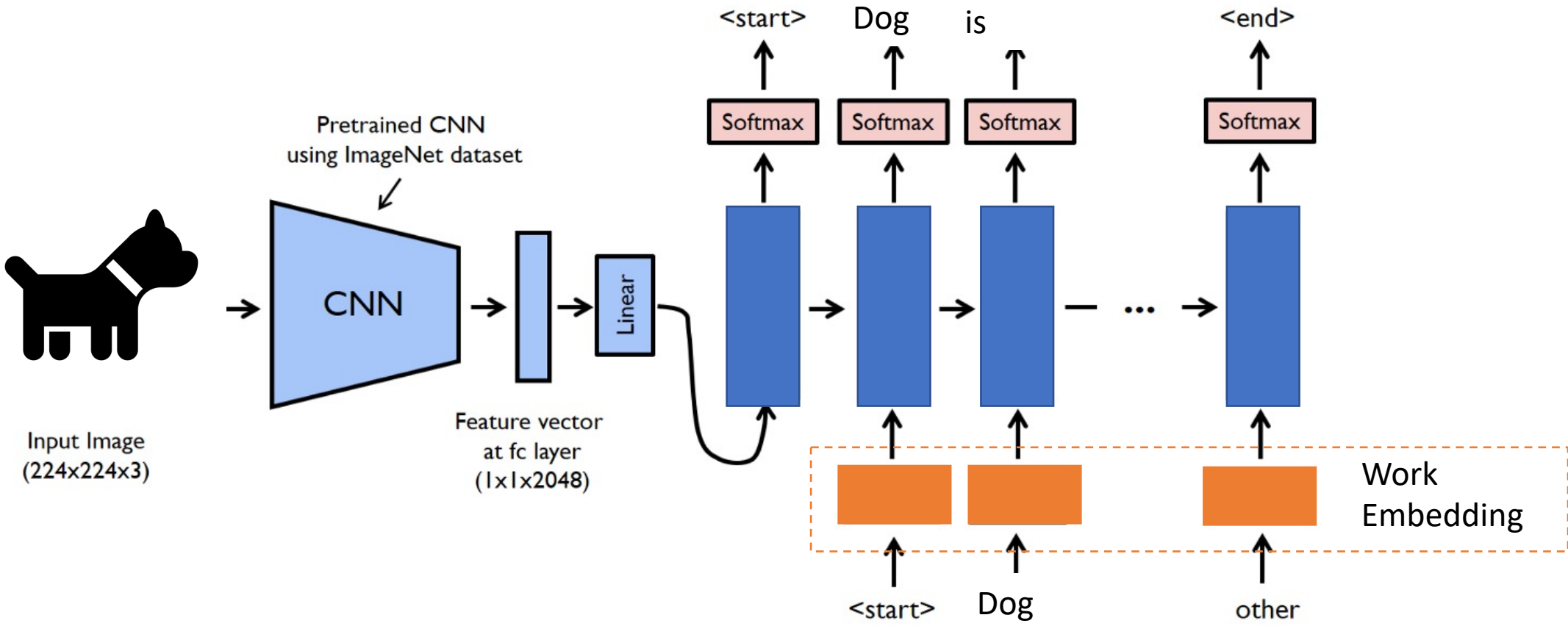
Label: dog



One-Many

Image Captioning

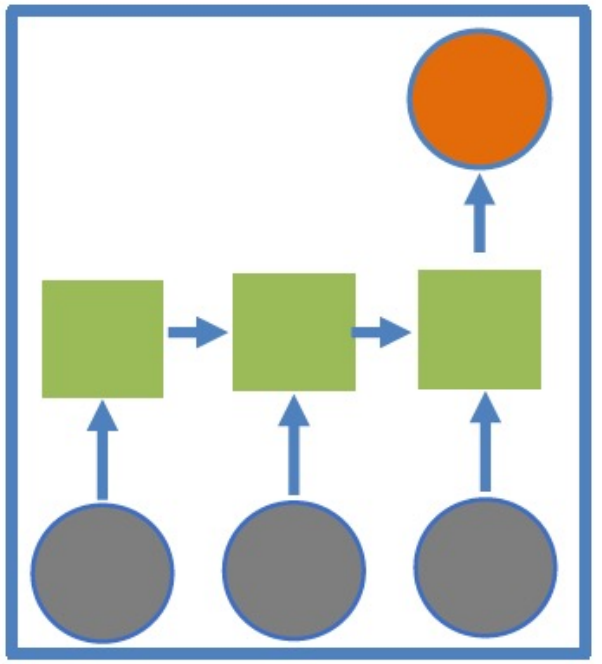



<start>
Dog
is
<end>
Softmax
Softmax
Softmax
Softmax
Pretrained CNN
using ImageNet dataset
CNN
Linear
Input Image
(224x224x3)
Feature vector
at fc layer
(1x1x2048)
Work
Embedding
<start>
Dog
other

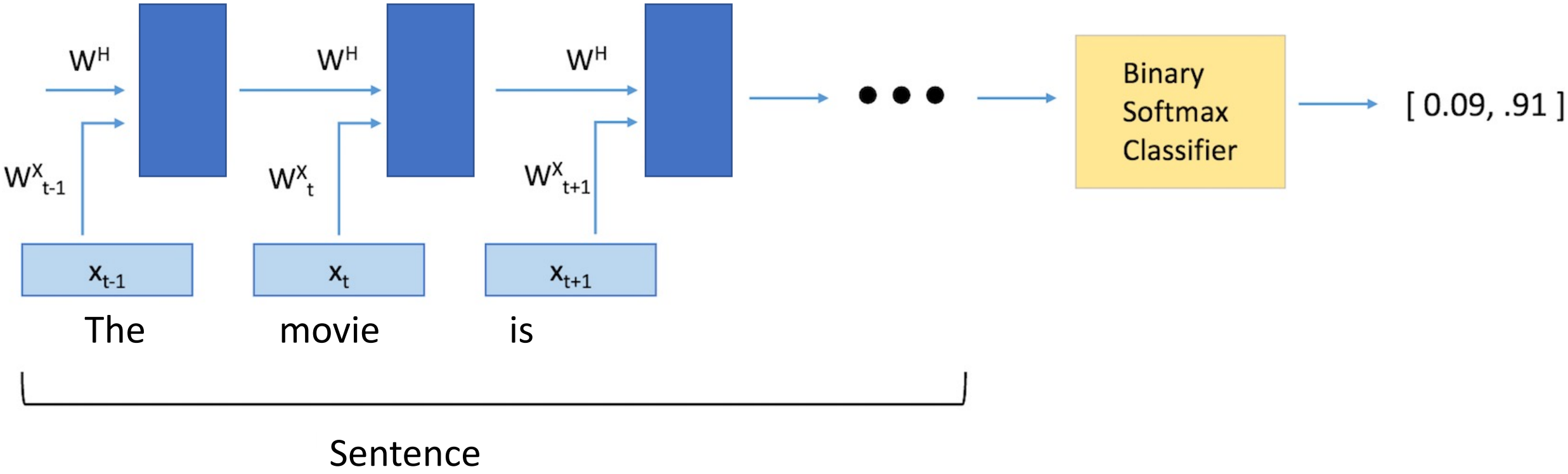
Many-One

Sentiment Analysis

Ranking: 0.8

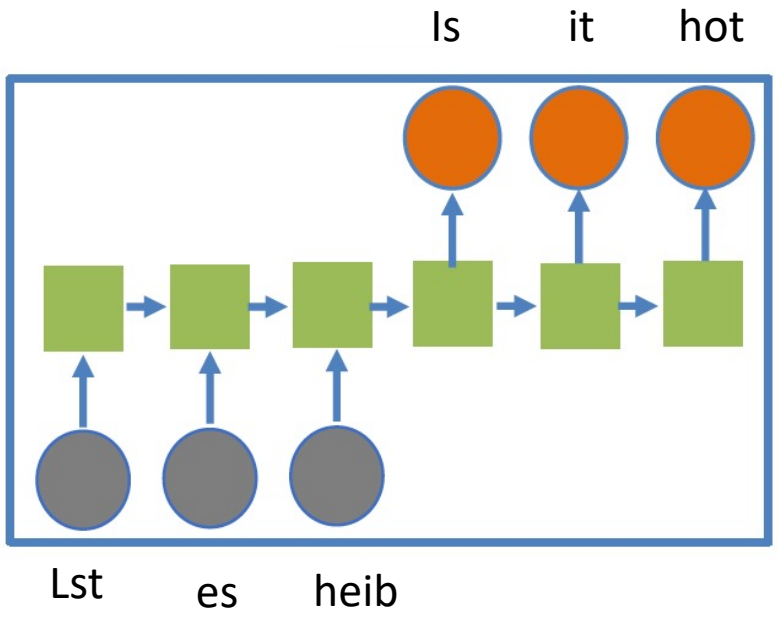


The movie is interesting

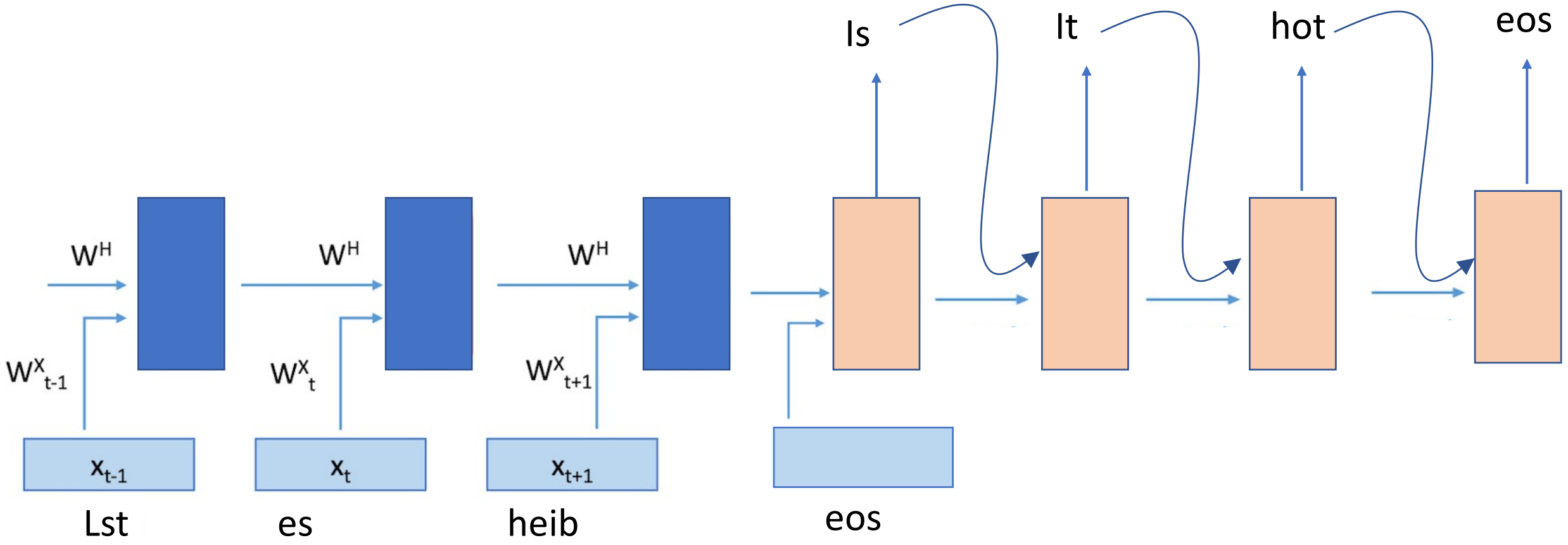

W^H
W^H
W^H
W^X_{t-1}
W^X_t
W^X_{t+1}
x_{t-1}
x_t
x_{t+1}
The
movie
is
Sentence
Binary Softmax Classifier
[0.09, .91]

Many-Many

Language Translation



Many-Many



Vanilla RNN

```
model = keras.models.Sequential([
    keras.layers.SimpleRNN(2,
        return_sequences=True,
        input_shape=[None, 1]),
    keras.layers.Dropout(0.3),
    keras.layers.SimpleRNN(1),
    keras.layers.Dense(1)
])

model.compile(
  loss='mse',
  optimizer='Adam',
  metrics=['mae', 'mse'],
)
```

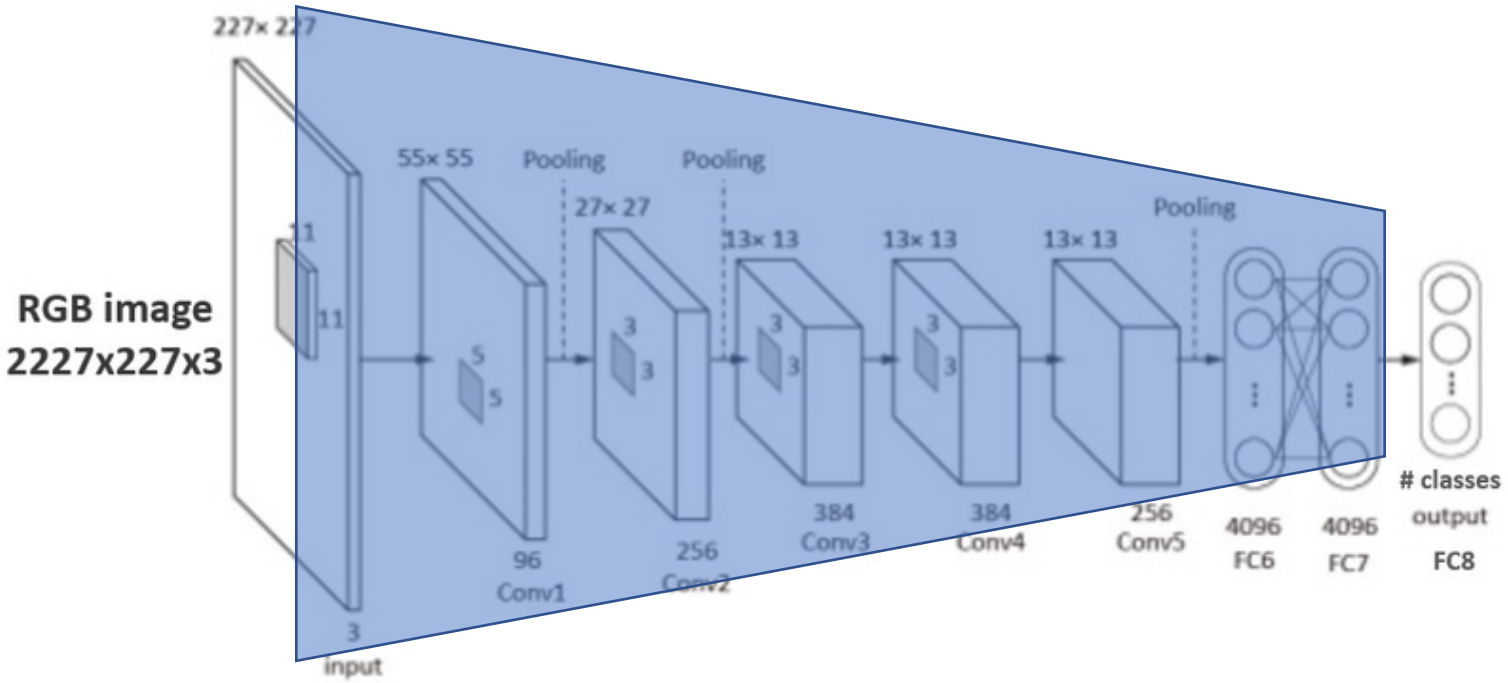
LSTM

```
model = keras.models.Sequential([
    keras.layers.LSTM(20,
        return_sequences=True,
        input_shape=[None, 1]),
    keras.layers.Dropout(0.2),
    keras.layers.LSTM(10),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(1)
])

model.compile(
  loss='mse',
  optimizer='Adam',
  metrics=['mae', 'mse'],
)
```

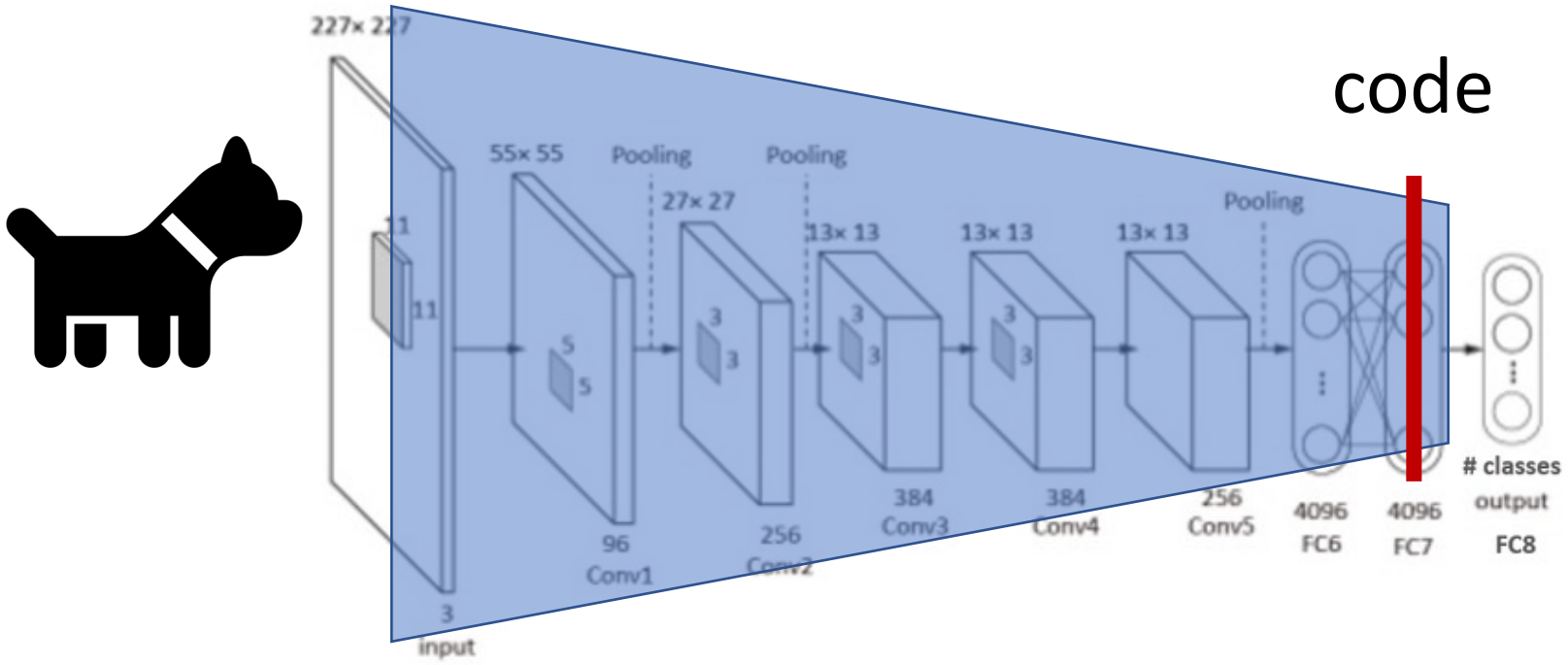
GRU

```
model = keras.models.Sequential([
    keras.layers.Conv1D(filters=20, kernel_size=4,
        strides=2, padding="valid",
                           input_shape=[None, 1]),
    keras.layers.GRU(2, input_shape=[None, 1],
        activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(1),
])

model.compile(
  loss='mse',
  optimizer='Adam',
  metrics=['mae', 'mse'],
)
```

Autoencoder

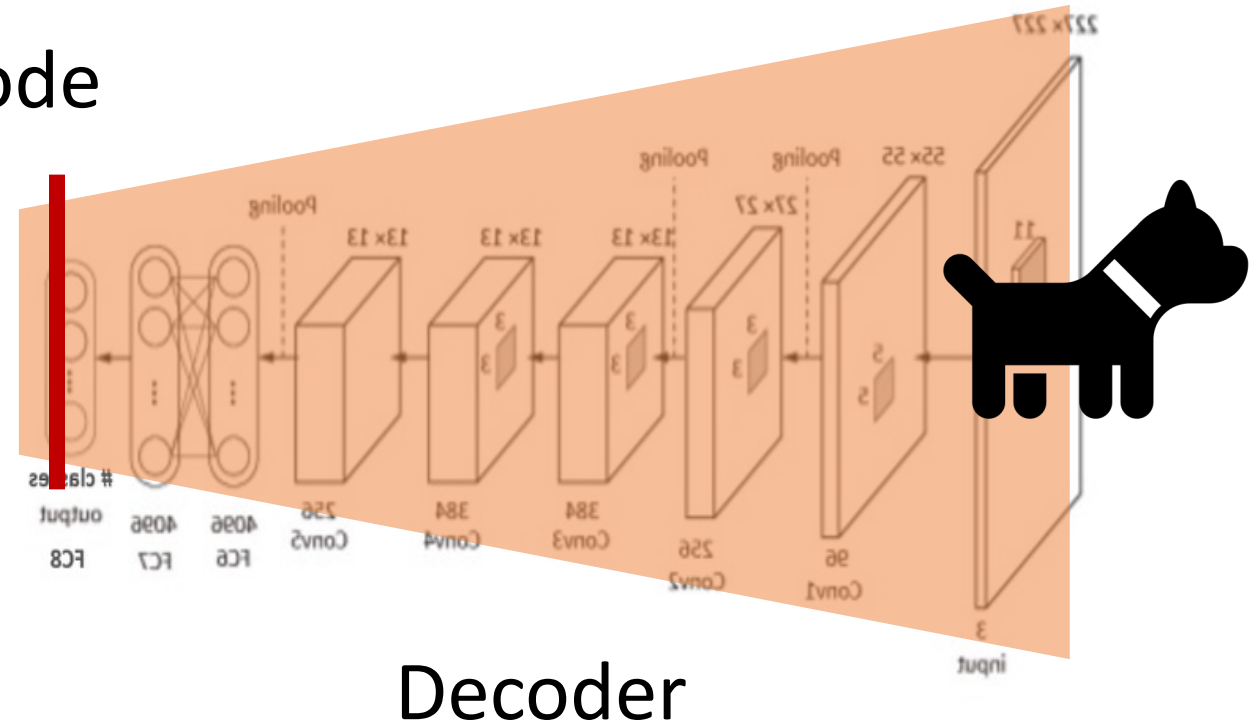

RGB image
2227x227x3
227× 227
11
11
3
input
55× 55
5
5
96
Conv1
Pooling
27× 27
3
3
256
Conv2
Pooling
13× 13
3
3
384
Conv3
13× 13
3
3
384
Conv4
13× 13
256
Conv5
Pooling
4096
FC6
4096
FC7
classes
output
FC8

Encoder


code
227× 227
55× 55
Pooling
Pooling
27× 27
13× 13
13× 13
13× 13
Pooling
11
11
5
5
3
3
3
3
3
3
3
3
96
Conv1
256
384
Conv3
384
Conv4
256
Conv5
4096
FC6
4096
FC7
classes
output
FC8
input

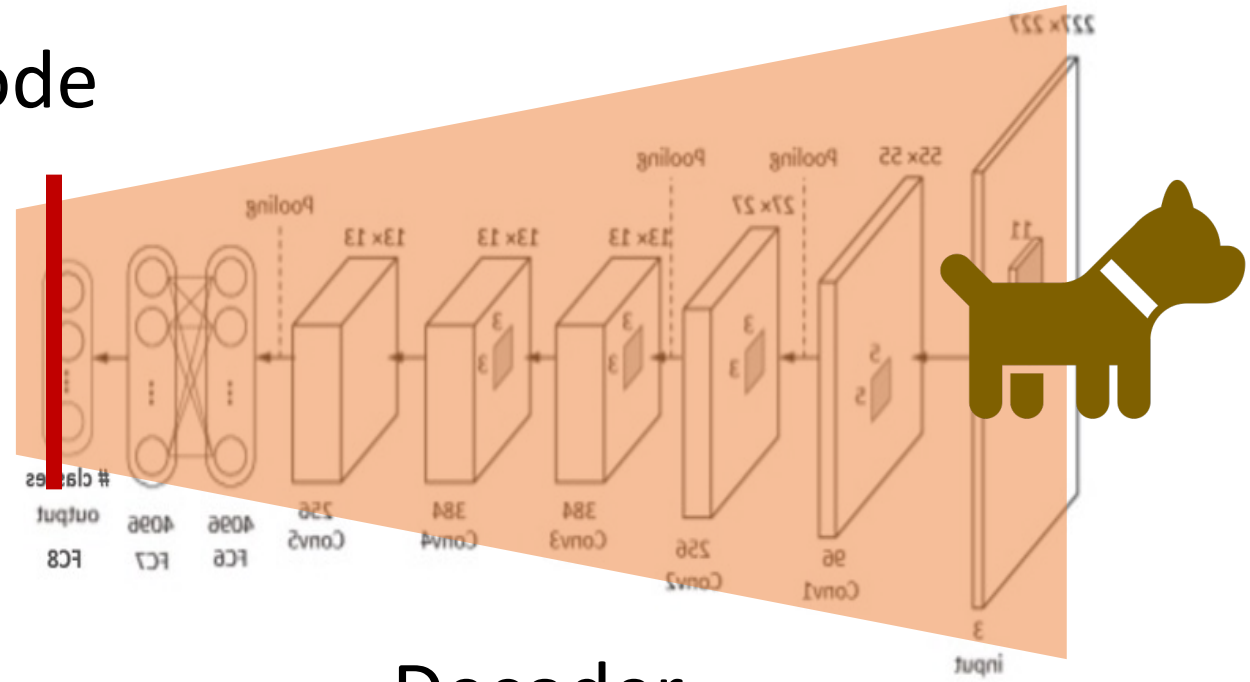
Encoder

code



Decoder

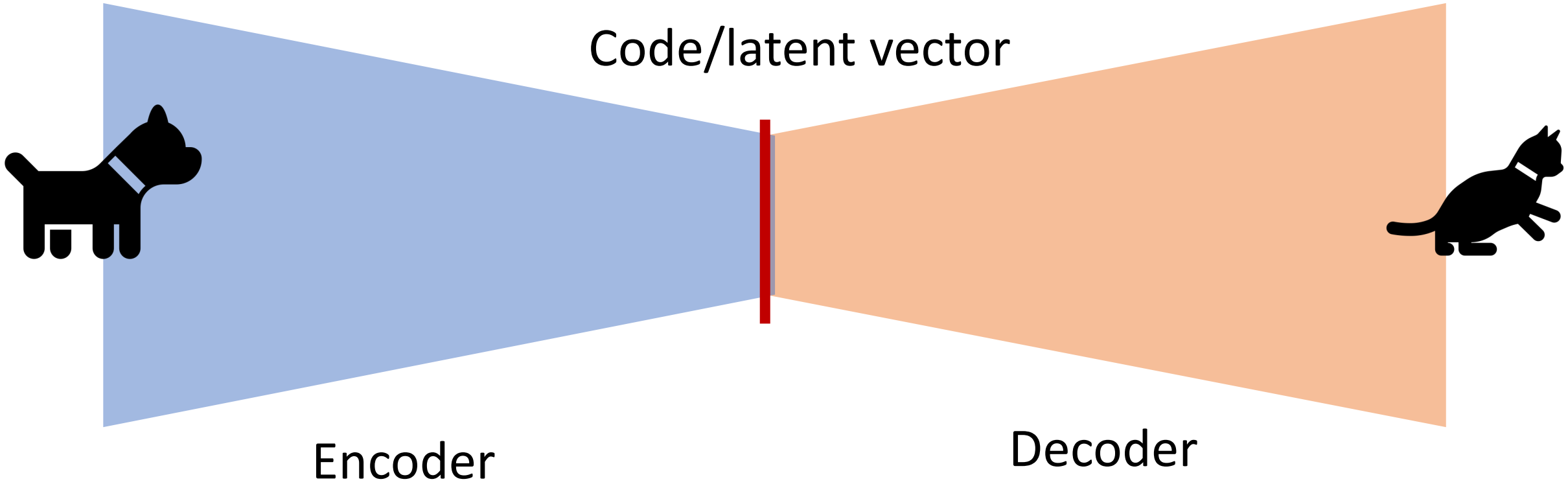
code



Decoder

Decoder

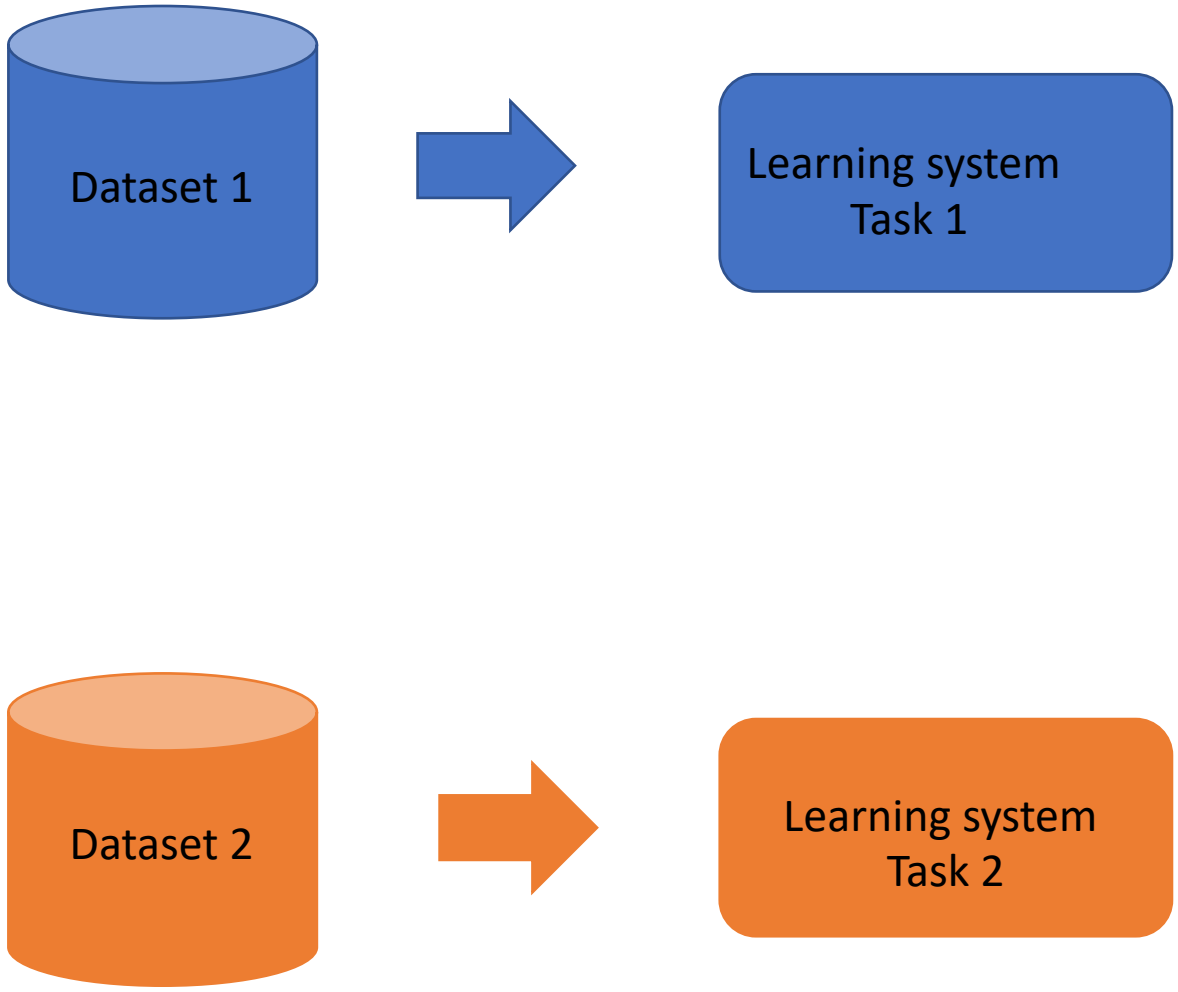
Autoencoder



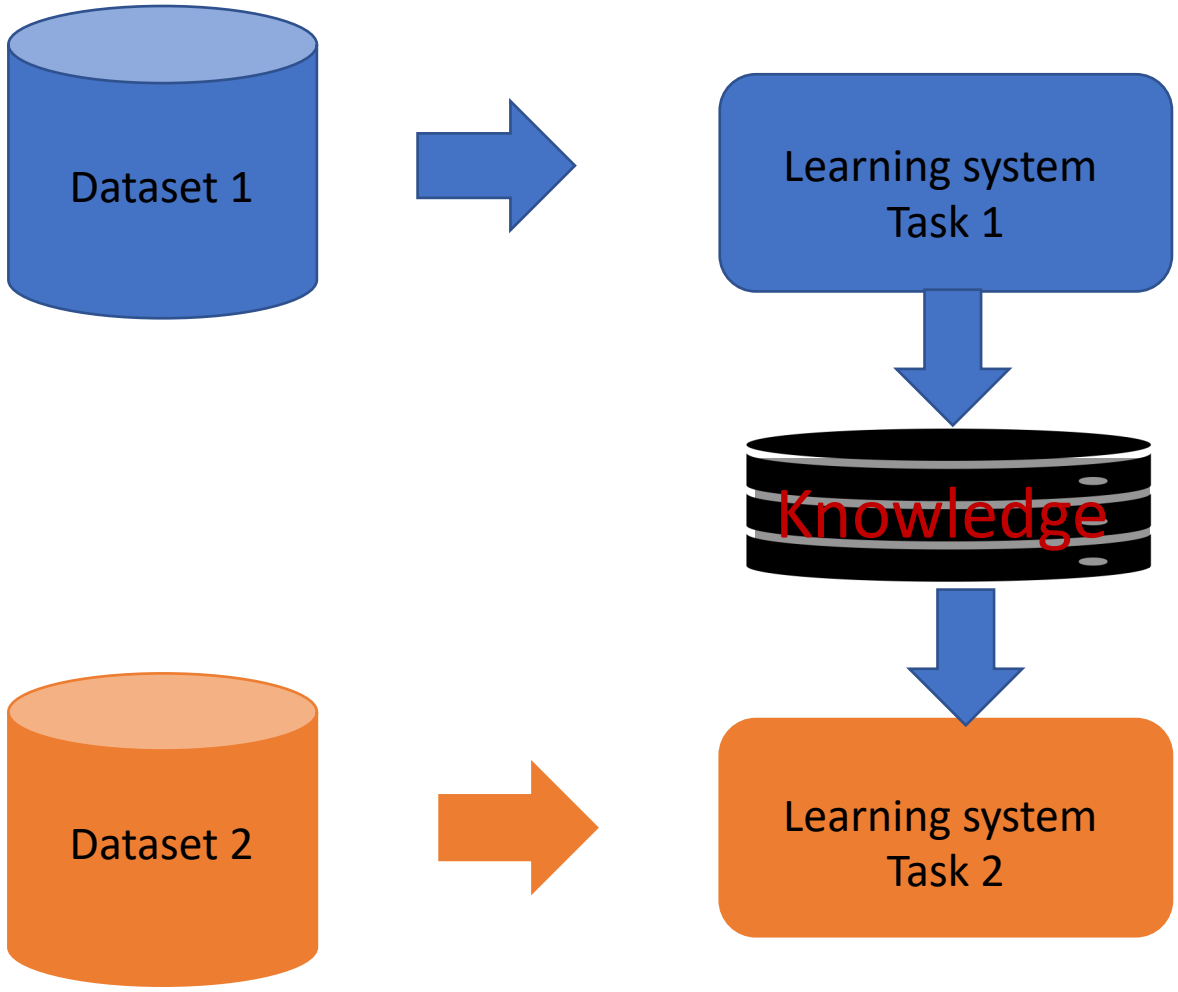
```
filter_1 = 64
filter_2 = 32
filter_3 = 16
kernel_size = (4, 4)
pool_size = (2, 2)
latent_dim = 16

input_layer = Input(shape=(60, 80, 3))

encoder_conv1 = Conv2D(
    filter_1,
    kernel_size,
    activation='sigmoid',
    padding='same')(input_layer)

encoder_pool1 = MaxPool2D(
    pool_size,
    padding='same')(encoder_conv1)

encoder_conv2 = Conv2D(
    filter_2,
    kernel_size,
    activation='sigmoid',
    padding='same')(encoder_pool1)

code_output = MaxPool2D(
    pool_size,
    padding='same')(encoder_conv2)

encoder = Model(input_layer, code_ouput, name='encoder')
```

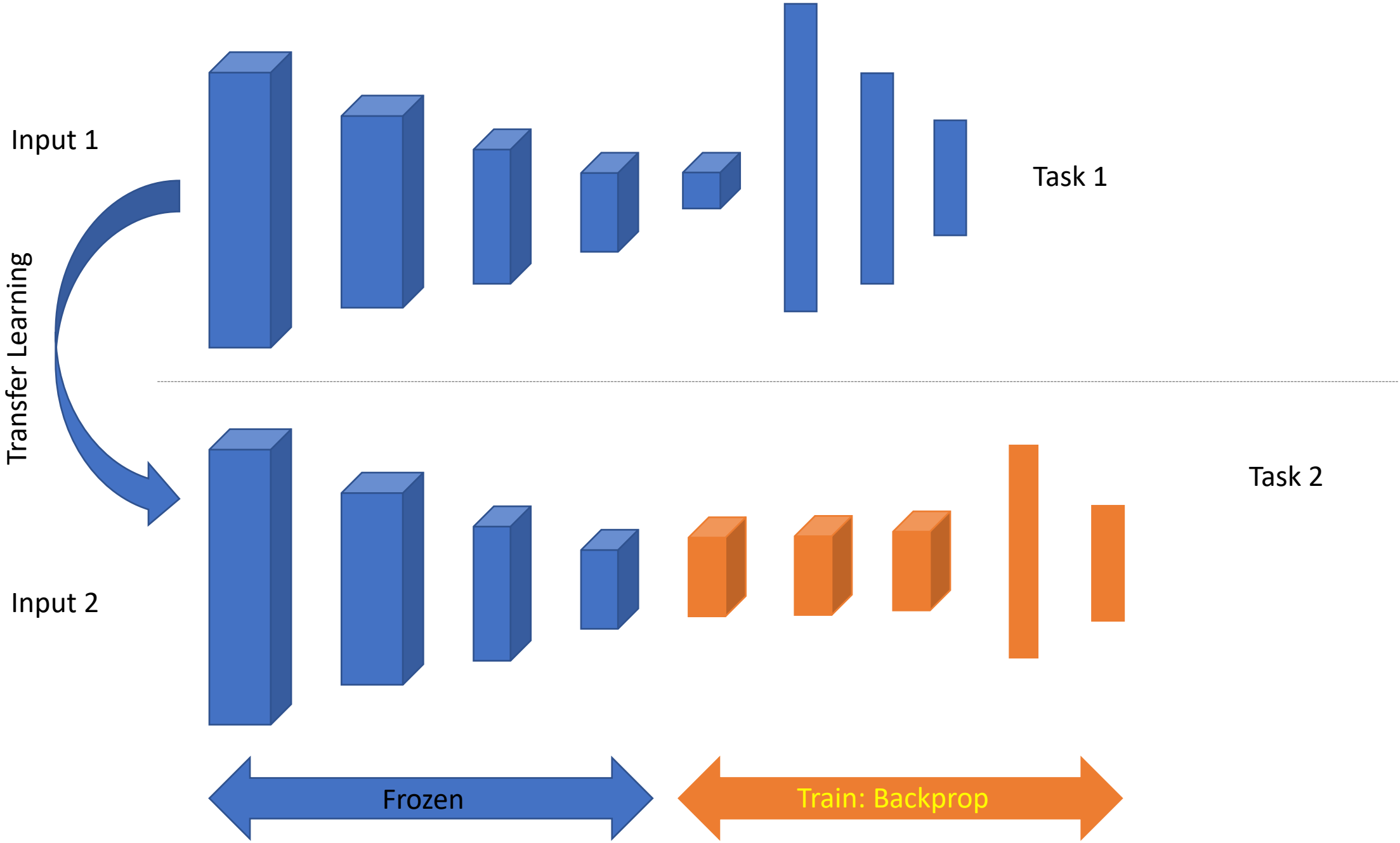
```
code_inputs = Input(shape=(15, 20, 32), name="decoder_input")

decoder_conv1 = Conv2D(
    filter_2,
    kernel_size,
    activation='sigmoid',
    padding='same')(code_inputs)

decoder_up1 = UpSampling2D(
    pool_size)(decoder_conv1)

decoder_conv2 = Conv2D(
    filter_1,
    (1,1),
    activation='sigmoid',
    name="conv3")(decoder_up1)

decoder_up2 = UpSampling2D(
    pool_size,
    name="up3")(decoder_conv2)

output_layer = Conv2D(
    3,
    (kernel_size),
    padding='same',
    activation='sigmoid')(decoder_up2)

decoder = Model(code_inputs, output_layer, name='decoder')
```

Transfer Learning



Transfer Learning




Input 1
Transfer Learning
Task 1
Input 2
Task 2
Frozen
Train: Backprop

```
# Load `DenseNet201`.

densenet = tf.keras.applications.DenseNet201(input_shape=IMG_SHAPE,
                                             weights='imagenet',
                                             include_top=False)

# Extend the model.

model = tf.keras.Sequential([
  densenet,
  tf.keras.layers.GlobalAveragePooling2D(),
  tf.keras.layers.Dense(101, activation='softmax'),
])
```