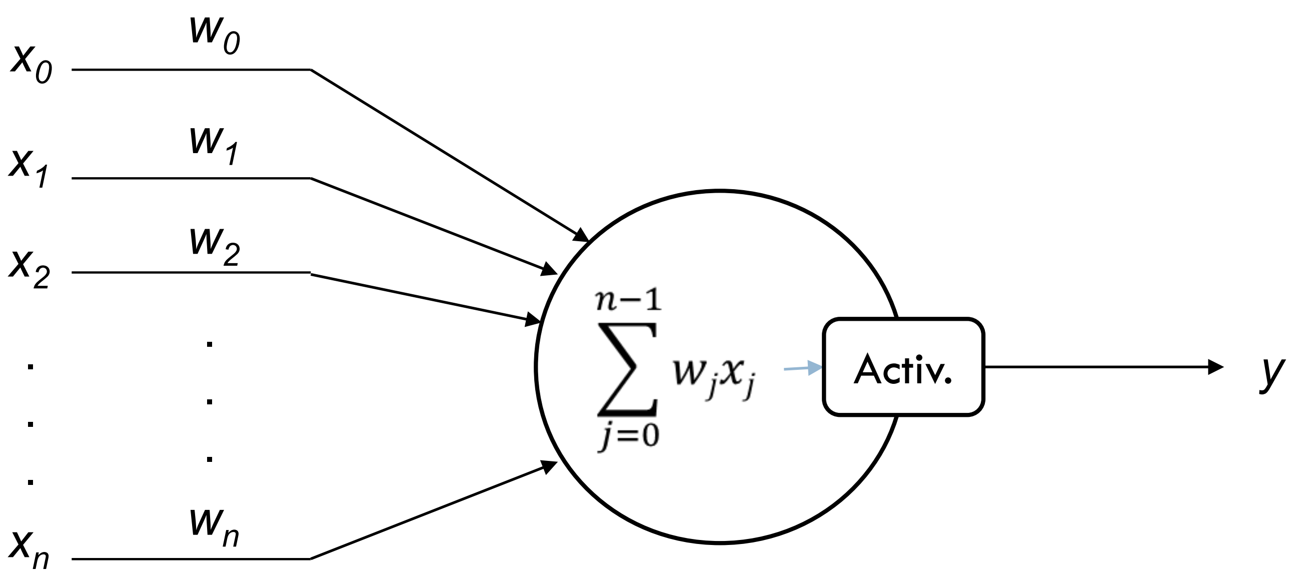
NEURAL NETWORKS

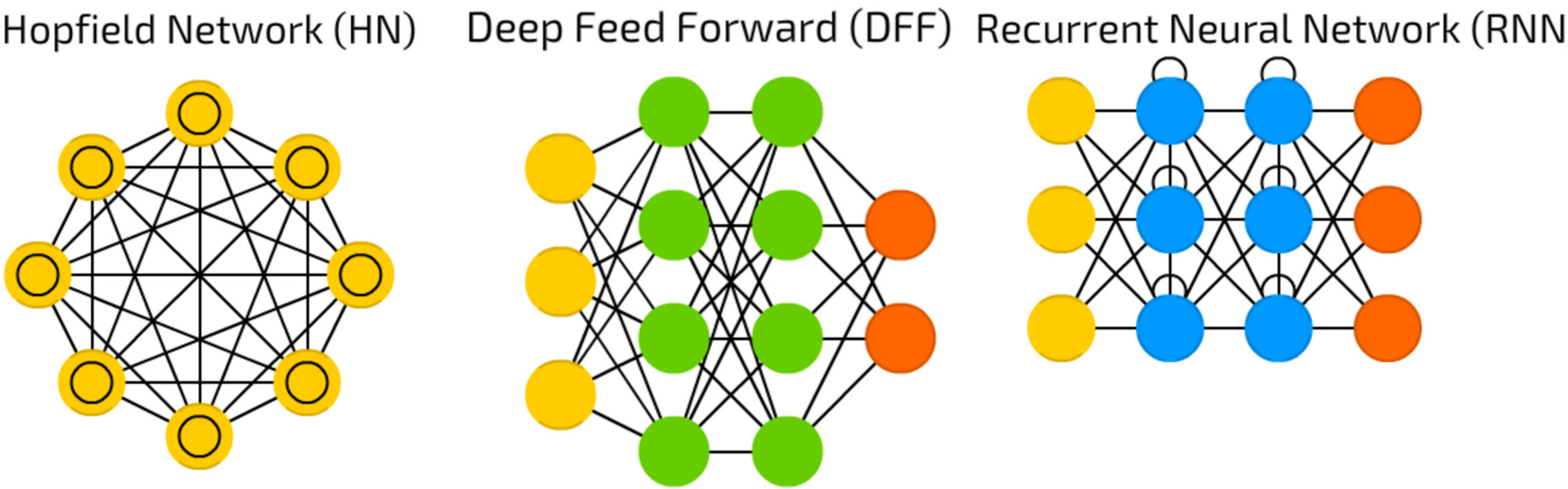
Instructors: Crista Lopes

Artificial Neuron – Perceptron



Neural Networks

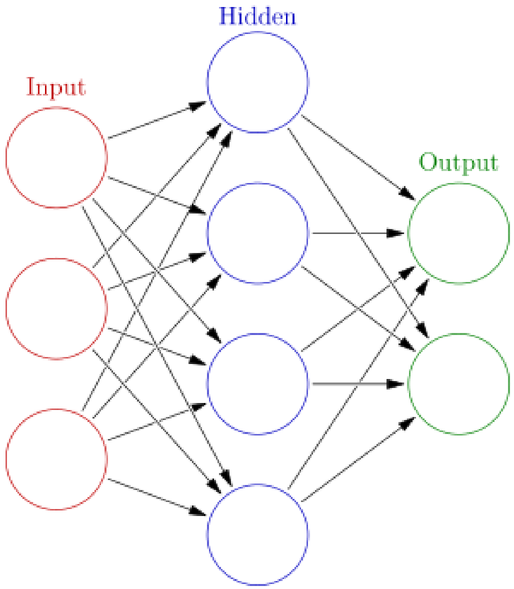
- Networks of perceptrons connected to each other in some topology

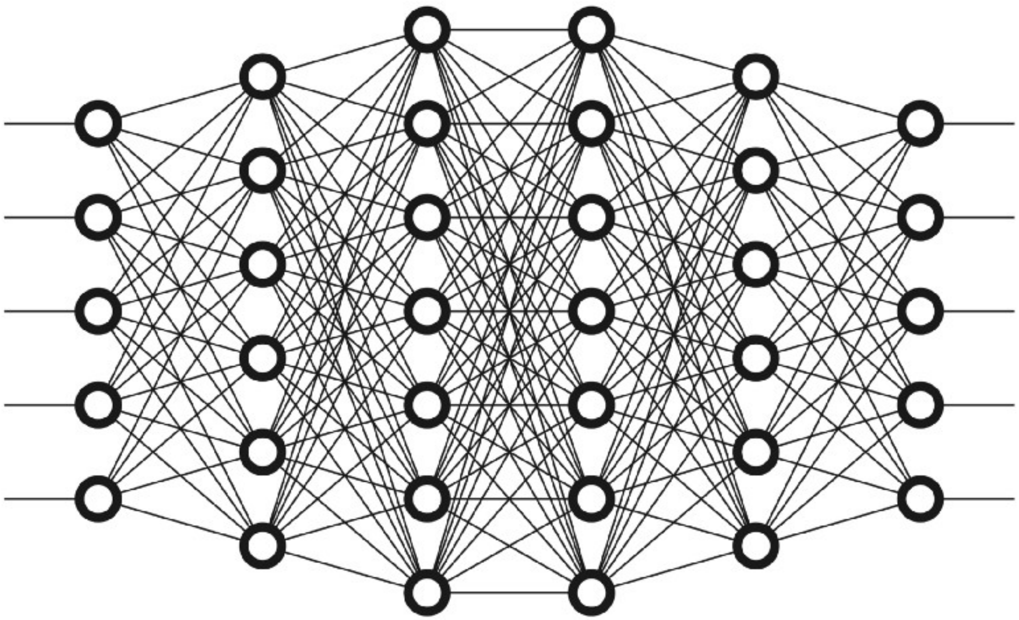


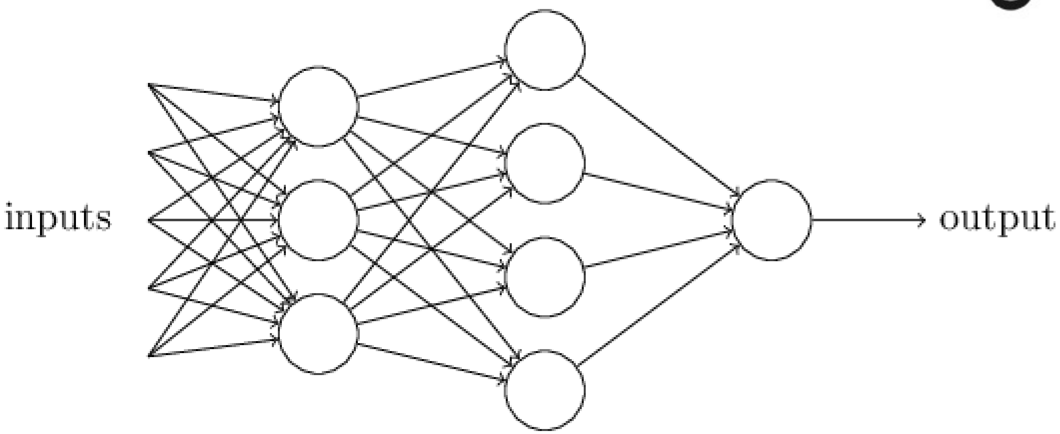
Dataflow Programming

- The network implements a <u>function</u>
  - Input neurons get the input
  - Output neurons show the output
- Neurons receive input from other neurons, and send their outputs to the neurons they are connected to
- Neurons are data transformers
- No memory – again, a <u>pure function</u>

Feed Forward Neural Networks

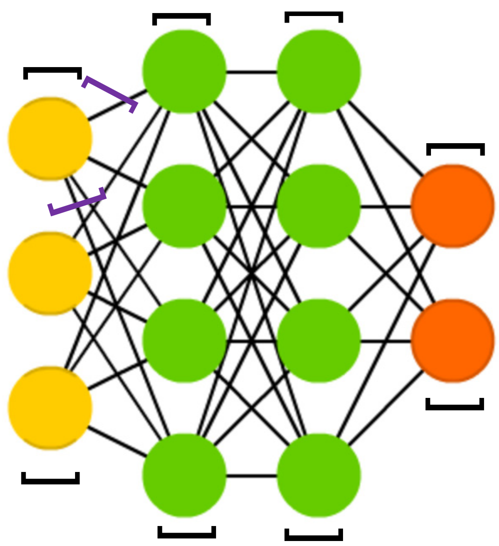






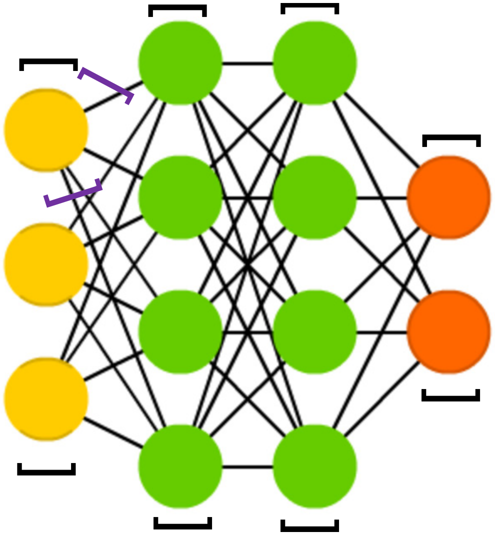
How to model NNs?

- Neurons-as-objects may seem appropriate
  - Would capture network changes dynamically (creation/deletion of neurons)
- Actors, maybe?



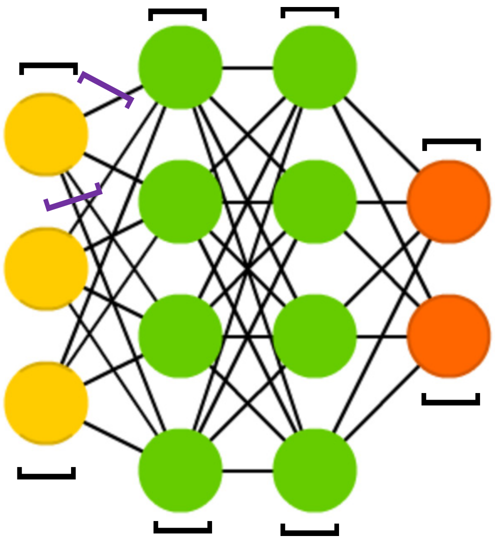
How to model NNs?

- ~~Neurons-as-objects may seem appropriate~~
  - ~~Would capture network changes dynamically (creation/deletion of neurons)~~
- ~~Actors, maybe?~~



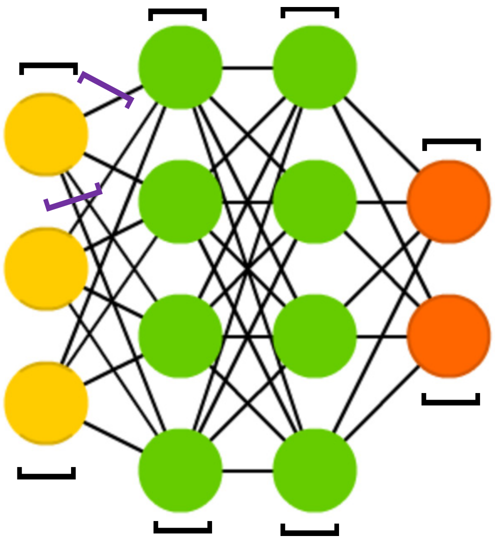
How to model NNs?

- Popular frameworks: no dynamicity, everything is static
- Arrays all the way down!



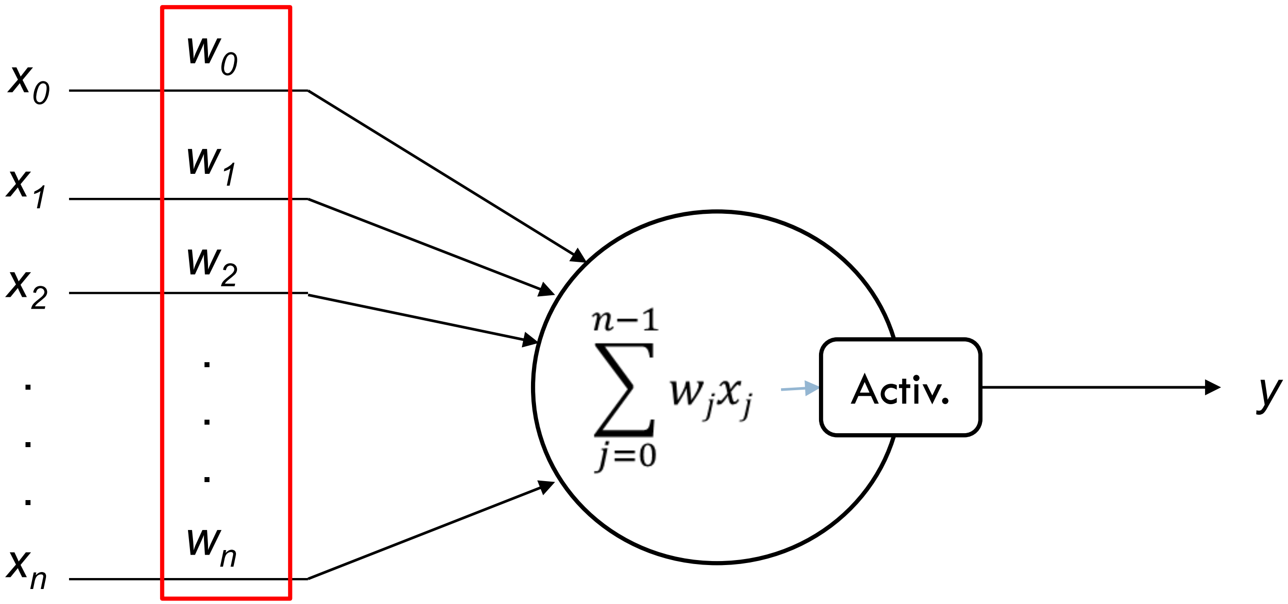
NNs as Array Programming

- Static network allows for powerful array operations
- Highly parallelizable
- Use of GPUs



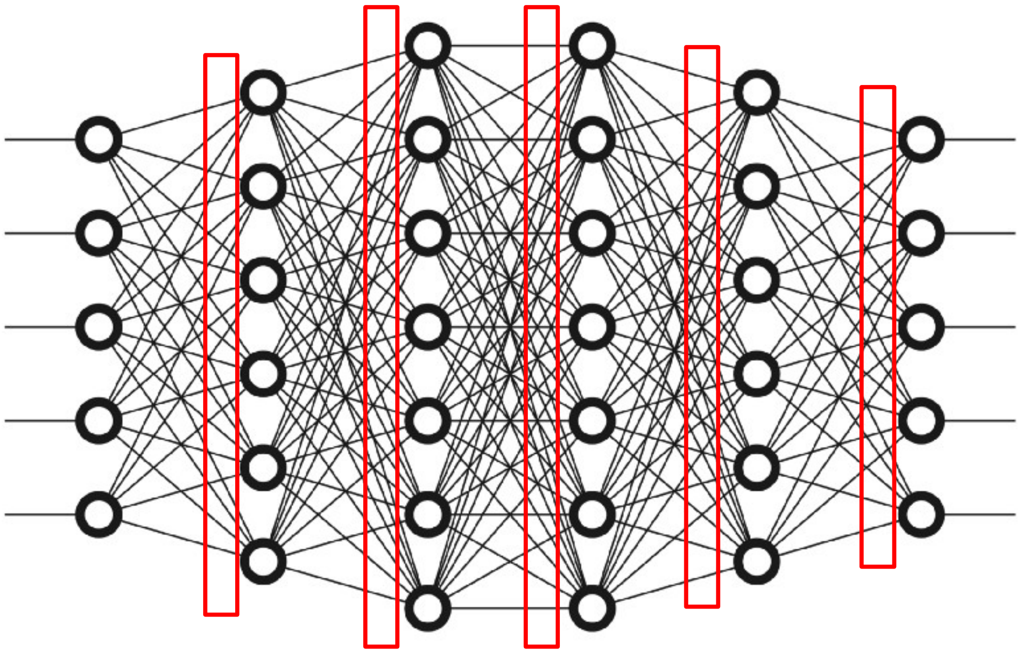
Tensors

- Multidimensional arrays of numbers representing data transformation functions



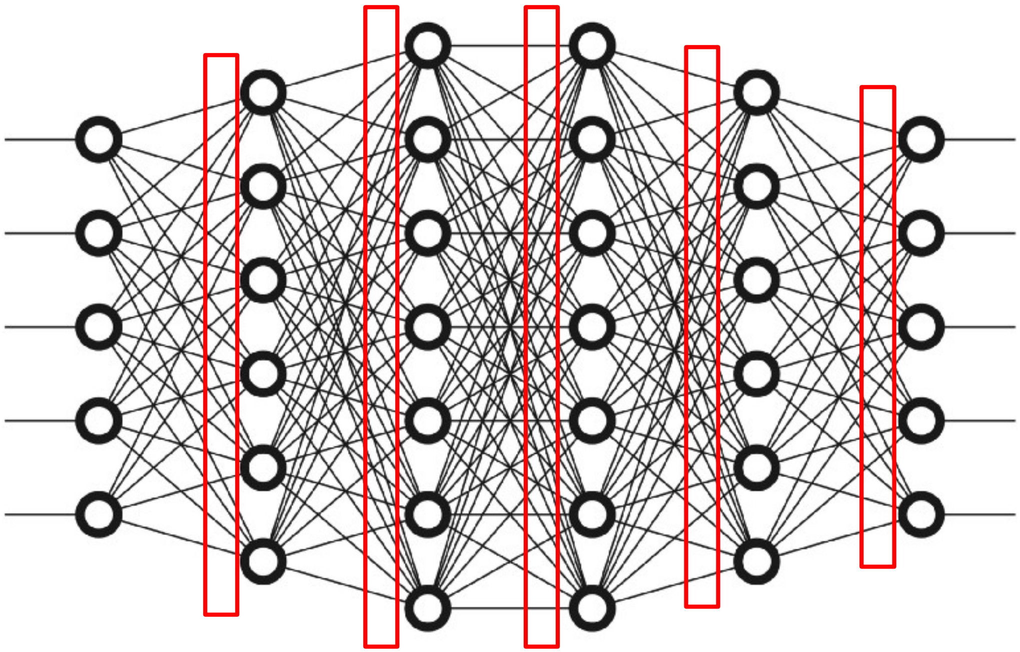
Tensors

- Multidimensional arrays of numbers representing data transformation functions



Learning

- The values of these tensors are **not** set upfront by the programmer; they are **inferred** (learned) during "training" using input/output examples



Anatomy of Neural Network Programs

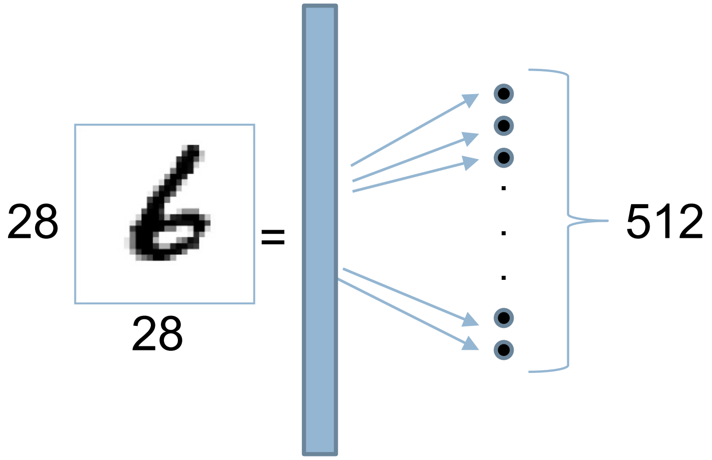
Using Keras

Anatomy of a NN Program

1. Prepare the data
2. Define the network model
3. "Compile" the network model along with certain operational parameters (loss function, optimizer, evaluation metrics)
4. Train the model using the training data (aka *fit*)
5. Use the network on new data (aka *predict*)
    1. Evaluate it using given metrics

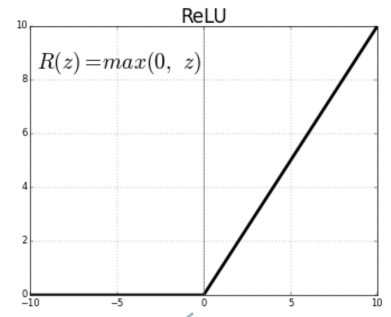
First Keras Example: Recognition of hand-written digits

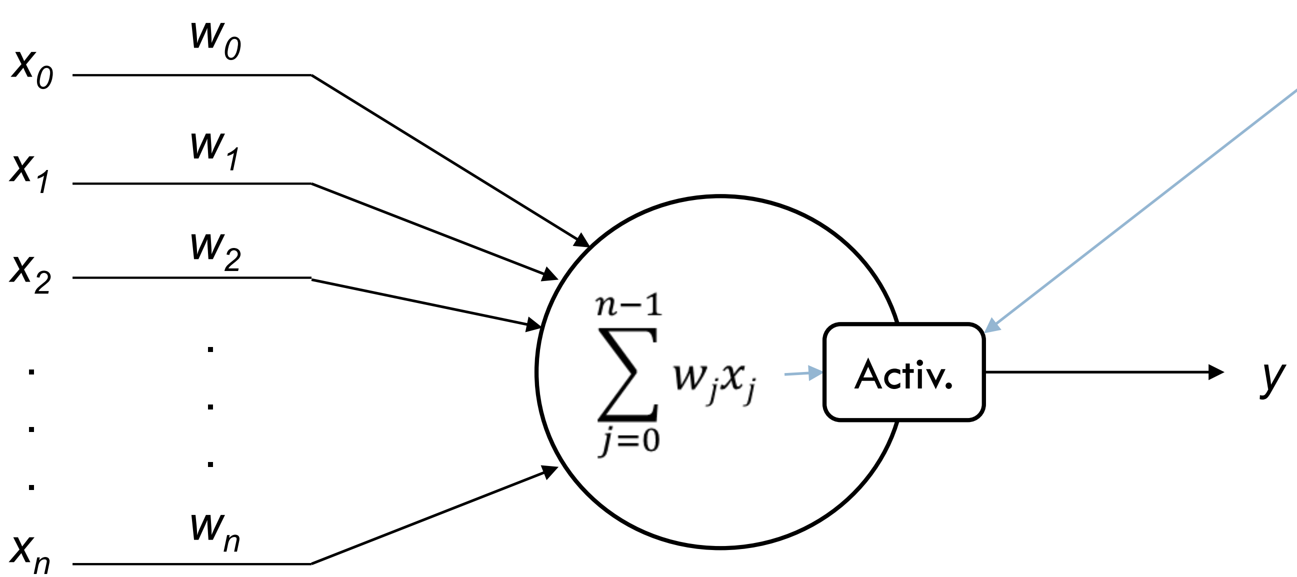
```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu',
                         input_shape=(28 * 28,)))
```



Keras Example : Recognition of hand-written digits

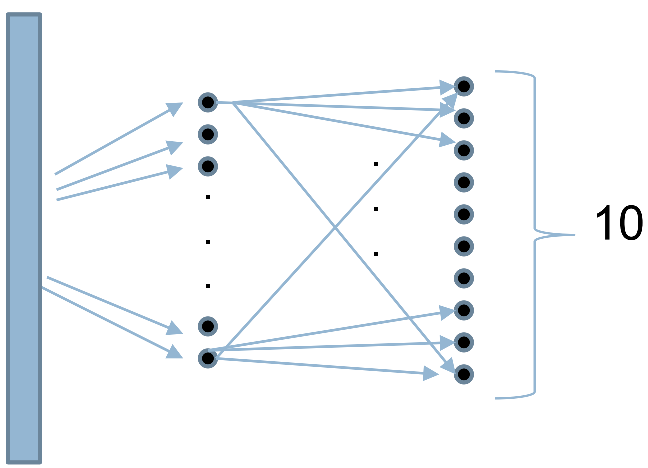
```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu',
                                input_shape=(28 * 28,)))
```





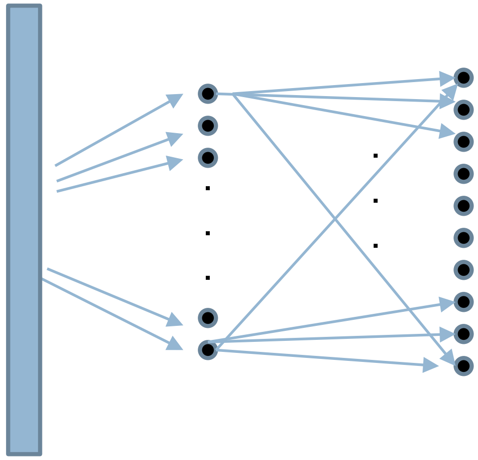
Keras Example : Recognition of hand-written digits

```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu',
                                    input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
```



Keras Example :
Recognition of hand-written digits

```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu',
                                input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
```



Softmax([1, 43, 21]) = [0.000001,
0.999997,
0.000002]

Keras Example :

Recognition of hand-written digits

```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu',
                               input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])

network.fit(train_images, train_labels,epochs=5,batch_size=128)
```

Training phase: inferring the weights (aka parameter values) of the model.
Backpropagation.

Keras Example : Recognition of hand-written digits

```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu',
                         input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])

network.fit(train_images, train_labels,epochs=5,batch_size=128)

predictions = network.predict(test_images)
```

The actual “program”

Keras Example : without learning?

```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu',
                         input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
```

Set the weights manually ☺

```
predictions = network.predict(test_images)
```

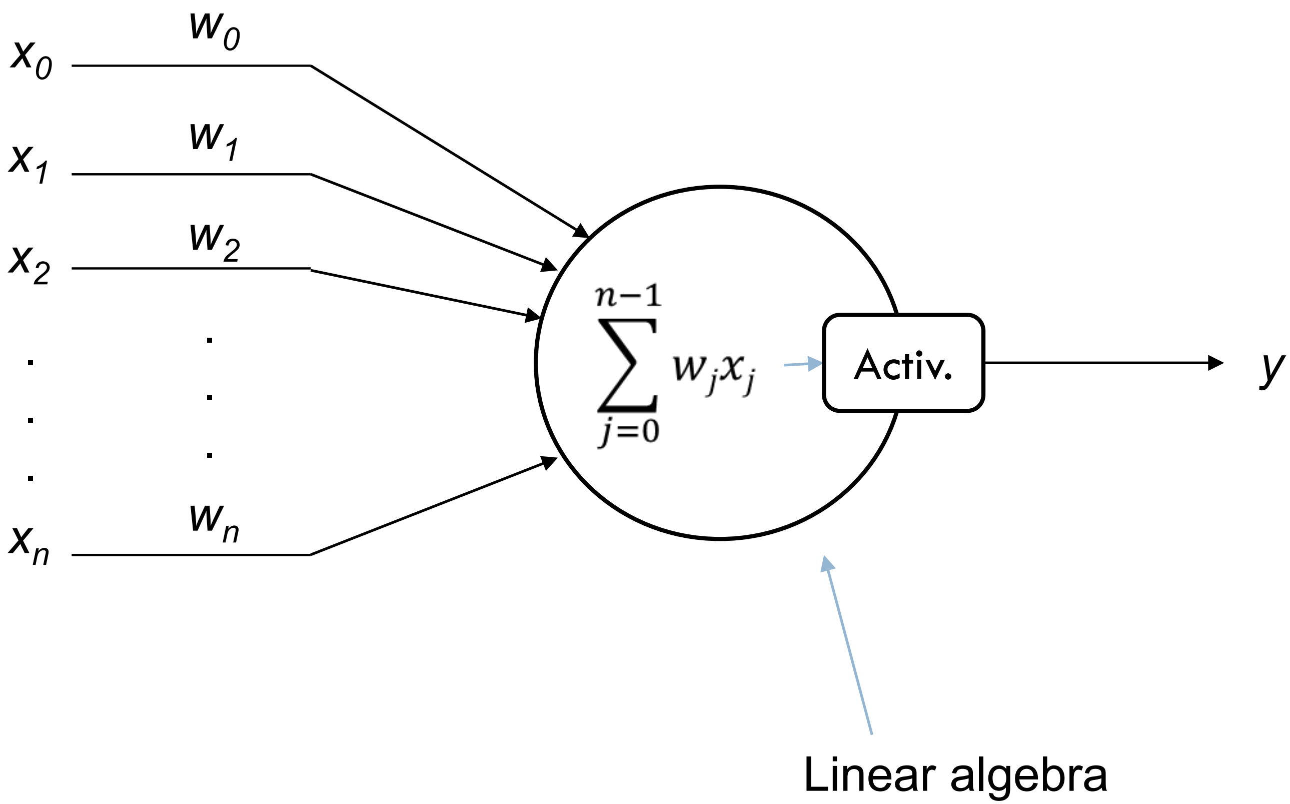
Very high-level array programming style!

Network Models as Monadic Structures

- Each layer is a function
- Pure function: side effects are held off
- Many layers = chain of functions
- Chain is manipulated as an object (compile)
- Predict = run on given input

Data Representations

Artificial Neuron



Numbers only!

Types of Data

- Numerical: images, videos, scalar variables, etc.
- Categorical: strings, text, symbols, etc.
  - Must be converted to numbers before they are fed into a DNN

Data Representations

- All data is encoded as vectors of numbers
- These encodings are an important part of solving problems in this style
  - Some encodings make the problem easy to solve
  - Others make the problem hard
- Deep neural nets can be seen as a sequence of data representation transformations
- The “learning” part is about searching for suitable representations of the data

One Hot Encodings for categorical data

- Popular and simple, but quite expensive
- Given N different things, use N bits, where only one bit is 1 and all others are 0
  - Excellent orthogonality
- E.g. one-hot encoding of the alphabet, 26 letters:
  - A = 10000000000000000000000000
  - B = 01000000000000000000000000
  - C = 00100000000000000000000000
  - Etc.

One-hot vs. binary

- A = 10000000000000000000000000
- B = 01000000000000000000000000

- A = 01000001
- B = 01000010
- “ = 00100010

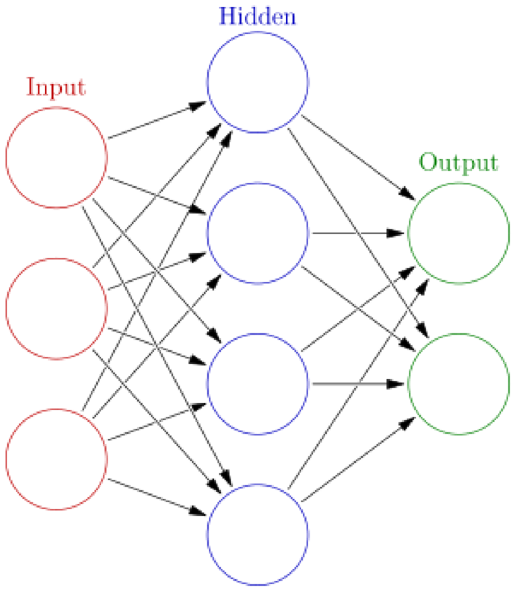
- One hot has less rules than ASCII
- The ASCII rules don’t mean anything in vector space

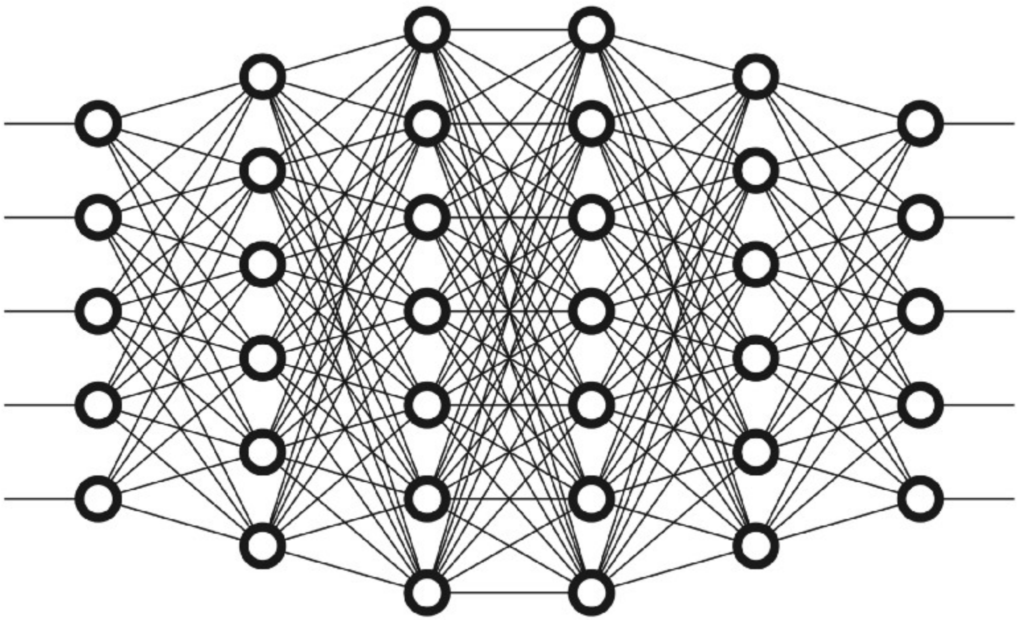
Encodings vs. Embeddings

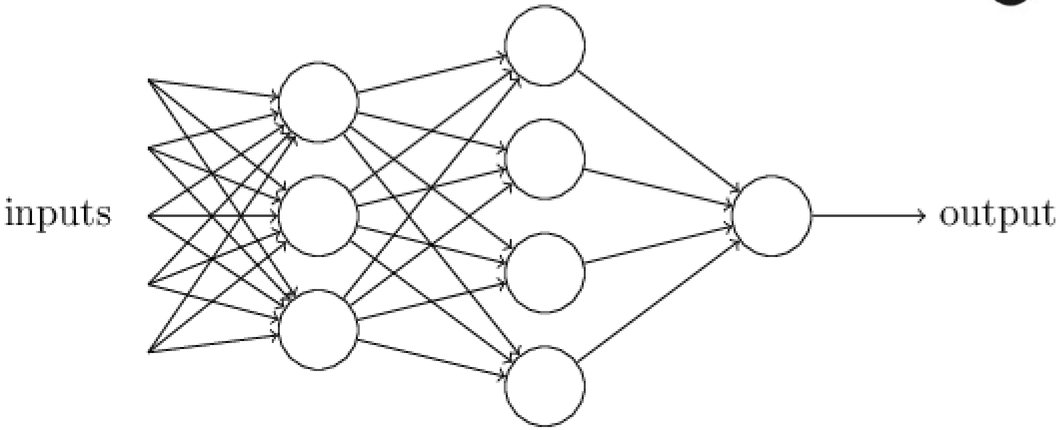
- Both represent data in as numerical vectors
- Embeddings are encodings where the proximity in the space is meaningful
  - E.g. word2vec embeddings place words with similar meaning in close proximity of each other
- For certain problems, it helps using embeddings to start with

Popular Network Architectures

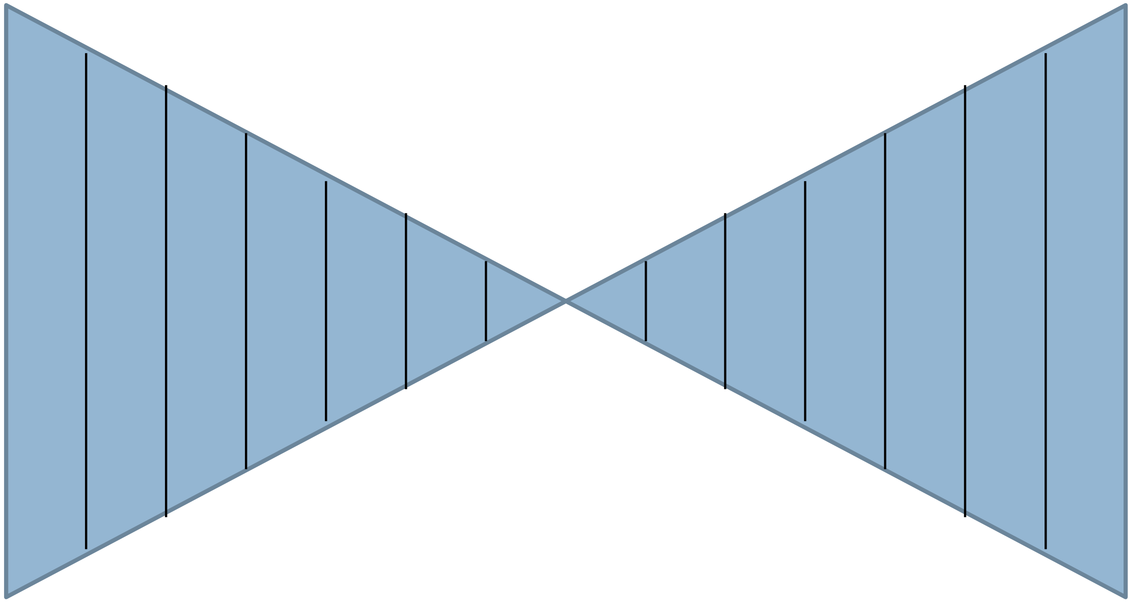
Feed Forward Neural Networks







Encoder-Decoder Architectures

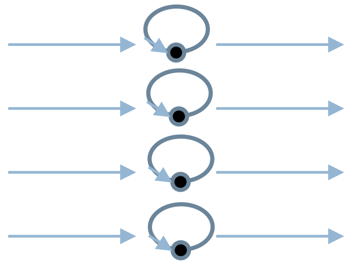


We’re dealing with real numbers!

Recurrent Neural Networks

- Problem: NNs don’t have memory, they are stateless, but many functions depend on prior values
  - E.g. regex \w+
    input: sequence of characters
    output: sequence of words

- Solution: Networks “with loops”

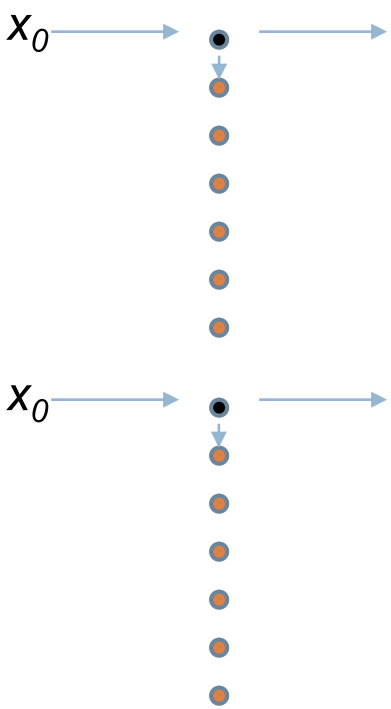


Loops and Arrays

- Problem: loops break the fixed-sized constraint of this style

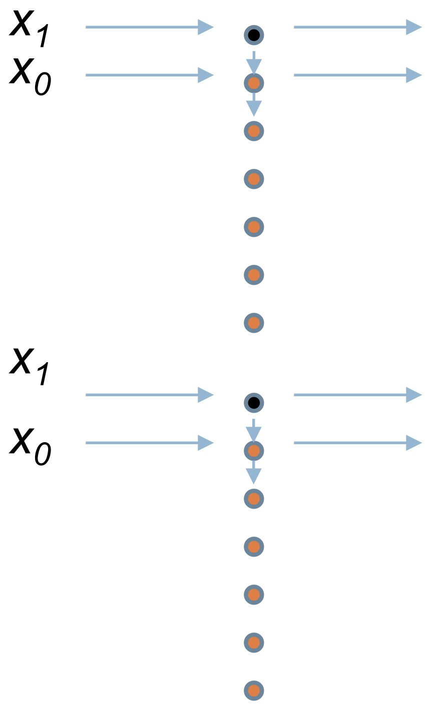
Loops and Arrays

- Problem: loops break the fixed-sized constraint of this style
- Solution: fixed-length loop unrolling



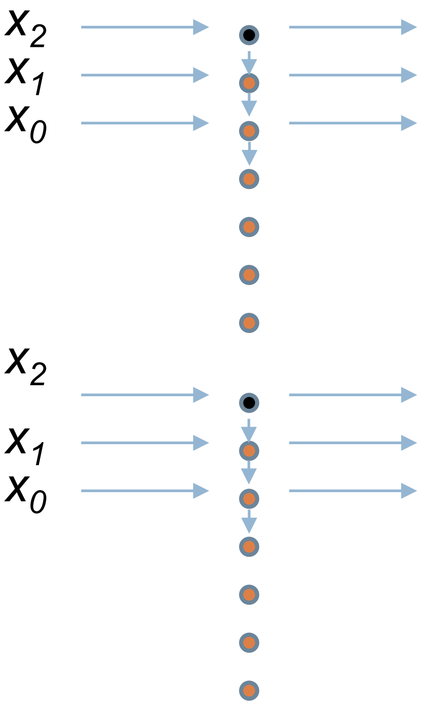
Loops and Arrays

- Problem: loops break the fixed-sized constraint of this style
- Solution: fixed-length loop unrolling



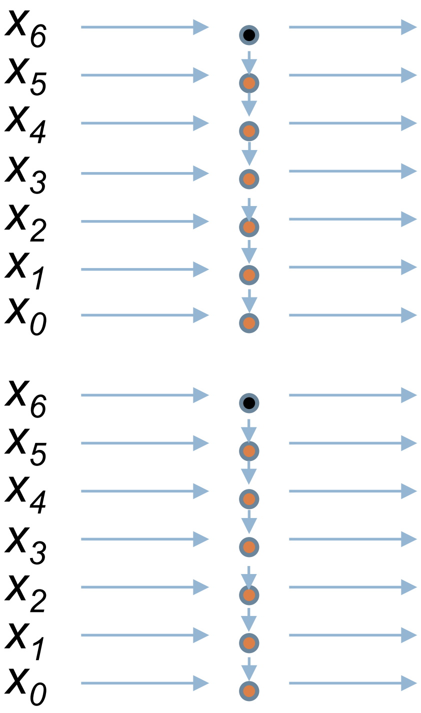
Loops and Arrays

- Problem: loops break the fixed-sized constraint of this style
- Solution: fixed-length loop unrolling



Loops and Arrays

- Problem: loops break the fixed-sized constraint of this style
- Solution: fixed-length loop unrolling

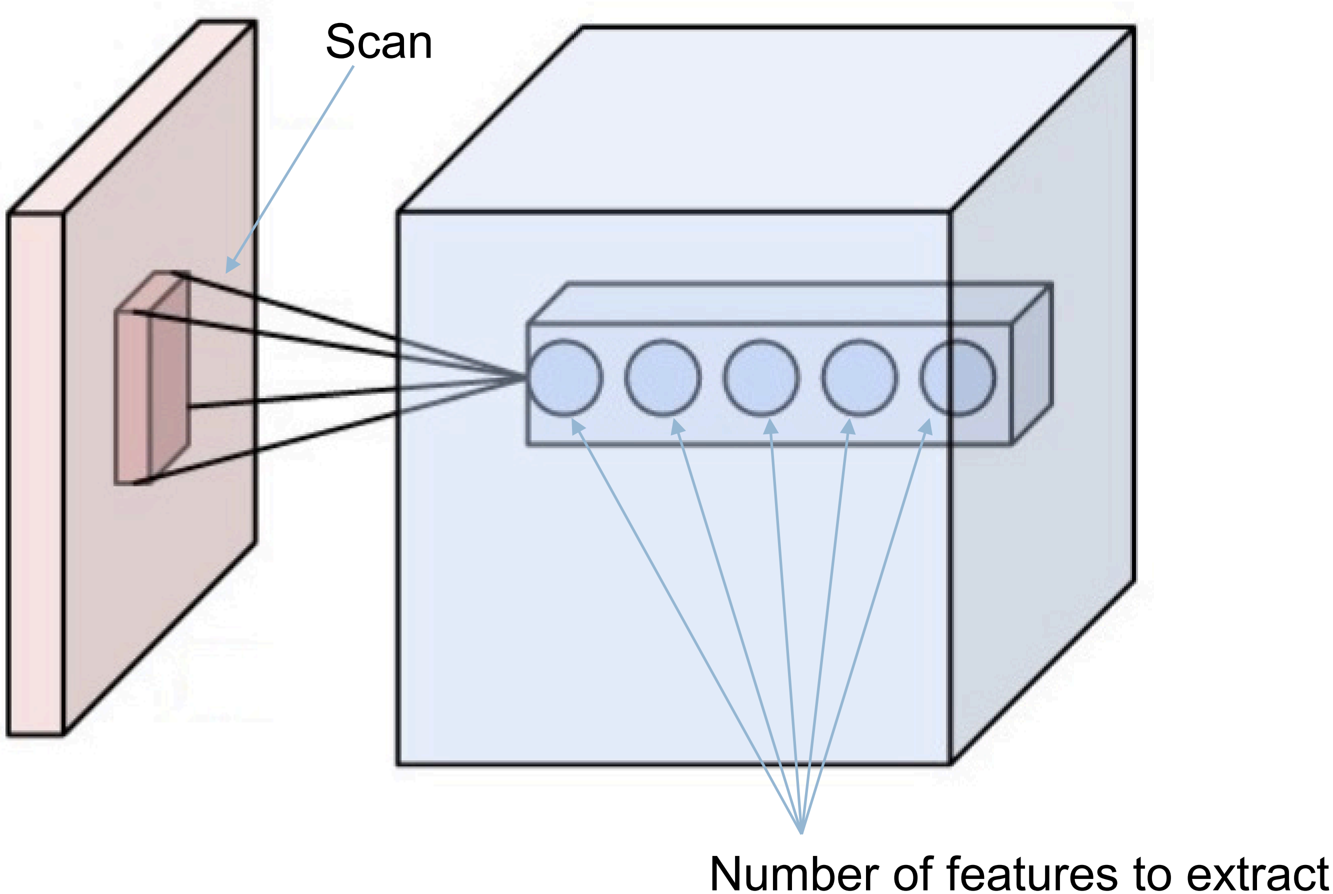


A simple RNN

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
```

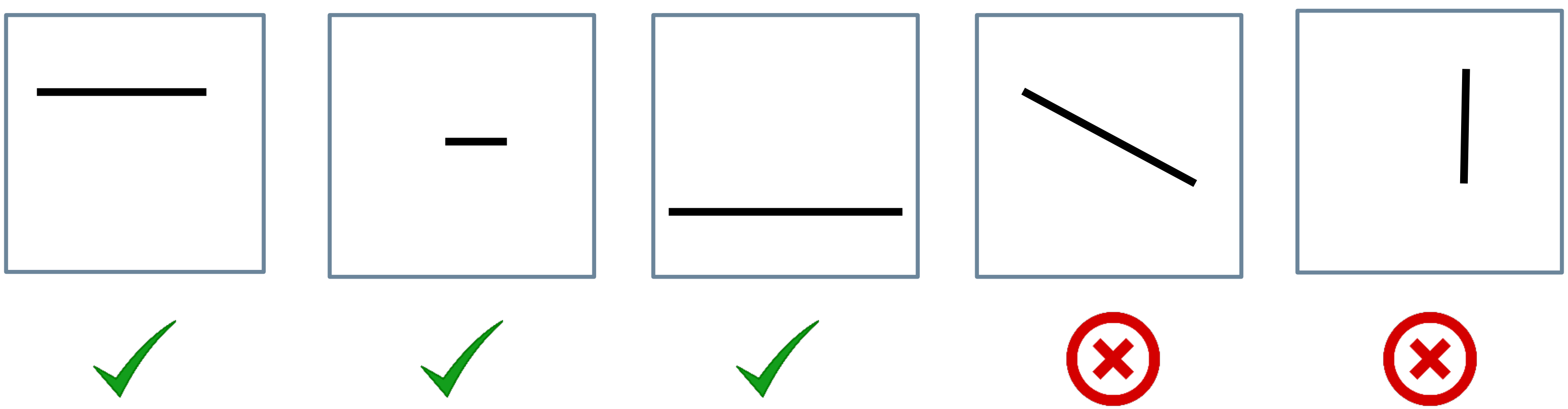
How many “iterations”

Convolutional Neural Networks



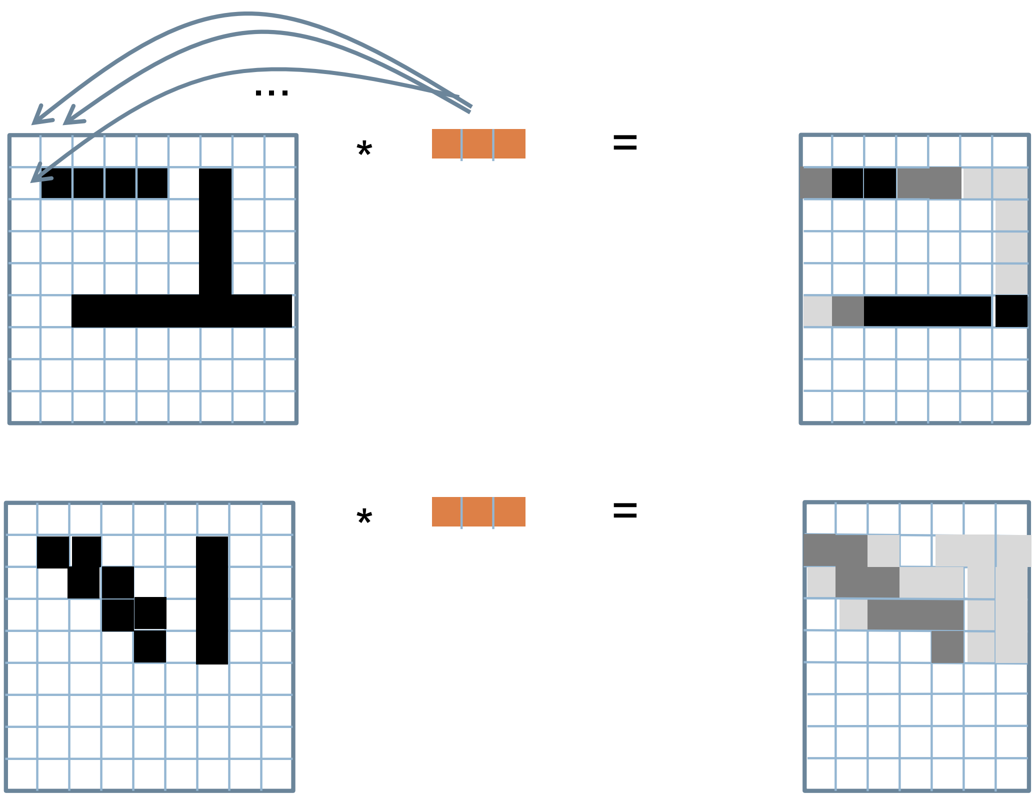
Convolution Basics

- Example: does the image have an horizontal line?
  - Output: Yes (1) or No (0)



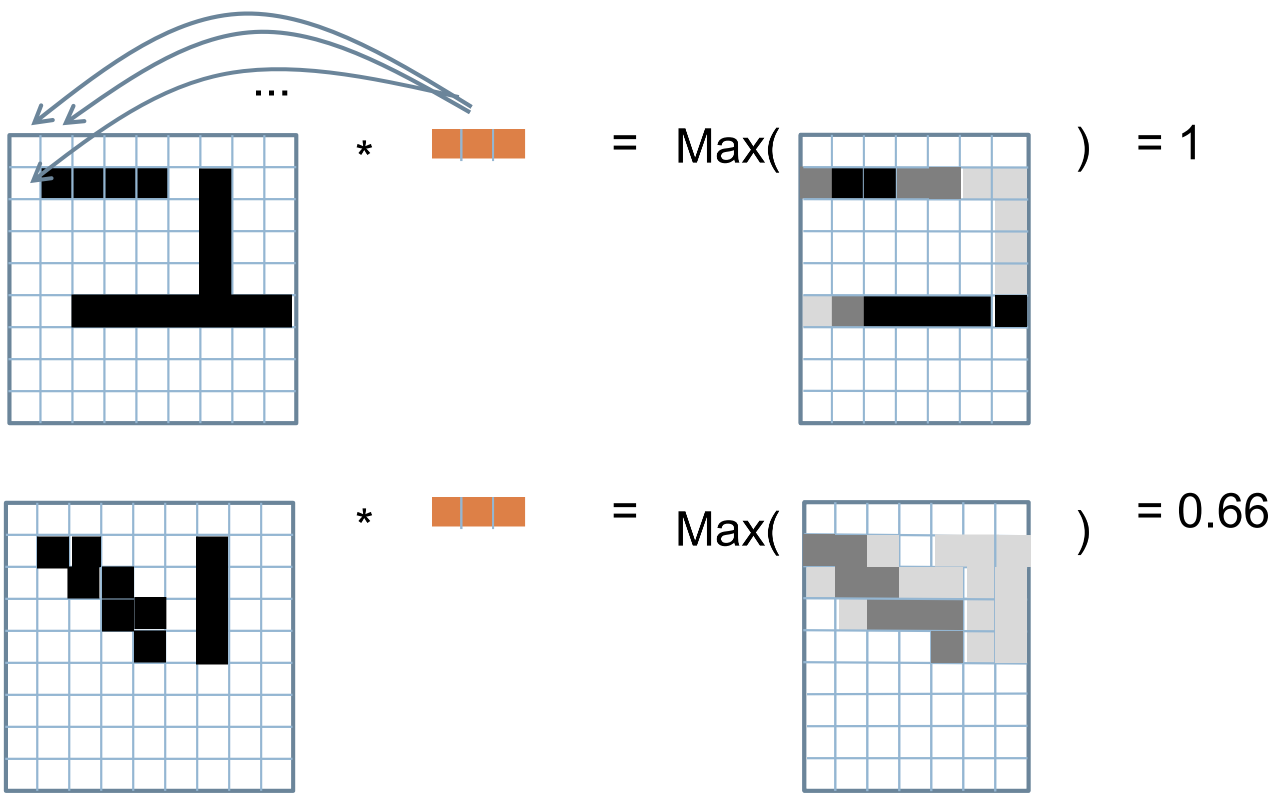
Convolution Basics

- Example: does the image have an horizontal line?
- Scan the image with a filter (aka convolution)



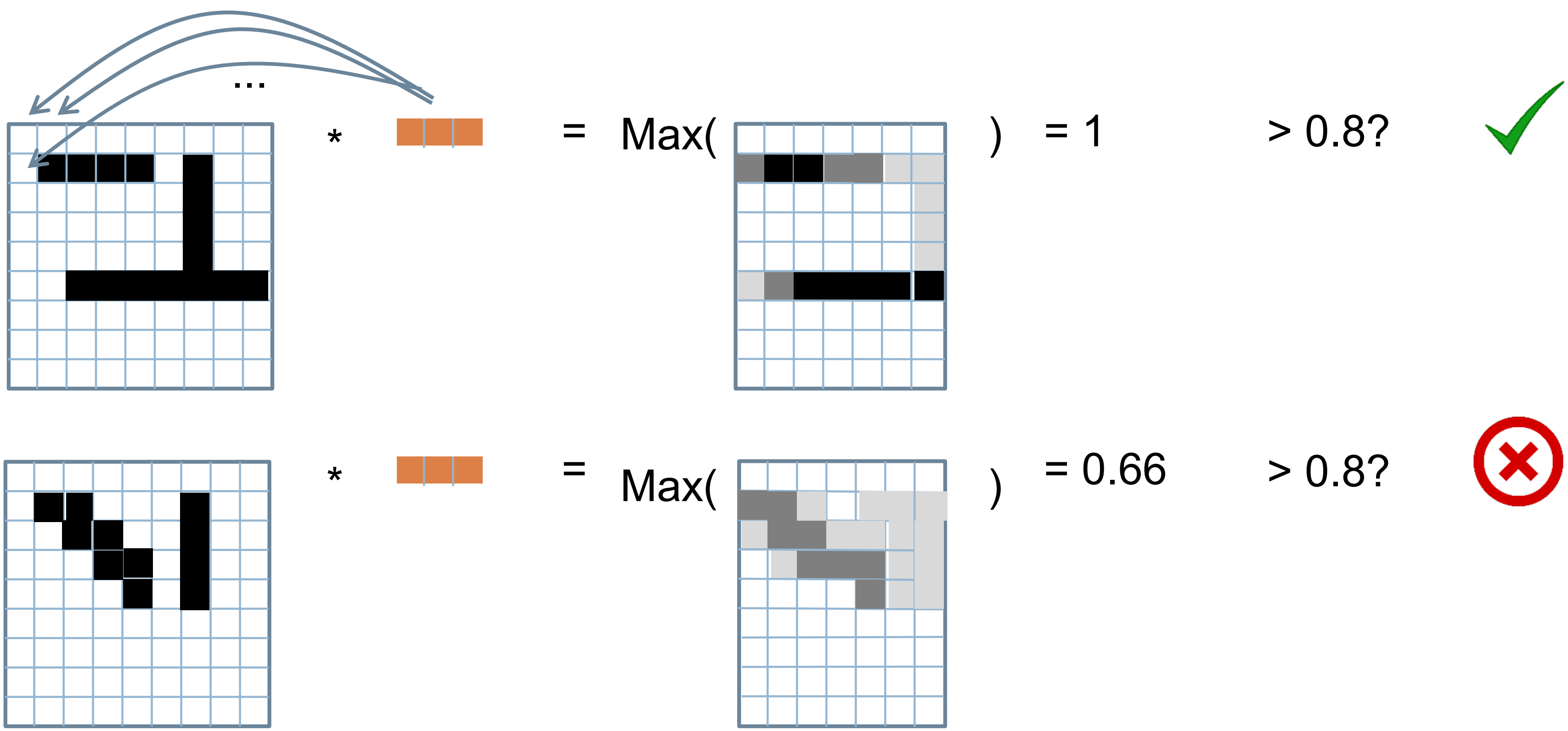
Convolution Basics

- Example: does the image have an horizontal line?
- Then pool



Convolution Basics

- Example: does the image have an horizontal line?
- Then threshold



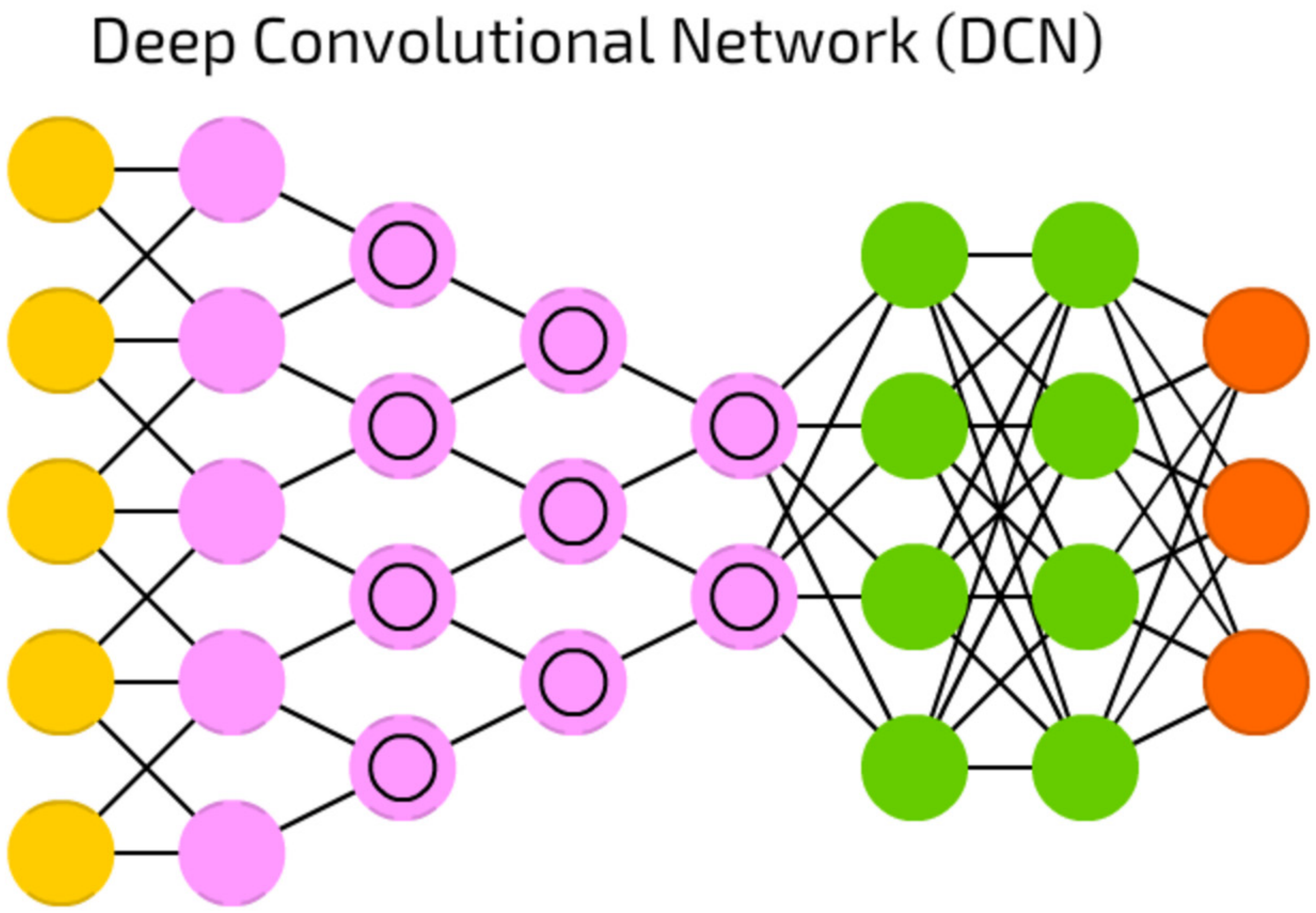
Convolution Neural Networks

- The number and shape of filters is specified by programmer (aka hyperparameters)
- The exact values of the weights are learned

Typical CNN

```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

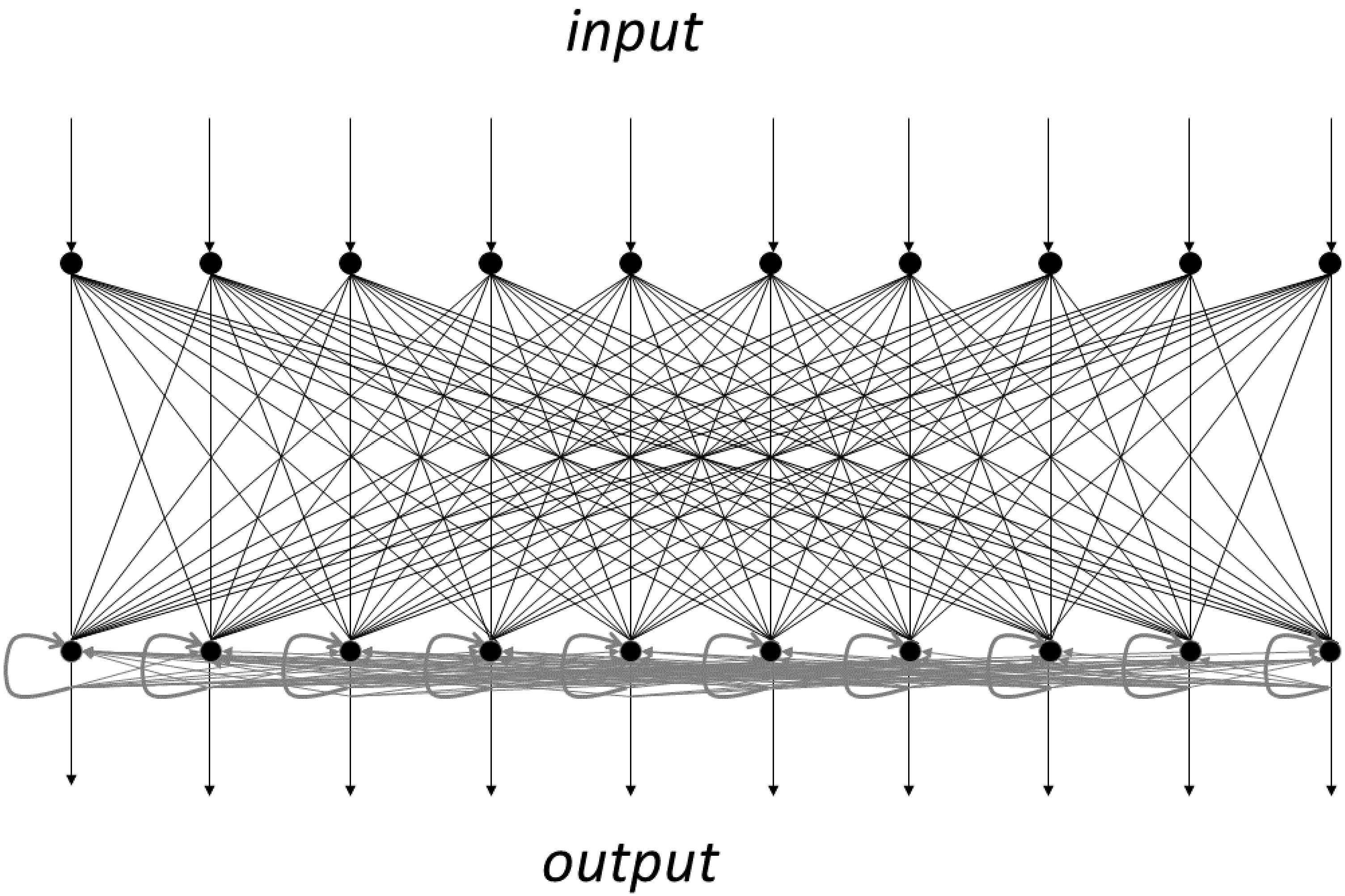
Typical CNN



Recurrent Neural Networks

- Problem: NNs don’t have memory, they are stateless, but many functions depend on prior values
  - E.g. regex \w+
    input: sequence of characters
    output: sequence of words

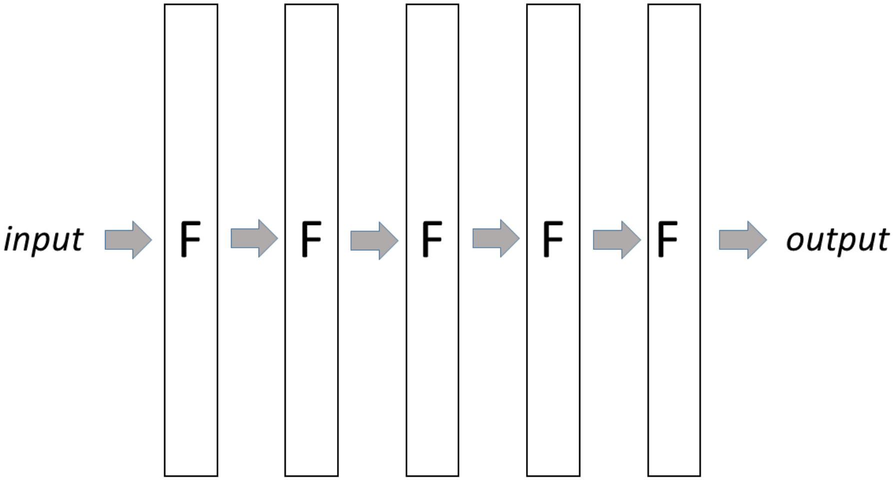
RNNs



Loop unrolling

```
for (i=0; i<5; i++) {
  doSomething();
```

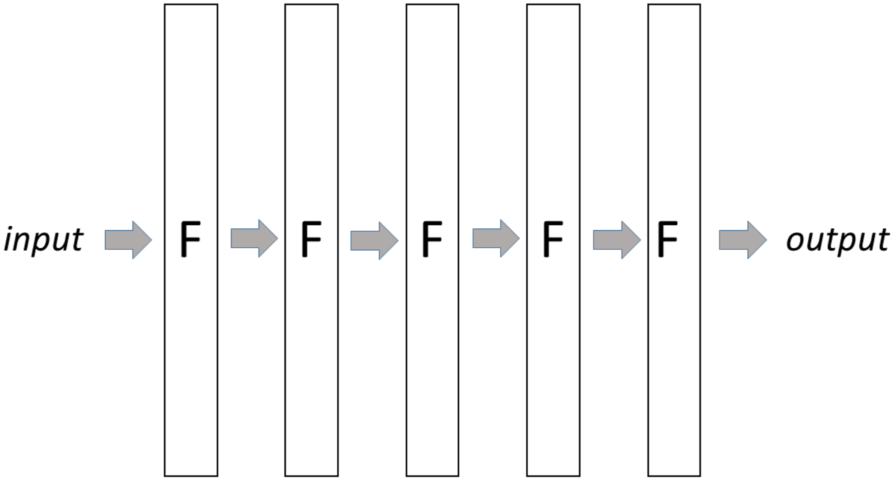
```
doSomething();
doSomething();
doSomething();
doSomething();
doSomething();
```



Loop unrolling

```
for (i=0; i<5; i++) {
  doSomething(i);
```

```
doSomething(0);
doSomething(1);
doSomething(2);
doSomething(3);
doSomething(4);
```



Loop unrolling

```
double d = someCalc();
for (i=0; i<5; i++) {
  d = doSomething(i, d);
```

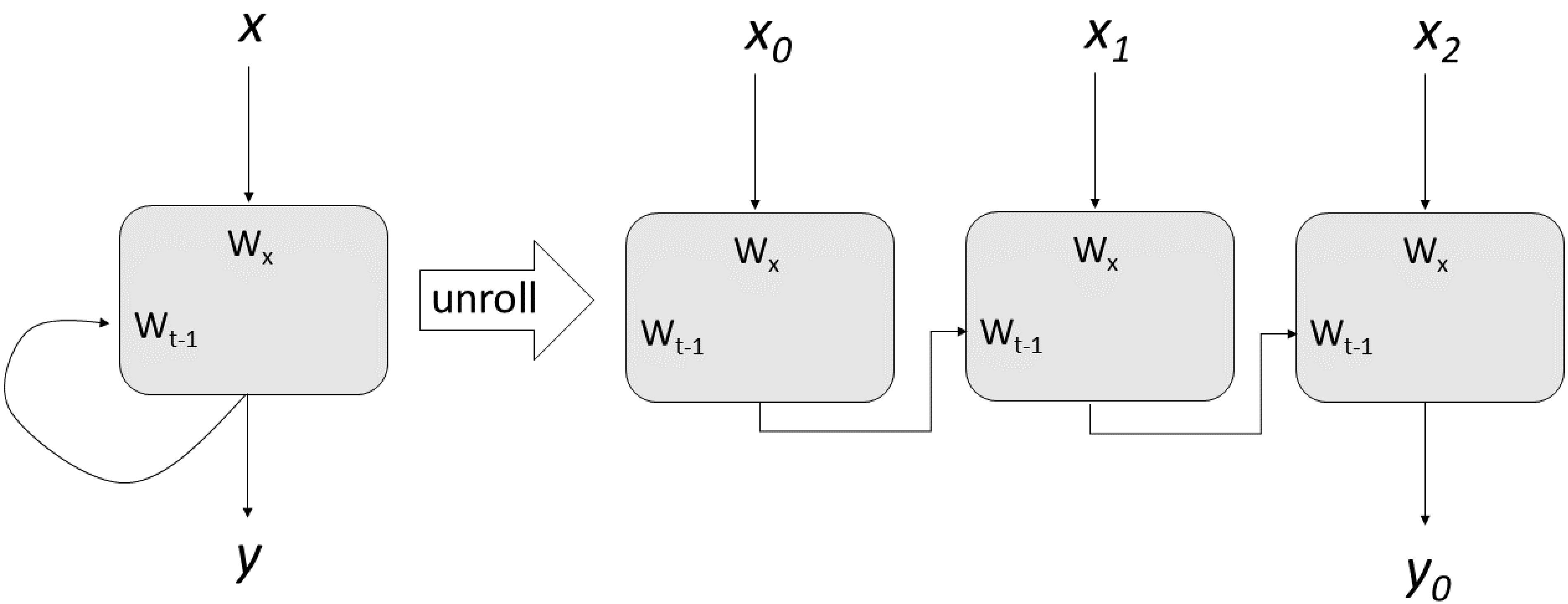
```
double d = someCalc();
d = doSomething(0, d);
d = doSomething(1, d);
d = doSomething(2, d);
d = doSomething(3, d);
d = doSomething(4, d);
```

Loop unrolling

```
double[] d = initArray();
for (i=0; i<5; i++) {
  d[i] = doSomething(i,
                     d[i-1],
                     d[i],
                     d[i+1]);
```

```
double d = someCalc();
d[0] = doSomething(0, -, d[0], d[1]);
d[1] = doSomething(1, d[0], d[1], d[2]);
d[2] = doSomething(2, d[1], d[2], d[3]);
d[3] = doSomething(3, d[2], d[3], d[4]);
d[4] = doSomething(4, d[3], d[4], d[5]);
```

Loop unrolling in NNs



Who defines N, number of iterations?

- The shape of the input!

Neural Network Programming Style

Constraints

- All data is represented as vectors of numbers
- A network is a chain of pure functions
  - Each layer is a function over the vectorized data
- These functions are linear algebra transformations expressed as multi-dimensional vectors (tensors)
- Typically, the network architecture and hyperparameters are explicitly specified
- [with learning] the exact values of the weights are inferred during a training phase with in/out examples

Observations

- Neural Networks are machines for **analog computing** – not {0, 1} but [0, 1]
- Well-known programming concepts can be reinvented in this style
  - Containment verification → convolution
  - Loops → recurrence
  - What else?