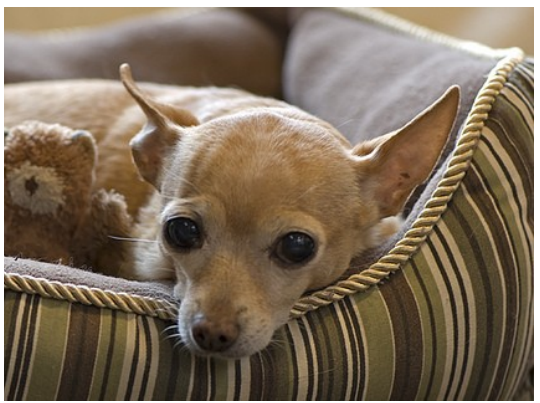
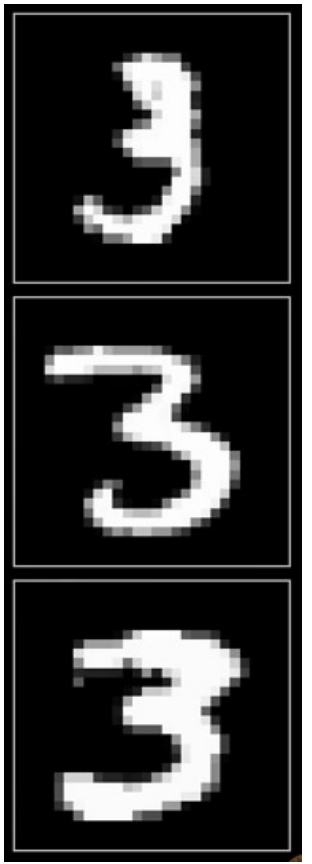
Using deep neural networks as a modelling framework to understand ventral stream computations

Emer Jones

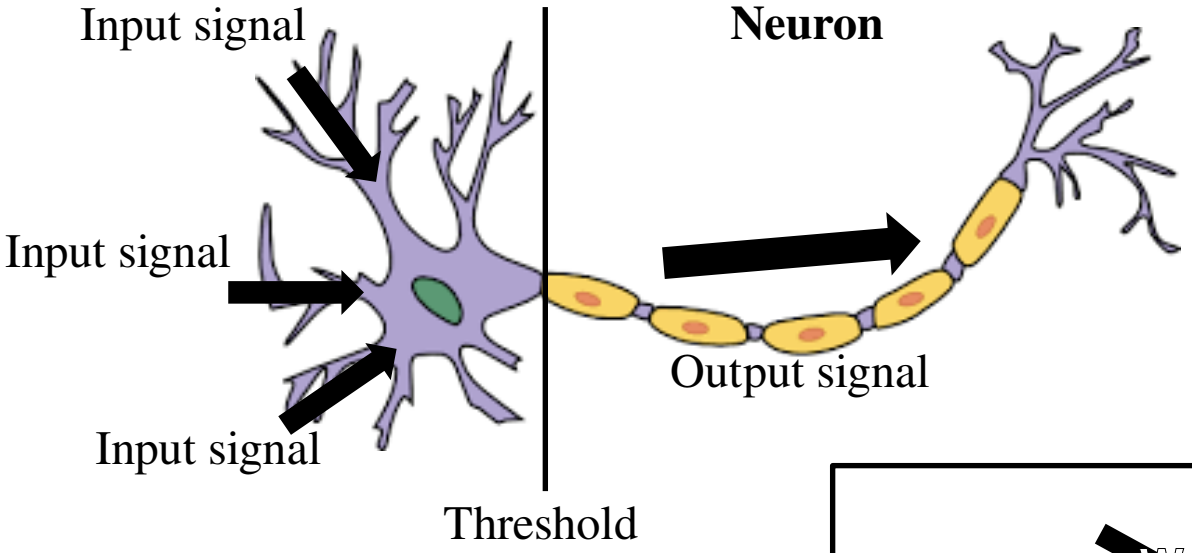
PhD student
MRC Cognition and Brain Sciences Unit,
University of Cambridge

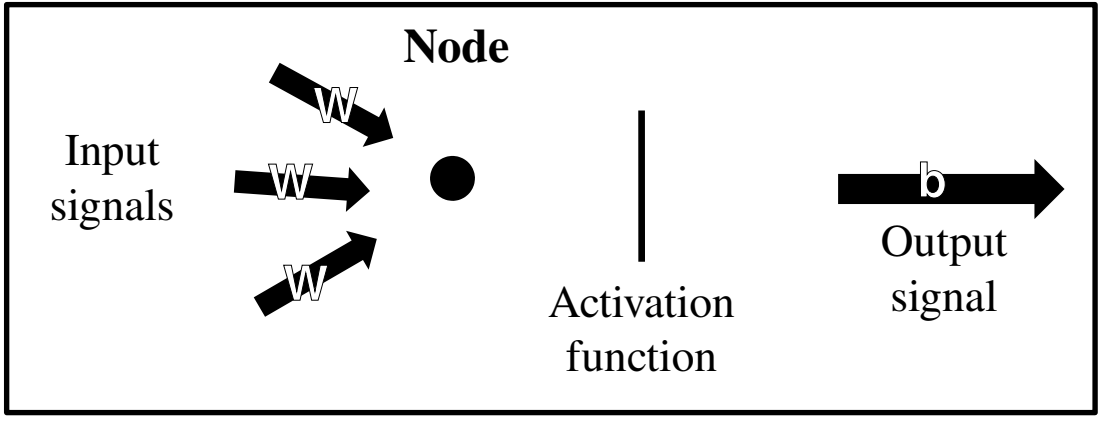


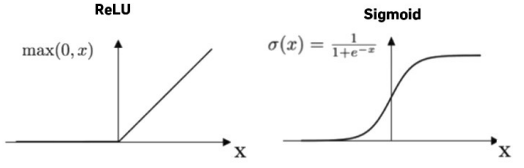




Images from ecoset and 3Blue1Brown

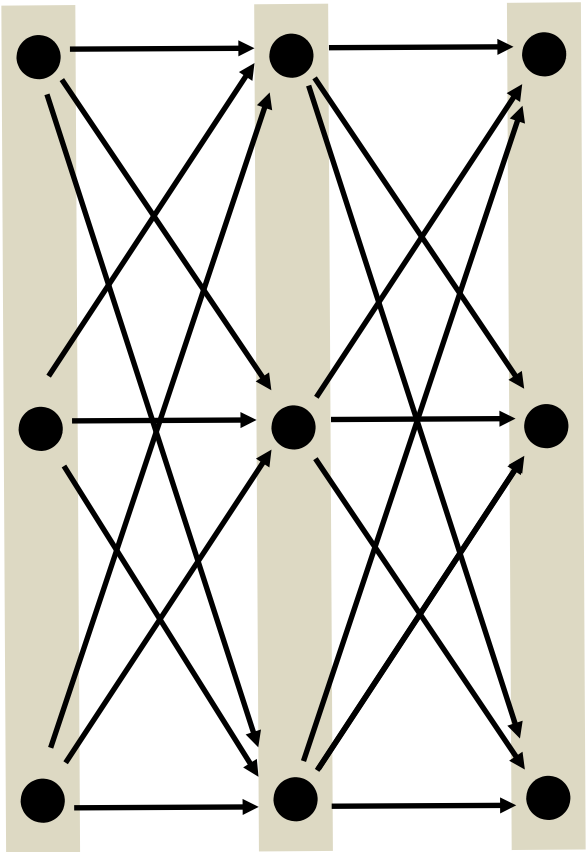




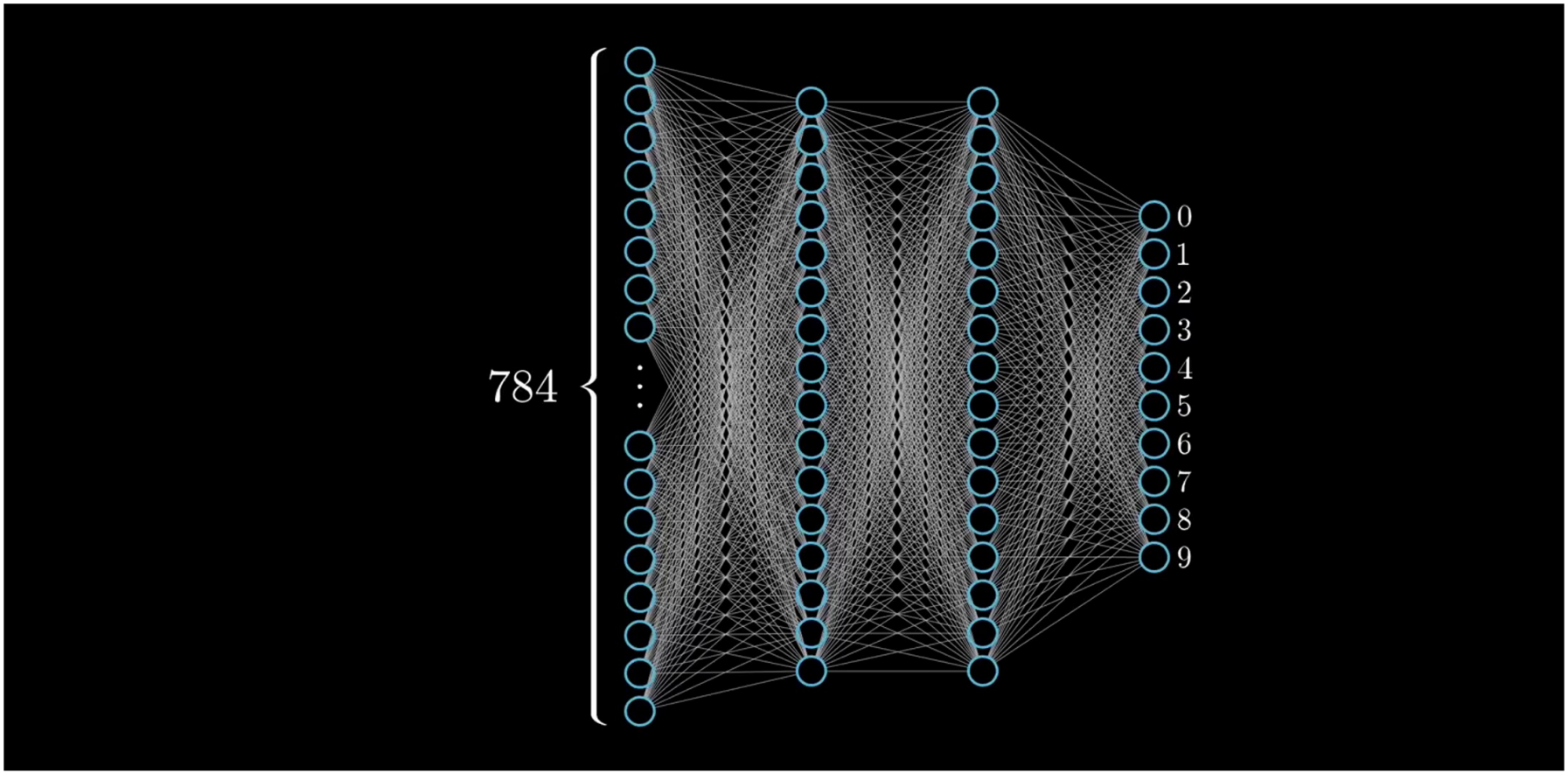


Images: https://docs.paperspace.com/machine-learning/wiki/activation-function
https://en.wikipedia.org/wiki/Neuron

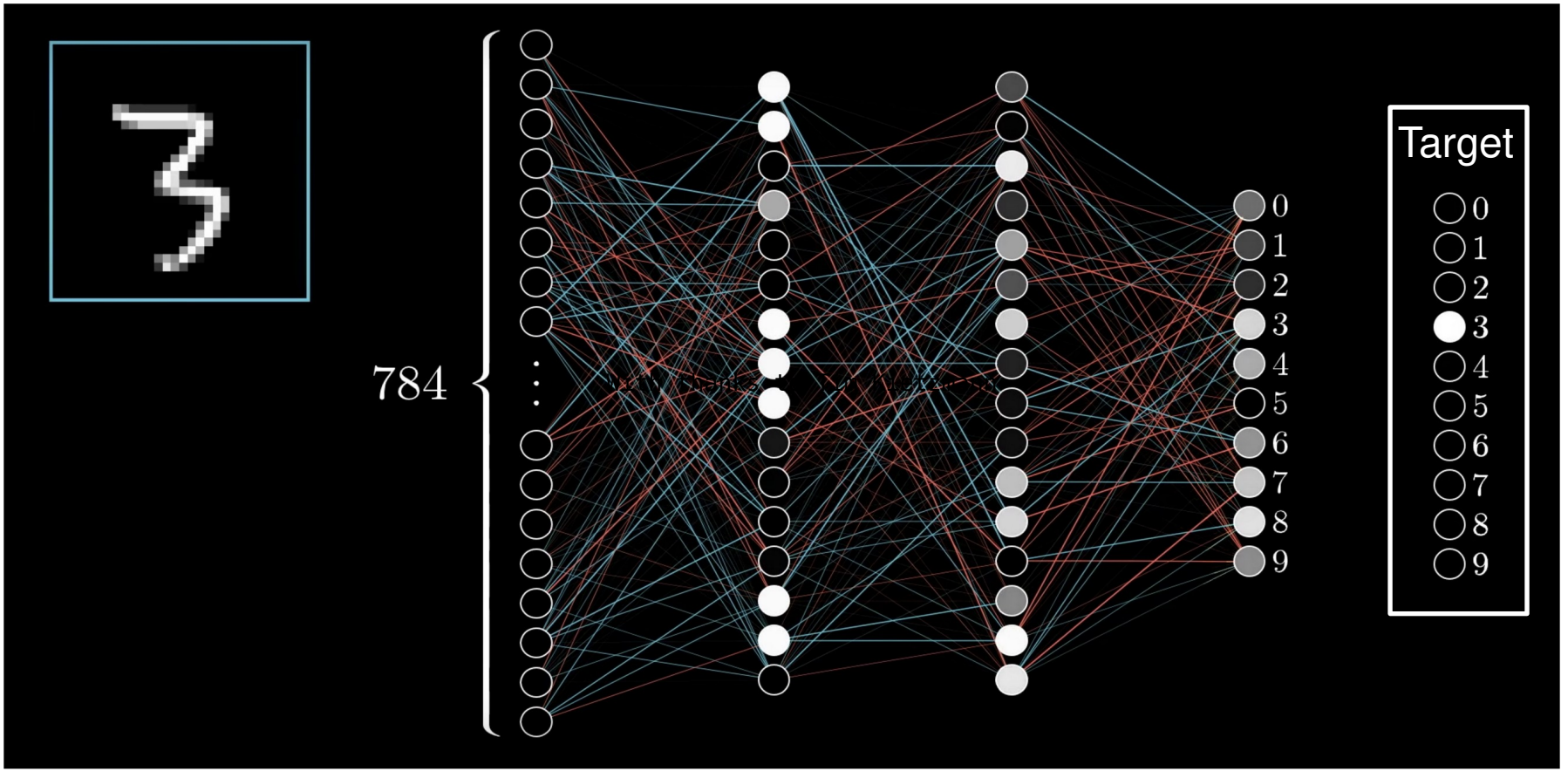
Input Layer:
Pixels of an image



Output Layer:
Probability that the image belongs to each category



With thanks to Tim Kietzmann



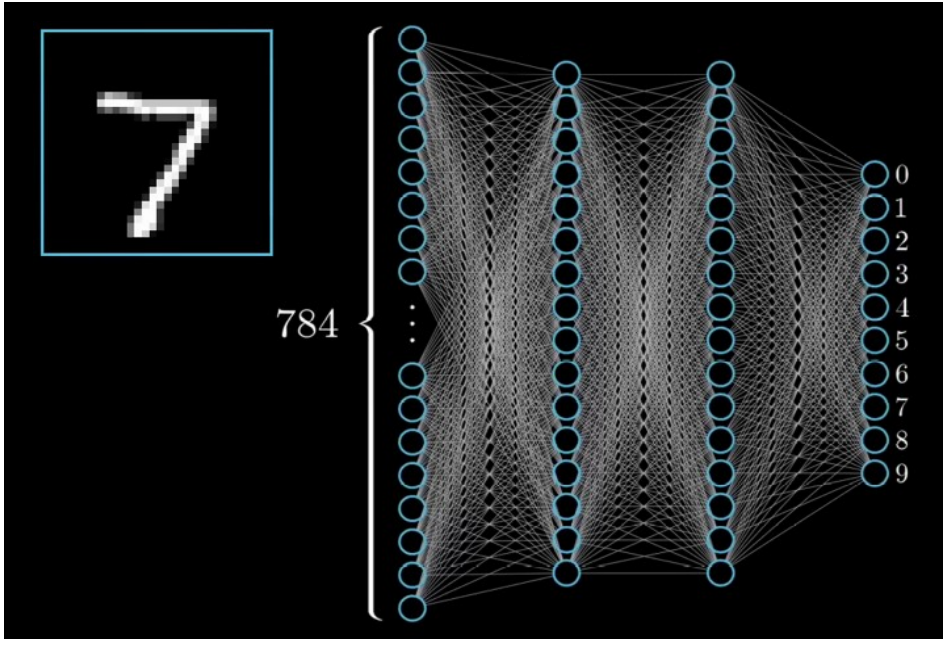
With thanks to Tim Kietzmann

python™ TensorFlow

BBSRC bioscience for the future | MRC Cognition and Brain Sciences Unit

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense


# Load MNIST data and reshape from 3D to 4D
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], 784)
x_test = x_test.reshape(x_test.shape[0], 784)




# Define the model architecture
inputs = Input(shape=(784,))
x = Dense(16)(inputs)
x = Dense(16)(x)
readout = Dense(10, activation=tf.nn.softmax)(x)
model = Model(inputs=inputs, outputs=readout)




# Define which loss & optimisation functions and metric(s) to use
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train and evaluate the model
model.fit(x=x_train, y=y_train, epochs=10)
model.evaluate(x_test, y_test)
```

```
x_train: (60000, 784)
y_train: (60000)
x_test:  (10000, 784)
y_test:  (10000)
```



Number of parameters: 13,002

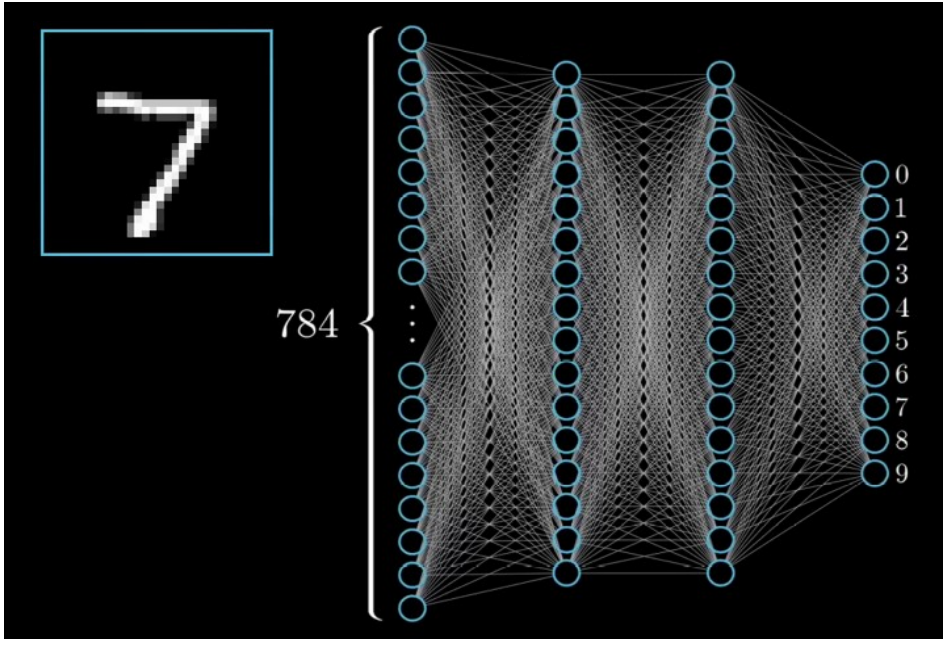
Accuracy: 87%

python TensorFlow

BBSRC bioscience for the future | MRC Cognition and Brain Sciences Unit

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense


# Load MNIST data and reshape from 3D to 4D
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], 784)
x_test = x_test.reshape(x_test.shape[0], 784)

# Rescale pixel values between 0 and 255 to values between 0 and 1
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255

# Define the model architecture
inputs = Input(shape=(784,))
x = Dense(16)(inputs)
x = Dense(16)(x)
readout = Dense(10, activation=tf.nn.softmax)(x)
model = Model(inputs=inputs, outputs=readout)




# Define which loss & optimisation functions and metric(s) to use
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train and evaluate the model
model.fit(x=x_train, y=y_train, epochs=10)
model.evaluate(x_test, y_test)
```

```
x_train: (60000, 784)
y_train: (60000)
x_test:  (10000, 784)
y_test:  (10000)
```



Number of parameters: 13,002

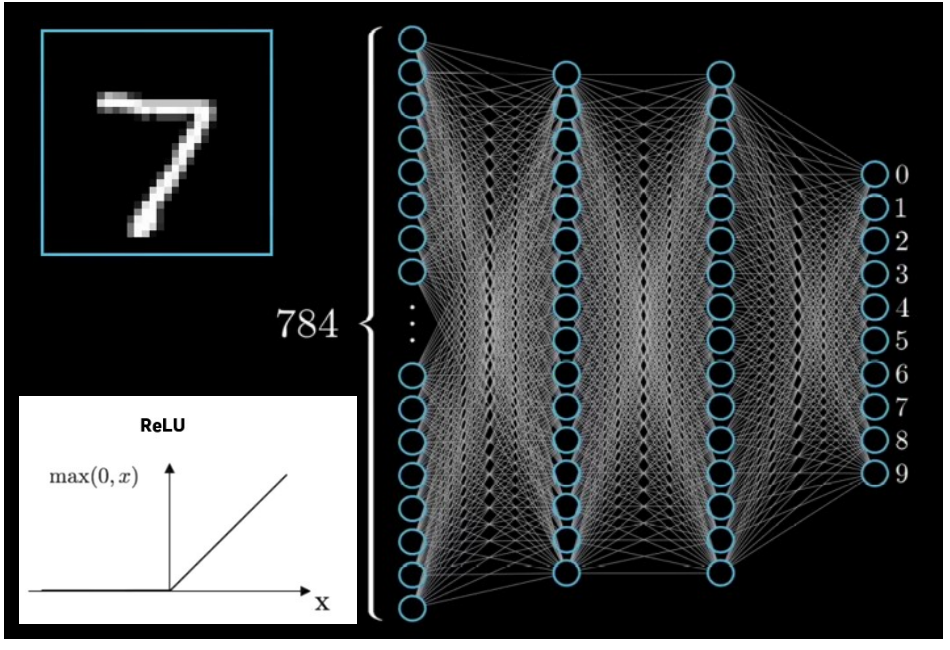
Accuracy: 93%

python TensorFlow

BBSRC bioscience for the future | MRC Cognition and Brain Sciences Unit

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense


# Load MNIST data and reshape from 3D to 4D
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], 784)
x_test = x_test.reshape(x_test.shape[0], 784)

# Rescale pixel values between 0 and 255 to values between 0 and 1
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255

# Define the model architecture
inputs = Input(shape=(784,))
x = Dense(16, activation=tf.nn.relu)(inputs)
x = Dense(16, activation=tf.nn.relu)(x)
readout = Dense(10, activation=tf.nn.softmax)(x)
model = Model(inputs=inputs, outputs=readout)




# Define which loss & optimisation functions and metric(s) to use
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train and evaluate the model
model.fit(x=x_train, y=y_train, epochs=10)
model.evaluate(x_test, y_test)
```

```
x_train: (60000, 784)
y_train: (60000)
x_test:  (10000, 784)
y_test:  (10000)
```



Number of parameters: 13,002

Accuracy: 96%

python™ TensorFlow

BBSRC bioscience for the future | MRC Cognition and Brain Sciences Unit

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Conv2D, Flatten, MaxPooling2D


# Load MNIST data and reshape from 3D to 4D
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape + (1,))
x_test = x_test.reshape(x_test.shape + (1,))

# Rescale pixel values between 0 and 255 to values between 0 and 1
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255

# Define the model architecture
inputs = Input(shape=(28,28,1))
x = Conv2D(8, kernel_size=(3,3))(inputs)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Conv2D(8, kernel_size=(3,3))(x)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Flatten()(x)
readout = Dense(10, activation=tf.nn.softmax)(x)
model = Model(inputs=inputs, outputs=readout)


# Define which loss & optimisation functions and metric(s) to use
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train and evaluate the model
model.fit(x=x_train, y=y_train, epochs=10)
model.evaluate(x_test, y_test)
```

x_train: (60000, 28, 28, 1)

Images: S Lazebnik

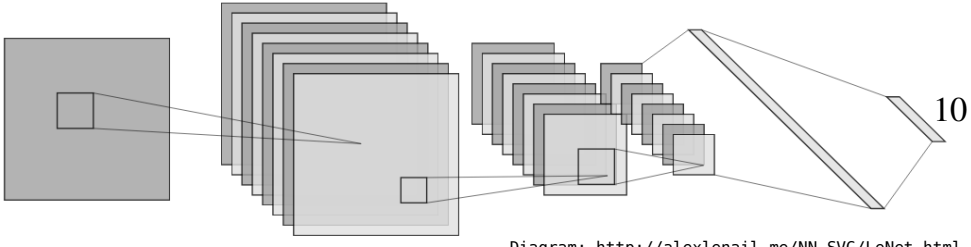


Diagram: http://alexlenail.me/NN-SVG/LeNet.html

Number of parameters: 2,674

Accuracy: 98%

python™ TensorFlow

BBSRC bioscience for the future | MRC Cognition and Brain Sciences Unit

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Conv2D, Flatten, MaxPooling2D


# Load MNIST data and reshape from 3D to 4D
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape + (1,))
x_test = x_test.reshape(x_test.shape + (1,))

# Rescale pixel values between 0 and 255 to values between 0 and 1
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255

# Define the model architecture
inputs = Input(shape=(28,28,1))
x = Conv2D(8, kernel_size=(3,3))(inputs)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Conv2D(8, kernel_size=(3,3))(x)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Flatten()(x)
readout = Dense(10, activation=tf.nn.softmax)(x)
model = Model(inputs=inputs, outputs=readout)
model.summary()

# Define which loss & optimisation functions and metric(s) to use
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train and evaluate the model
model.fit(x=x_train, y=y_train, epochs=10)
model.evaluate(x_test, y_test)
```

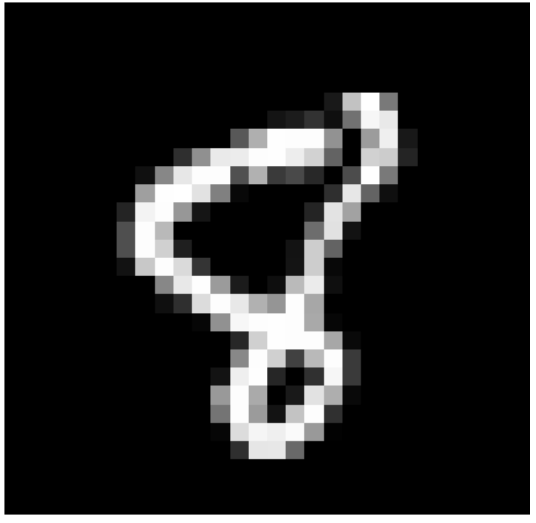
```
Train on 60000 samples
Epoch 1/10
60000/60000 [==============================] - 9s 147us/sample - loss: 0.3278 - accuracy: 0.9045
Epoch 2/10
60000/60000 [==============================] - 9s 147us/sample - loss: 0.1252 - accuracy: 0.9628
Epoch 3/10
60000/60000 [==============================] - 9s 155us/sample - loss: 0.0967 - accuracy: 0.9707
Epoch 4/10
60000/60000 [==============================] - 9s 150us/sample - loss: 0.0825 - accuracy: 0.9750
Epoch 5/10
60000/60000 [==============================] - 10s 159us/sample - loss: 0.0731 - accuracy: 0.9781
Epoch 6/10
60000/60000 [==============================] - 9s 150us/sample - loss: 0.0685 - accuracy: 0.9790
Epoch 7/10
60000/60000 [==============================] - 9s 147us/sample - loss: 0.0631 - accuracy: 0.9809
Epoch 8/10
60000/60000 [==============================] - 9s 147us/sample - loss: 0.0606 - accuracy: 0.9810
Epoch 9/10
60000/60000 [==============================] - 9s 150us/sample - loss: 0.0567 - accuracy: 0.9827
Epoch 10/10
60000/60000 [==============================] - 9s 144us/sample - loss: 0.0541 - accuracy: 0.9831

10000/10000 [==============================] - 1s 71us/sample - loss: 0.0565 - accuracy: 0.9822
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 26, 26, 8)         80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 8)         584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 8)           0
_________________________________________________________________
flatten (Flatten)            (None, 200)               0
_________________________________________________________________
dense (Dense)                (None, 10)                2010
=================================================================
Total params: 2,674
Trainable params: 2,674
Non-trainable params: 0
```

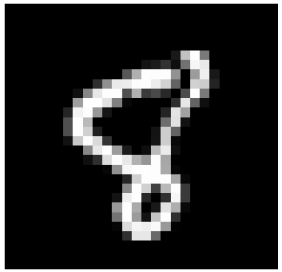
python™ TensorFlow

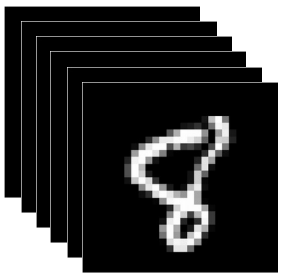
BBSRC bioscience for the future | MRC Cognition and Brain Sciences Unit

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Conv2D, Flatten, MaxPooling2D
from matplotlib import image

# Load MNIST data and reshape from 3D to 4D
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape + (1,))
x_test = x_test.reshape(x_test.shape + (1,))

# Rescale pixel values between 0 and 255 to values between 0 and 1
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255

# Define the model architecture
inputs = Input(shape=(28,28,1))
x = Conv2D(8, kernel_size=(3,3))(inputs)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Conv2D(8, kernel_size=(3,3))(x)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Flatten()(x)
readout = Dense(10, activation=tf.nn.softmax)(x)
model = Model(inputs=inputs, outputs=readout)
model.summary()

# Define which loss & optimisation functions and metric(s) to use
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train and evaluate the model
model.fit(x=x_train, y=y_train, epochs=10)
model.evaluate(x_test, y_test)

# Single image inference
test_image = image.imread('test_img.png')
pred = model.predict(test_image.reshape(1, 28, 28, 1))
print(pred.argmax())
```

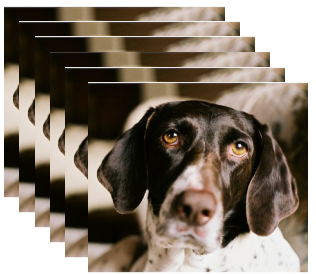


```
Predicted digit for test image: 8
```



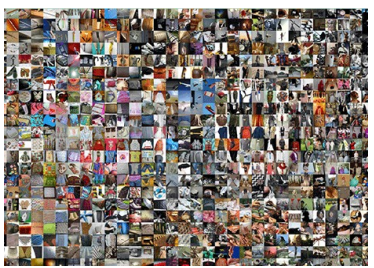


60,000



>1 million

10 classes



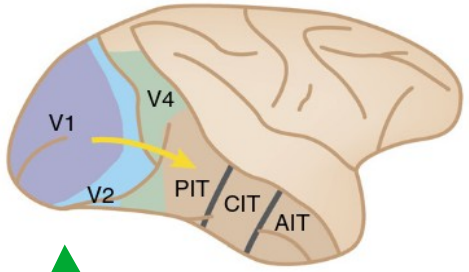
100s/1000s
of classes

- Data preprocessing
  (& augmentation)
- Validation set
- Saving checkpoints during training

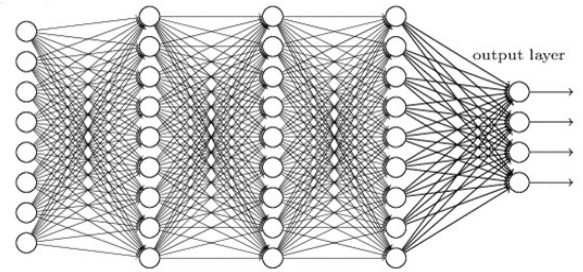
- Bigger models
  (10-100 million parameters)
- Normalisation, regularisation
- Lateral/top-down/skip connections

- Computational resources

Image sources: MNIST, ecoset, https://www.centreforthestudyof.net/?p=3428

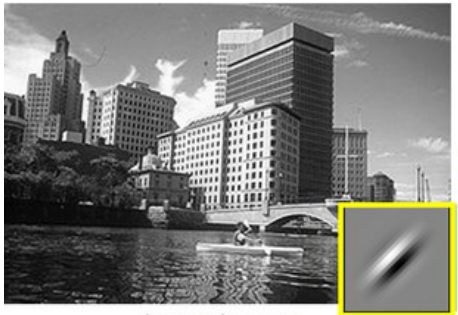


“Dog”



“Dog”

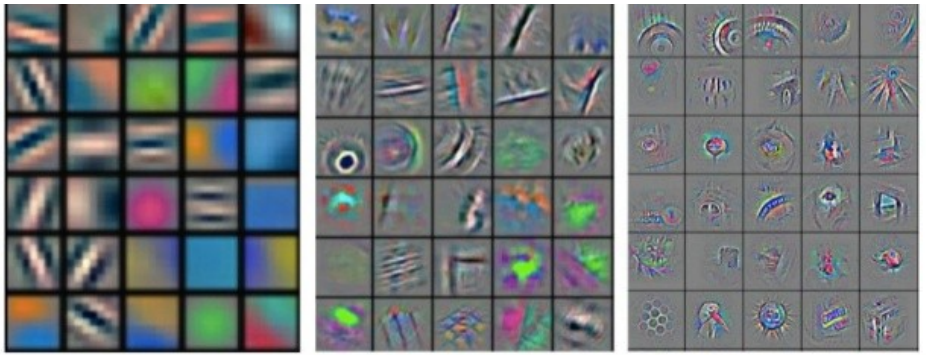
Images: Yamins & DiCarlo Nat Neurosci (2016); https://www.rsipvision.com/exploring-deep-learning/



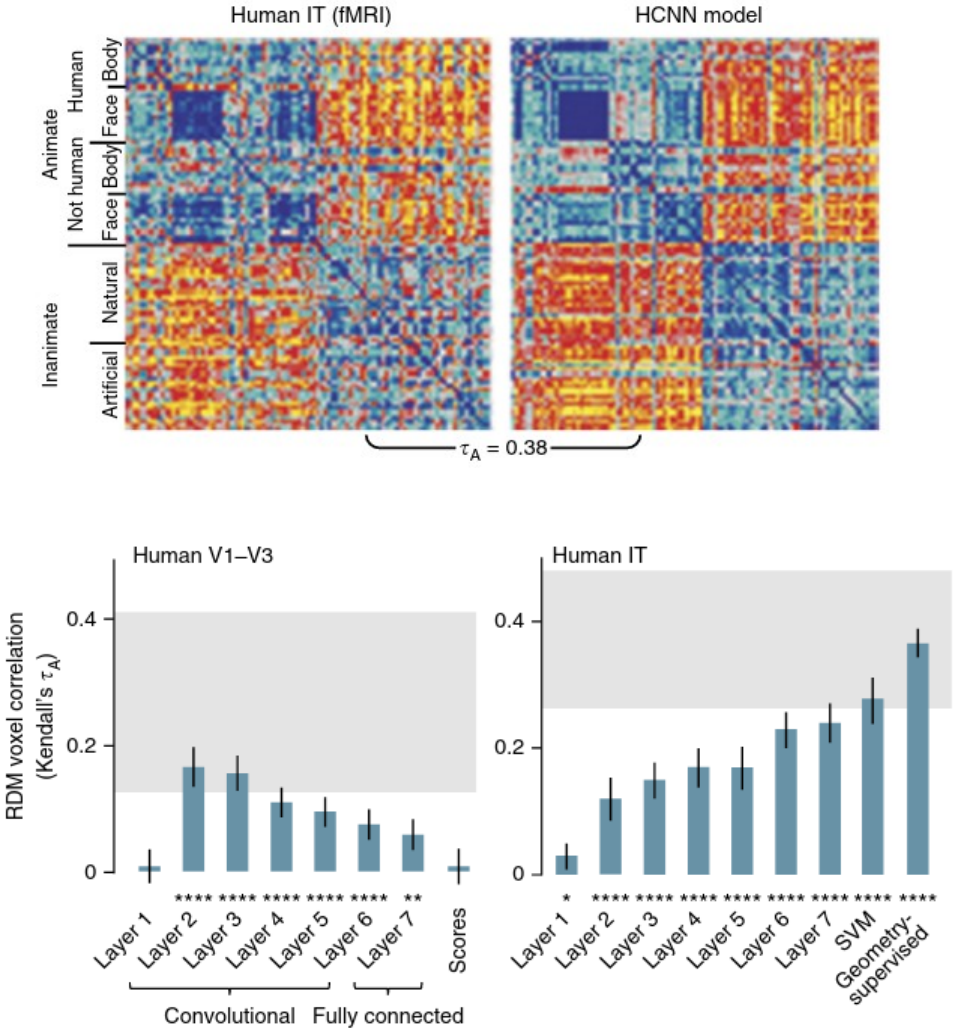
Input image

S Lazebnik

Input image



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



Yamins & DiCarlo, Nat Neurosci (2016), adapted from Khaligh-Razavi & Kriegeskorte, PloS Comput Biol (2014)

Deep Neural Networks in Computational Neuroscience

FREE

Tim C. Kietzmann, Patrick McClure and Nikolaus Kriegeskorte

https://doi.org/10
Published online:

Summary

The goal of co
the nervous sy
behavior. At th
computationa
stimuli to neu
range from sir
to dominate se
network" sugg
current DNNs
simplification
perform comp
and auditory s
on to motor co
addition to the
predicting neu
any other curr
which are requ
performance.
black boxes, th
four directly m
objective, and

Perspective | Published: 28 October 2019

A deep learning framework for neuroscience

Blake A. Richards, Timothy P. Lillicrap, […] Konrad P. Kording

Nature Neuroscien
35k Accesses | **40**

Abstract

Systems neurosc
perceptual, cogn
computational s
networks, the th
learning rules an
utilizes brain-ins
become central t
systems. Here we
systems neurosc
drive theoretical
principled persp

Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future

Grace W. Lindsay

Abstract

■ Convolutional neural network
findings in the study of biological
successful tools in computer visio
both neural activity and behavi
highlights what, in the context
model in computational neuro

INTRODUCTION

Computational models serve
science. They can validate in
works by providing a way to
They also offer a means to ex
ideal experimental testing g

Perspective | Published: 16 November 2020

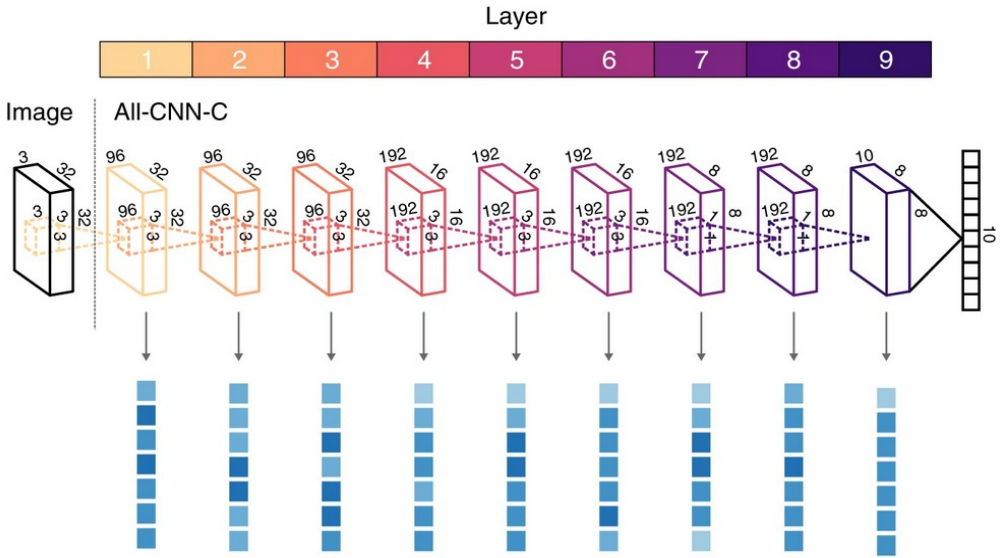
If deep learning is the answer, what is the question?

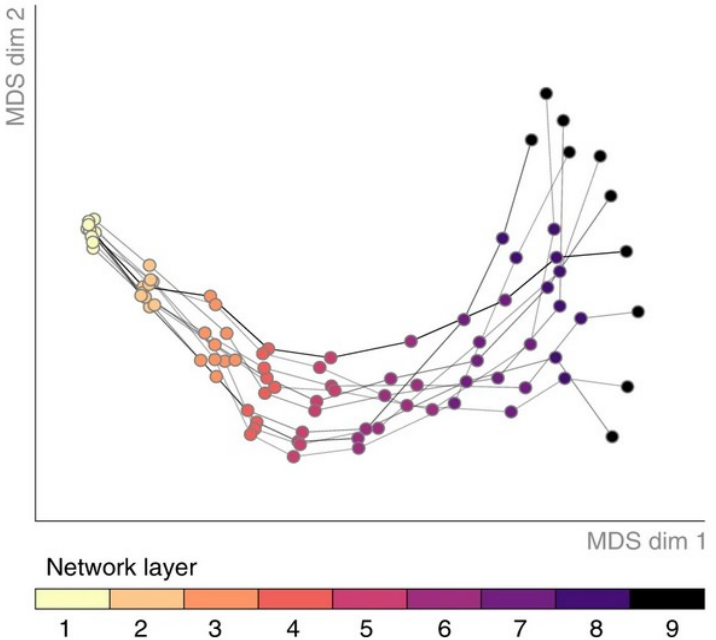
Andrew Saxe, Stephanie Nelli & Christopher Summerfield

Nature Reviews Neuroscience (2020) | Cite this article
4987 Accesses | **160** Altmetric | Metrics

Abstract

Neuroscience research is undergoing a minor revolution. Recent advances in machine learning and artificial intelligence research have opened up new ways of thinking about





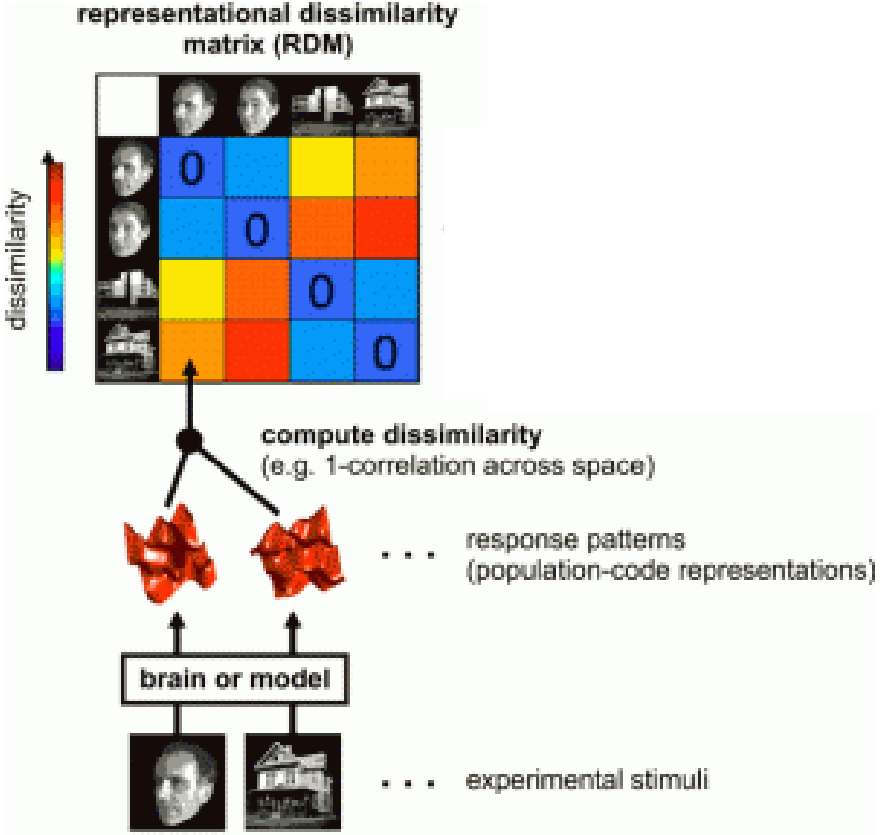
Models should be treated as participants

Mehrer, J. et al. Individual differences among deep neural network models. Nat Commun (2020)

ecj35@cam.ac.uk

 emercjones

 github.com/emercjones/dnn-examples



Kriegeskorte, N et al. Front Syst Neurosci (2008)