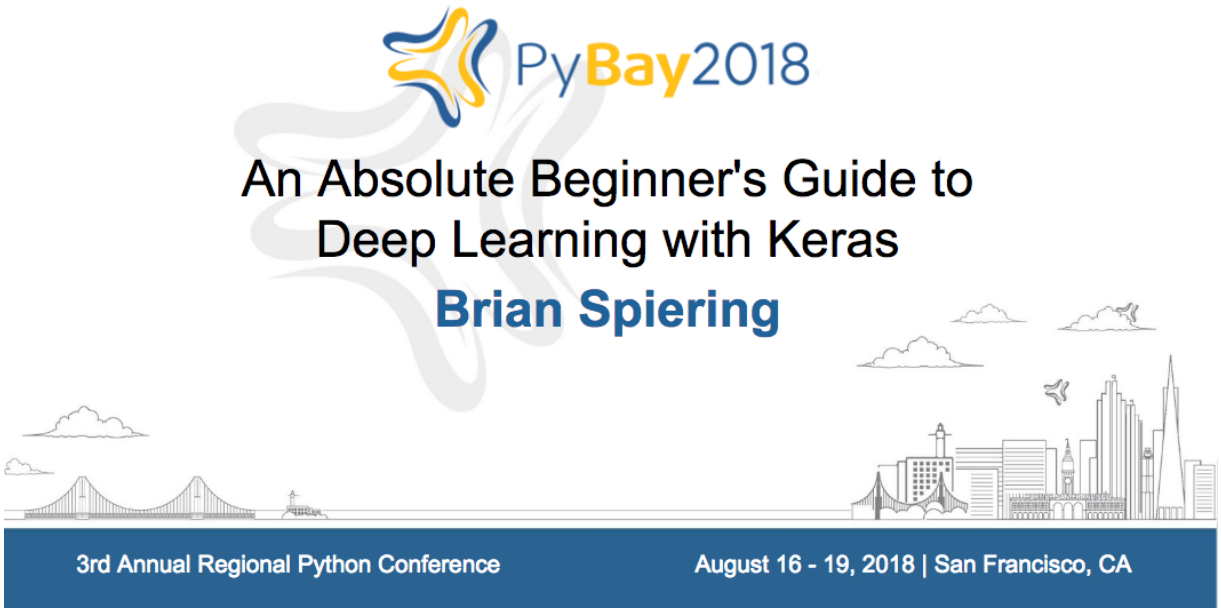


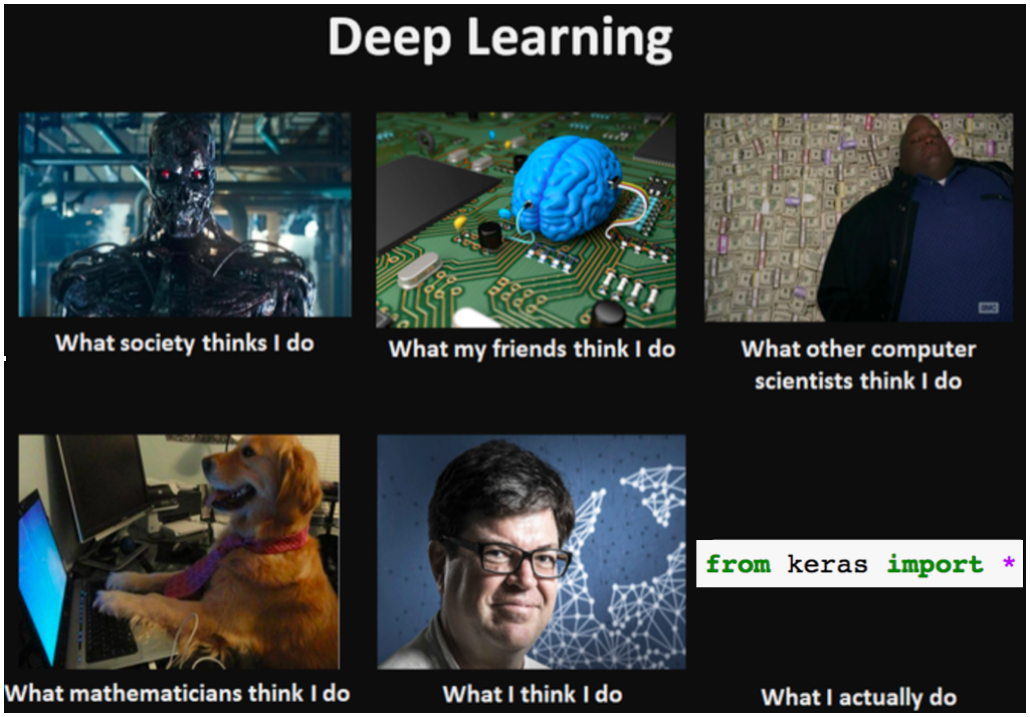
bit.ly/pybay-keras (http://bit.ly/pybay-keras)

Who Am I?

Brian Spiering

What Do I Do?

Professor @



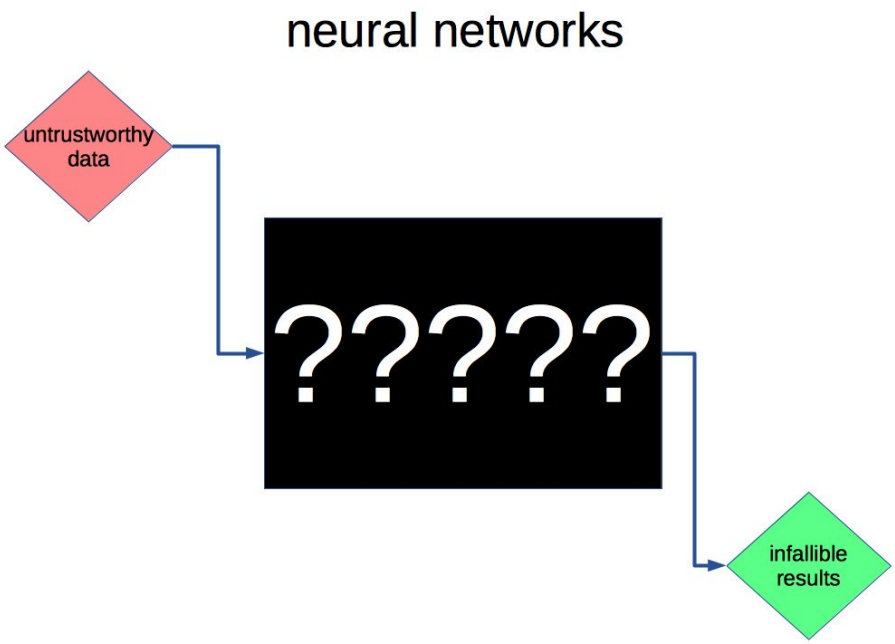
Keras - Neural Networks for humans

A high-level, intuitive API for Deep Learning.

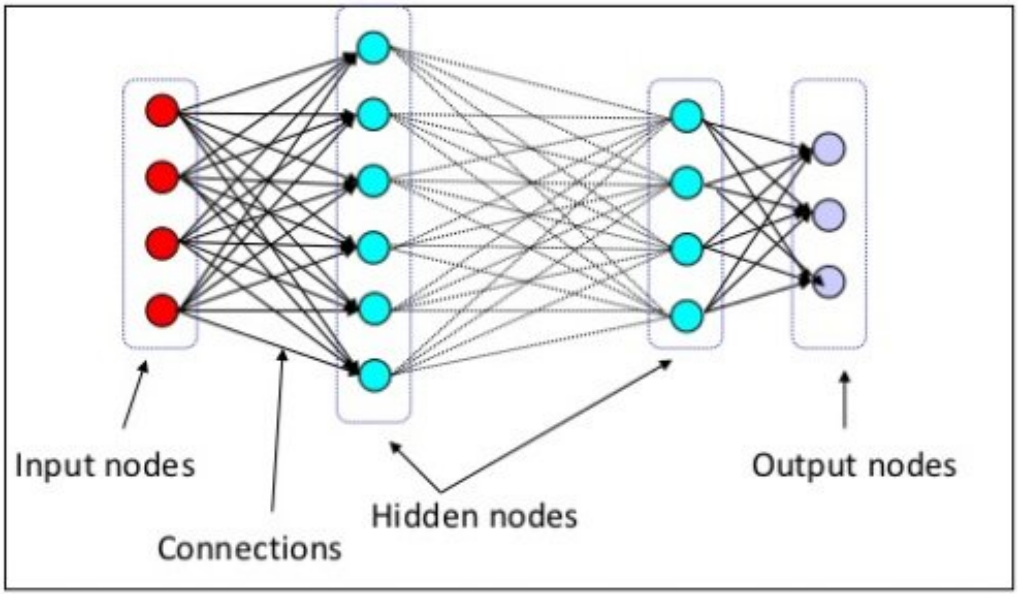
Easy to define neural networks, then automatically handles execution.

A simple, modular interface which allows focus on learning and enables fast experimentation

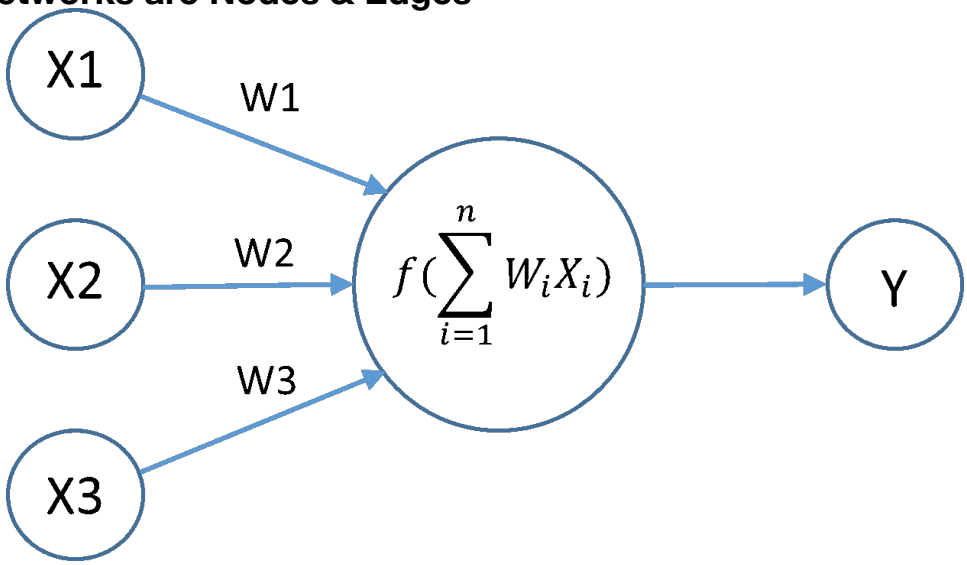
Deep Learning 101



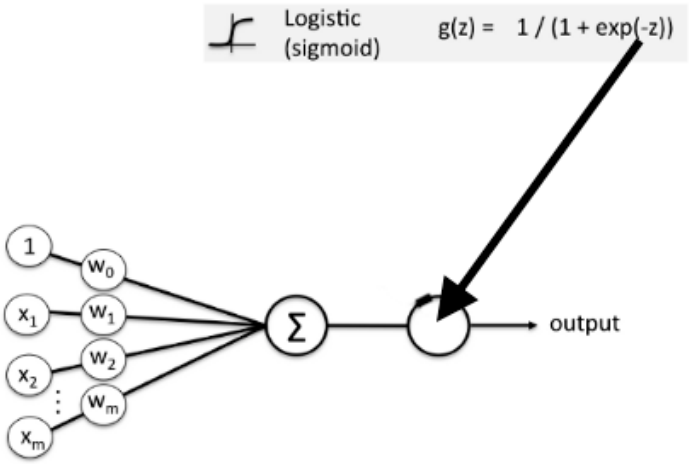
Deep Learning (DL) are Neural networks (NN) with >1 hidden layer



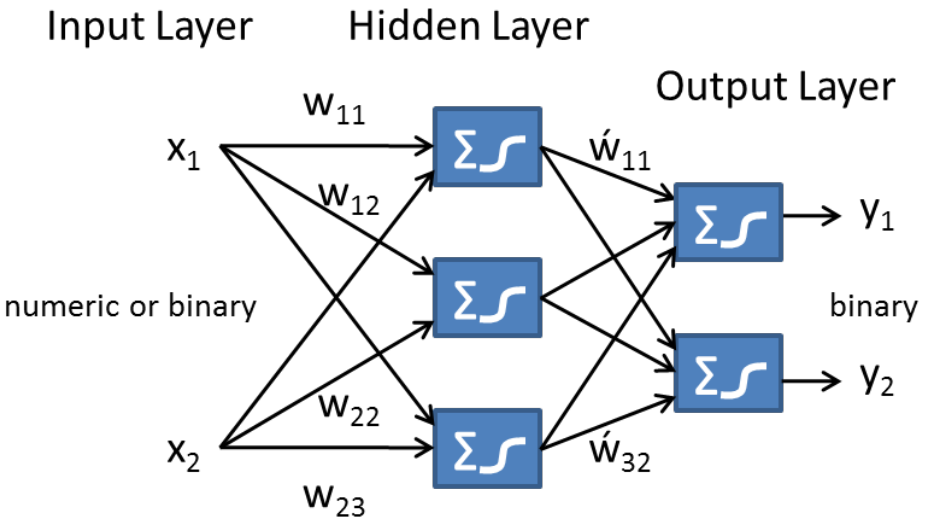
Neural Networks are Nodes & Edges



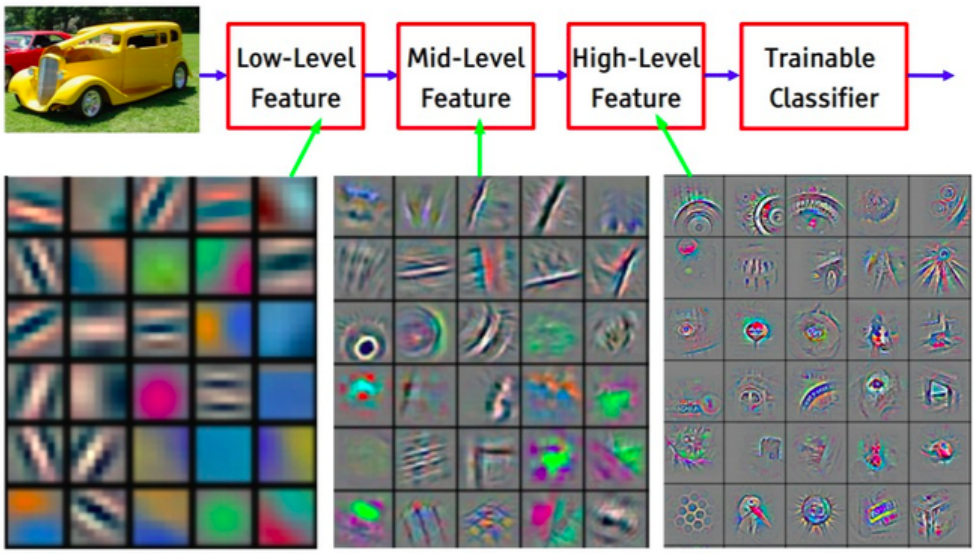
Nonlinear function allows learning of nonlinear relationships



Groups of nodes all the way down



Deep Learning isn't magic, it is just very good at finding patterns



Deep Learning has fewer steps than traditional Machine Learning

Traditional Machine Learning Flow

Deep Learning Flow

If you want to follow along…

GitHub repo: bit.ly/pybay-keras (http://bit.ly/pybay-keras)

If you want to type along…

1. **Run a local Jupyter Notebook**
2. **Binder (https://mybinder.org/v2/gh/brianspiering/keras-intro/master): In-Browser Jupyter Notebook**
3. **Colaboratory (https://colab.research.google.com/): "Google Docs for Jupyter Notebooks"**

In [1]:
```
reset -fs
```

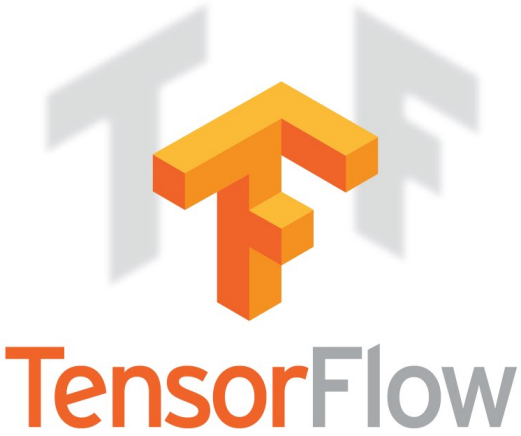
In [2]:
```
import keras
```

```
/Users/brian/anaconda3/envs/keras/lib/python3.6/importlib/_bootstrap.py:205: Run
timeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expe
cted 96, got 88
  return f(*args, **kwds)
Using TensorFlow backend.
```

In [3]:
```
# What is the backend / execution engine?
```

In [4]:
```
keras.backend.backend()
```

Out[4]:
```
'tensorflow'
```



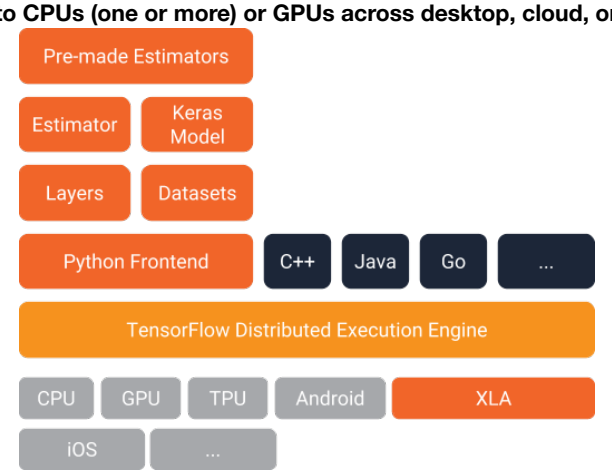
"An open-source software library for Machine Intelligence"

Numerical computation using data flow graphs.

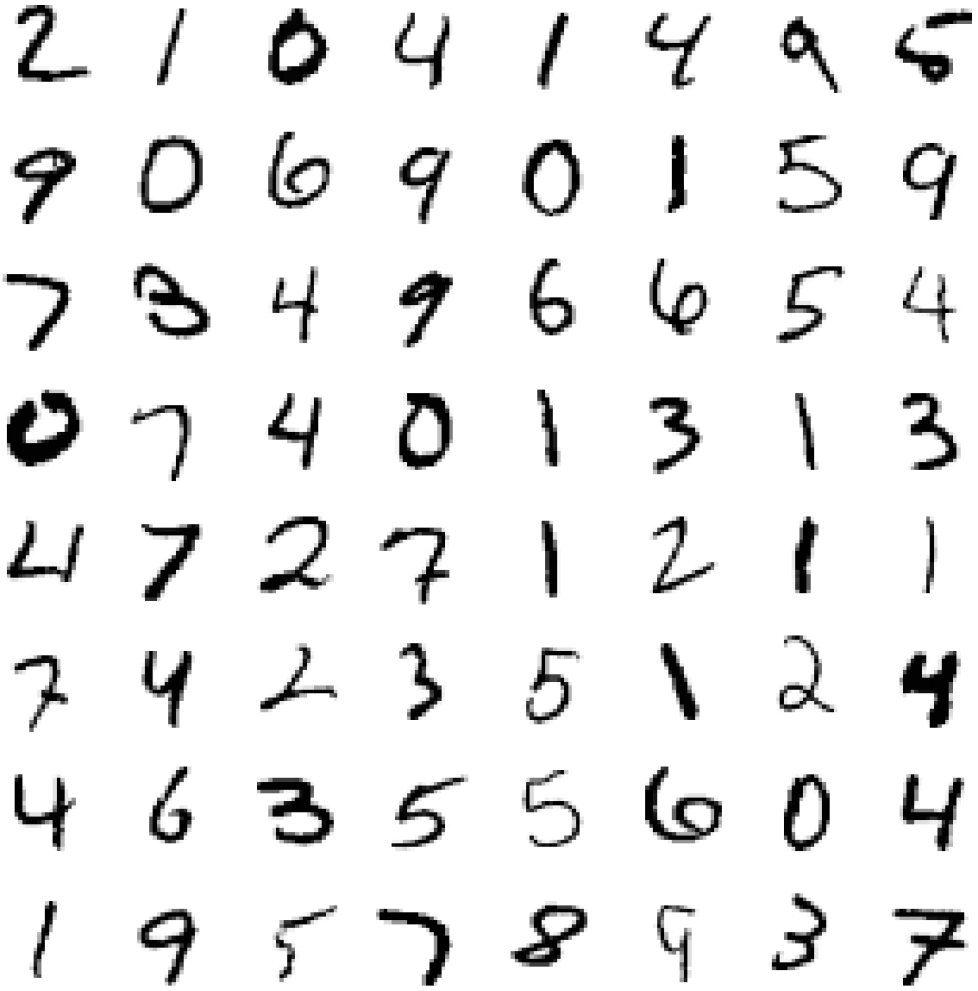
TensorFlow: A great backend

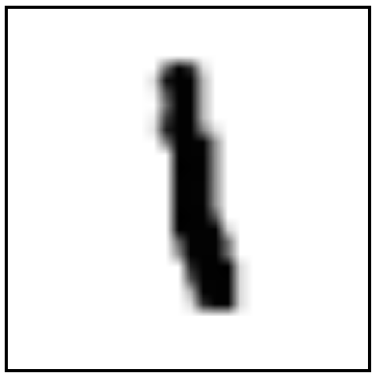
A very flexible architecture that allows you to do almost any operation.

Then deploy the computation to CPUs (one or more) or GPUs across desktop, cloud, or mobile device.



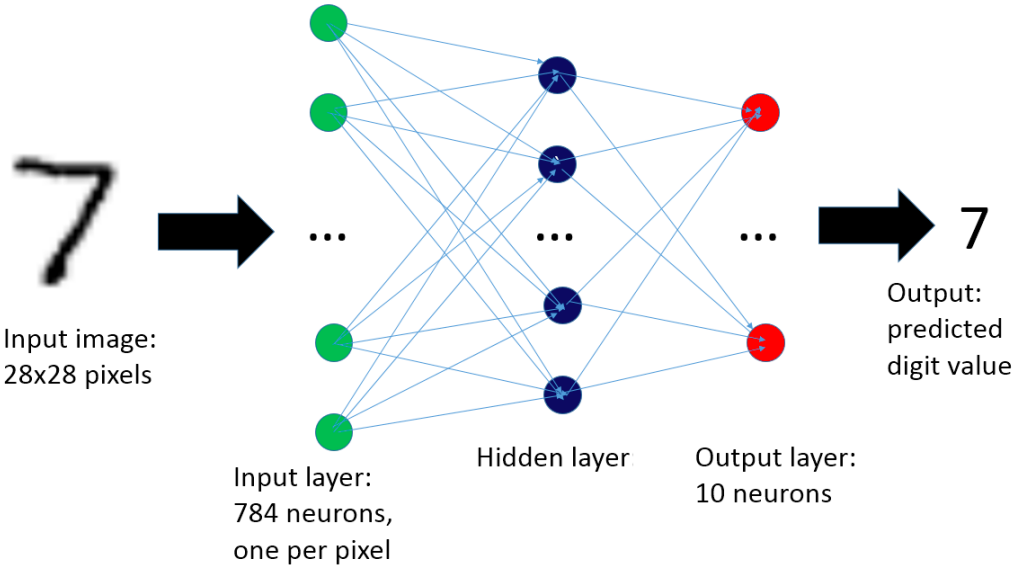
MNIST handwritten digit database:
The “Hello World!” of Computer Vision





$\approx$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$



In [5]:
```
# Import data
```

In [6]:
```
from keras.datasets import mnist
```

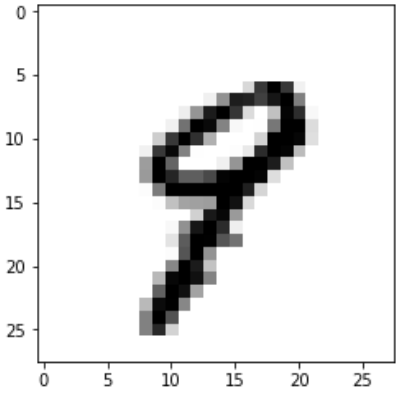
In [7]:
```
# Setup train and test splits
```

In [8]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

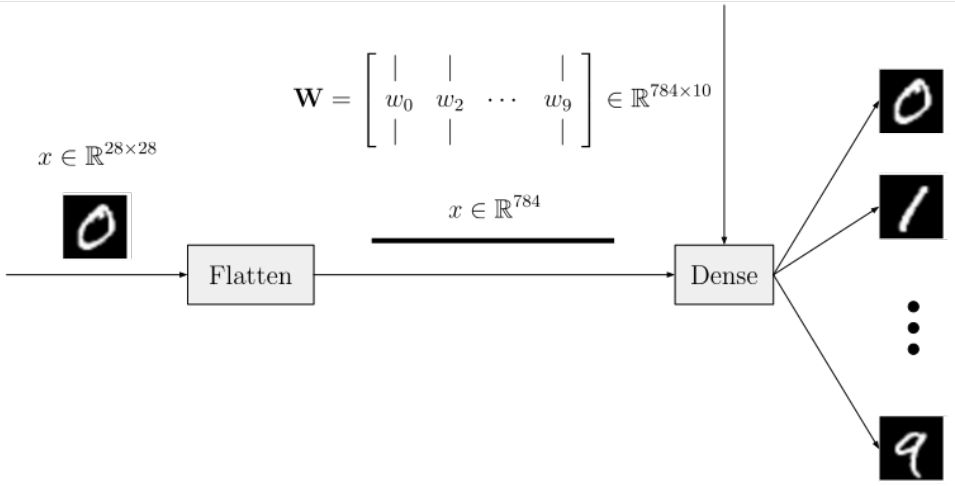
In [9]:
```
from random import randint
from matplotlib import pyplot

%matplotlib inline
```

In [10]:
```
pyplot.imshow(x_train[randint(0, x_train.shape[0])], cmap='gray_r');
```



Munge data



Convert image matrix into vector to feed into first layer

In [11]:
```
# Munge Data
# Transform from matrix to vector, cast, and normalize
```

In [12]:
```
image_size = 784 # 28 x 28

x_train = x_train.reshape(x_train.shape[0], image_size) # Transform from matrix to
vector
x_train = x_train.astype('float32') # Cast as 32 bit integers
x_train /= 255 # Normalize inputs from 0-255 to 0.0-1.0

x_test = x_test.reshape(x_test.shape[0], image_size) # Transform from matrix to ve
ctor
x_test = x_test.astype('float32') # Cast as 32 bit integers
x_test /= 255 # Normalize inputs from 0-255 to 0.0-1.0
```

In [13]:
```
# Convert class vectors to binary class matrices
```

In [14]:
```
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

In [15]:
```
# Import the most common type of neural network
```

In [16]:
```
from keras.models import Sequential
```

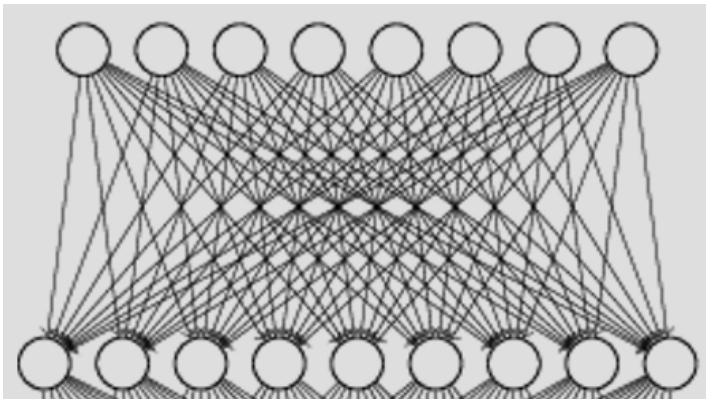
RTFM - https://keras.io/layers/ (https://keras.io/layers/)

In [17]:
```
# Define model instance
```

In [18]:
```
model = Sequential()
```

In [19]:
```
# Import the most common type of network layer, fully interconnected
```

In [20]:
```
from keras.layers import Dense
```



In [21]:
```
# Define input layer
```

In [22]:
```
layer_input = Dense(units=512,            # Number of nodes
                    activation='sigmoid', # The nonlinearity
                    input_shape=(image_size,))
model.add(layer_input)
```

In [23]:
```
# Define another layer
```

In [24]:
```
model.add(Dense(units=512, activation='sigmoid'))
```

In [25]:
```
# Define output layers
```

In [26]:
```
layer_output = Dense(units=10,             # Number of digits (0-9)
                     activation='softmax') # Convert neural activation to probabil
ity of category

model.add(layer_output)
```

In [27]:
```
# Print summary
```

In [28]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

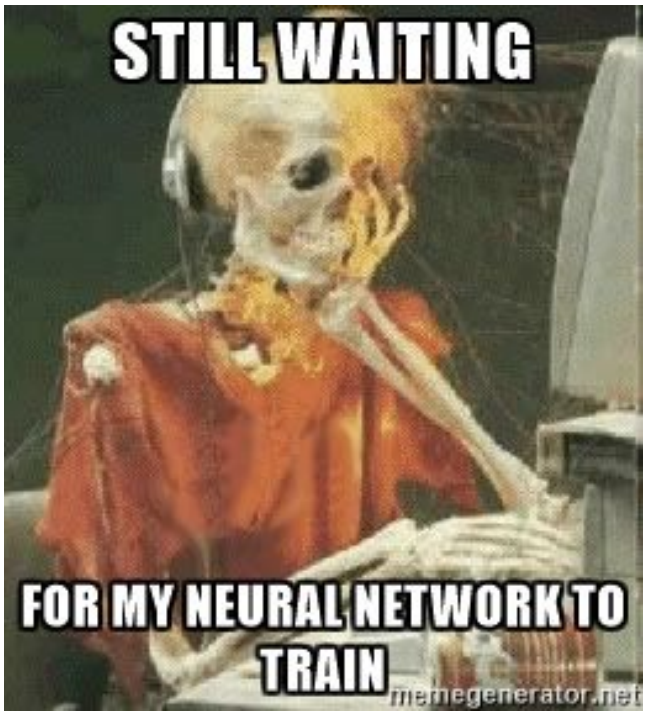
In [29]:
```
# Yes - we compile the model to run it
```

In [30]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

In [31]:
```
# Train the model
```

In [32]:
```
training = model.fit(x_train,
                     y_train,
                     epochs=5, # Number of passes over complete dataset
                     verbose=True,
                     validation_split=0.1)
```

```
Train on 54000 samples, validate on 6000 samples
Epoch 1/5
54000/54000 [==============================] - 13s 236us/step - loss: 2.1548 - a
cc: 0.3199 - val_loss: 1.9109 - val_acc: 0.4287
Epoch 2/5
54000/54000 [==============================] - 13s 247us/step - loss: 1.5084 - a
cc: 0.6597 - val_loss: 1.0729 - val_acc: 0.7858
Epoch 3/5
54000/54000 [==============================] - 14s 250us/step - loss: 0.9052 - a
cc: 0.7871 - val_loss: 0.6801 - val_acc: 0.8432
Epoch 4/5
54000/54000 [==============================] - 14s 259us/step - loss: 0.6555 - a
cc: 0.8338 - val_loss: 0.5119 - val_acc: 0.8820
Epoch 5/5
54000/54000 [==============================] - 13s 248us/step - loss: 0.5399 - a
cc: 0.8579 - val_loss: 0.4344 - val_acc: 0.8862
```



In [33]:
```
# Let's see how well our model performs
```

In [34]:
```
loss, accuracy = model.evaluate(x_test,
                                y_test,
                                verbose=True)
print(f"Test loss: {loss:.3}")
print(f"Test accuracy: {accuracy:.3%}")
```

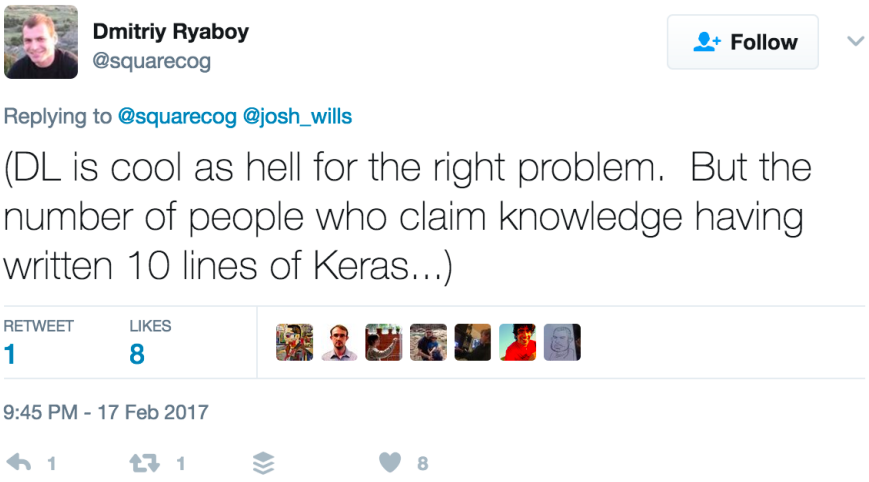
```
10000/10000 [==============================] - 1s 73us/step
Test loss: 0.482
Test accuracy: 86.530%
```

Keras' Other Features

- Common built-in functions (e.g., activation functions and optimiziters)
- Convolutional neural network (CNN or ConvNet)
- Recurrent neural network (RNN) & Long-short term memory (LSTM)
- Pre-trained models

Summary

- Keras is designed for human beings, not computers.
- Easier to try out Deep Learning (focus on the what, not the how).
- Simple to define neural networks.



Futher Study - Keras

- Keras docs
  - https://keras.io/ (https://keras.io/)
  - https://blog.keras.io/ (https://blog.keras.io/)
- Keras courses
  - https://www.edx.org/course/deep-learning-fundamentals-with-keras (https://www.edx.org/course/deep-learning-fundamentals-with-keras)
  - https://www.coursera.org/lecture/ai/keras-overview-7GfN9 (https://www.coursera.org/lecture/ai/keras-overview-7GfN9)

Futher Study - Deep Learning

- Linear Algebra, Probability, Machine Learning
- Study Deep Learning
  - fast.ai Course (http://www.fast.ai/)
  - Deep Learning Book (http://www.deeplearningbook.org/)

Bonus Material

In [35]:
```
reset -fs
```

In [36]:
```
from keras import *
```

In [37]:
```
whos
```

```
Variable          Type        Data/Info
-----------------------------------------
Input             function    <function Input at 0x127447488>
Model             type        <class 'keras.engine.training.Model'>
Sequential        type        <class 'keras.engine.sequential.Sequential'>
absolute_import   _Feature    _Feature((2, 5, 0, 'alpha<...>0, 0, 'alpha', 0), 1
6384)
activations       module      <module 'keras.activation<...>es/keras/activations
.py'>
applications      module      <module 'keras.applicatio<...>pplications/__init__
.py'>
backend           module      <module 'keras.backend' f<...>ras/backend/__init__
.py'>
callbacks         module      <module 'keras.callbacks'<...>ages/keras/callbacks
.py'>
constraints       module      <module 'keras.constraint<...>es/keras/constraints
.py'>
datasets          module      <module 'keras.datasets' <...>as/datasets/__init__
.py'>
engine            module      <module 'keras.engine' fr<...>eras/engine/__init__
.py'>
initializers      module      <module 'keras.initialize<...>s/keras/initializers
.py'>
layers            module      <module 'keras.layers' fr<...>eras/layers/__init__
.py'>
legacy            module      <module 'keras.legacy' fr<...>eras/legacy/__init__
.py'>
losses            module      <module 'keras.losses' fr<...>ackages/keras/losses
.py'>
metrics           module      <module 'keras.metrics' f<...>ckages/keras/metrics
.py'>
models            module      <module 'keras.models' fr<...>ackages/keras/models
.py'>
optimizers        module      <module 'keras.optimizers<...>ges/keras/optimizers
.py'>
preprocessing     module      <module 'keras.preprocess<...>eprocessing/__init__
.py'>
regularizers      module      <module 'keras.regularize<...>s/keras/regularizers
.py'>
utils             module      <module 'keras.utils' fro<...>keras/utils/__init__
.py'>
wrappers          module      <module 'keras.wrappers' <...>as/wrappers/__init__
.py'>
```

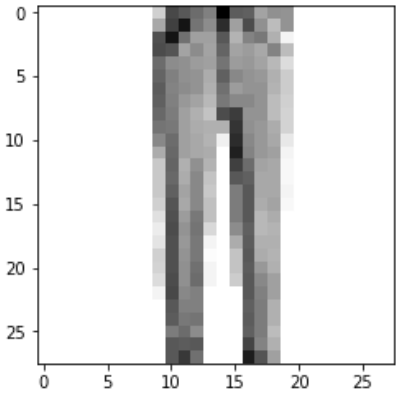
In [38]:
```
from keras.datasets import fashion_mnist
```

In [39]:
```
# Setup train and test splits
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

In [40]:
```
from random import randint
from matplotlib import pyplot

%matplotlib inline
```

In [41]:
```
pyplot.imshow(x_train[randint(0, x_train.shape[0])], cmap='gray_r');
```



In [42]:
```
# Define CNN model

# Redefine input dimensions to make sure conv works
img_rows, img_cols = 28, 28
x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)
```

In [43]:
```
import keras
```

In [44]:
```
# Convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

In [45]:
```
from keras.layers import Conv2D, Dense, Flatten, MaxPooling2D
```

In [46]:
```
# Define model
model = Sequential()
model.add(Conv2D(32,
             kernel_size=(3, 3),
             activation='sigmoid',
             input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='sigmoid'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='sigmoid'))
model.add(Dense(10, activation='softmax'))
```

In [47]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [48]:
```
# Define training
training = model.fit(x_train,
                     y_train,
                     epochs=5,
                     verbose=True,
                     validation_split=0.1)
```

```
Train on 54000 samples, validate on 6000 samples
Epoch 1/5
54000/54000 [==============================] - 173s 3ms/step - loss: 2.3262 - ac
c: 0.0986 - val_loss: 2.3071 - val_acc: 0.1055
Epoch 2/5
54000/54000 [==============================] - 173s 3ms/step - loss: 2.3100 - ac
c: 0.0996 - val_loss: 2.3076 - val_acc: 0.1003
Epoch 3/5
54000/54000 [==============================] - 173s 3ms/step - loss: 2.3107 - ac
c: 0.0994 - val_loss: 2.3047 - val_acc: 0.1032
Epoch 4/5
54000/54000 [==============================] - 187s 3ms/step - loss: 2.3111 - ac
c: 0.0989 - val_loss: 2.3290 - val_acc: 0.1055
Epoch 5/5
54000/54000 [==============================] - 182s 3ms/step - loss: 2.3101 - ac
c: 0.1000 - val_loss: 2.3131 - val_acc: 0.0925
```

In [49]:
```
loss, accuracy = model.evaluate(x_test,
                                y_test,
                                verbose=True)
print(f"Test loss: {loss:.3}")
print(f"Test accuracy: {accuracy:.3%}")
```

```
10000/10000 [==============================] - 9s 936us/step
Test loss: 2.31
Test accuracy: 10.000%
```

What is `keras`?

Keras (κέρας) means horn in Greek.

It is a reference to a literary image from ancient Greek and Latin literature.

First found in the Odyssey, where dream spirits (Oneiroi, singular Oneiros) are divided between those who deceive men with false visions, who arrive to Earth through a gate of ivory, and those who announce a future that will come to pass, who arrive through a gate of horn.

It's a play on the words κέρας (horn) / κραίνω (fulfill), and ἐλέφας (ivory) / ἐλεφαίρομαι (deceive).

Source (https://keras.io/#why-this-name-keras)