MNIST Convnet Sample Code

Tutorial on MNIST CNN:

First Reference: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/2.1-a-first-look-at-a-neural-network.ipynb

https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/5.1-introduction-to-convnets.ipynb

TF Keras has defined MNIST model and its training/testing image dataset, all you have to do is to import them into your program, which significantly save the time for learning. See github code below:

(1) The code for the handwritten digits recognition (to save trained model): 7-1convnets-NumeralDet-saveTrained.py
The folder: opencv/IP120-AI-DL/2018F/7-1convnets-NumeralDet-saveTrained.py /
The URL of the file: https://github.com/hualili/opencv/blob/master/IP120-AI-DL/2018F/7-1convnets-NumeralDet-saveTrained.py

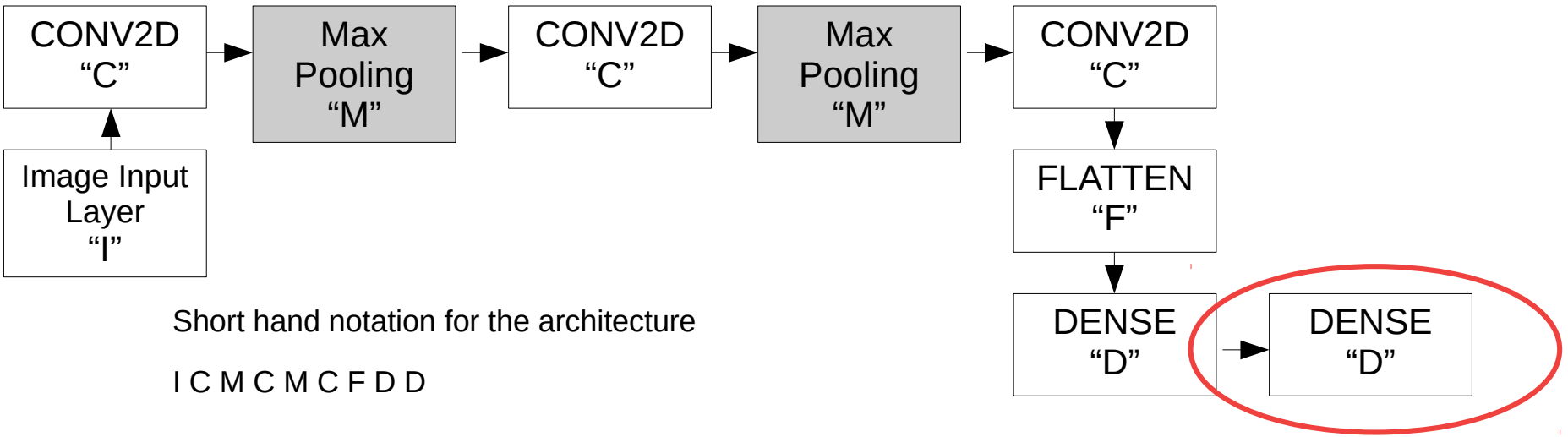
Step 1 Understand the Architecture (MNIST Convnet)

model.summary

```
>>> model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

MNIST Convnet Architecture

model.summary

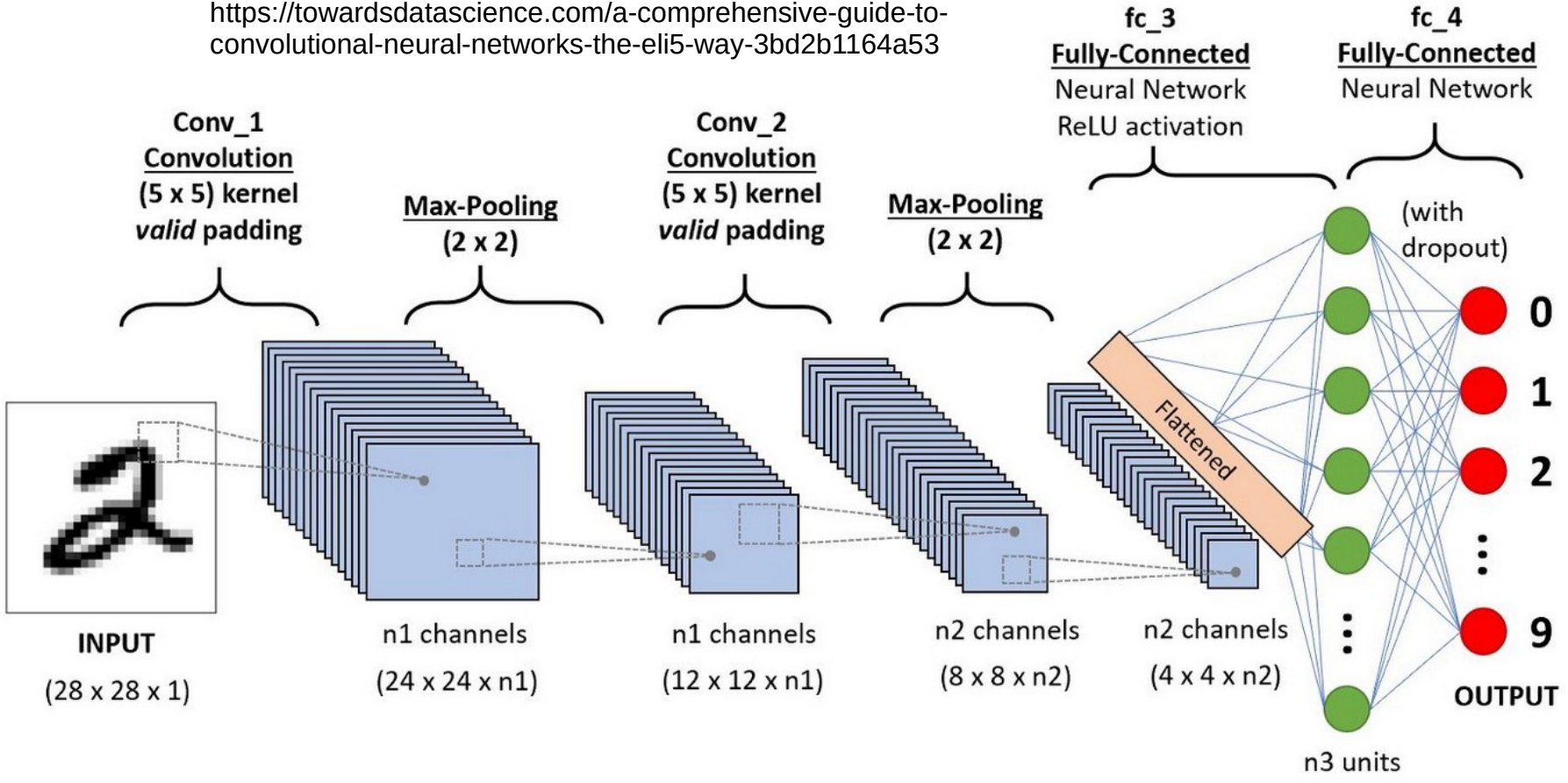


Short hand notation for the architecture

I C M C M C F D D

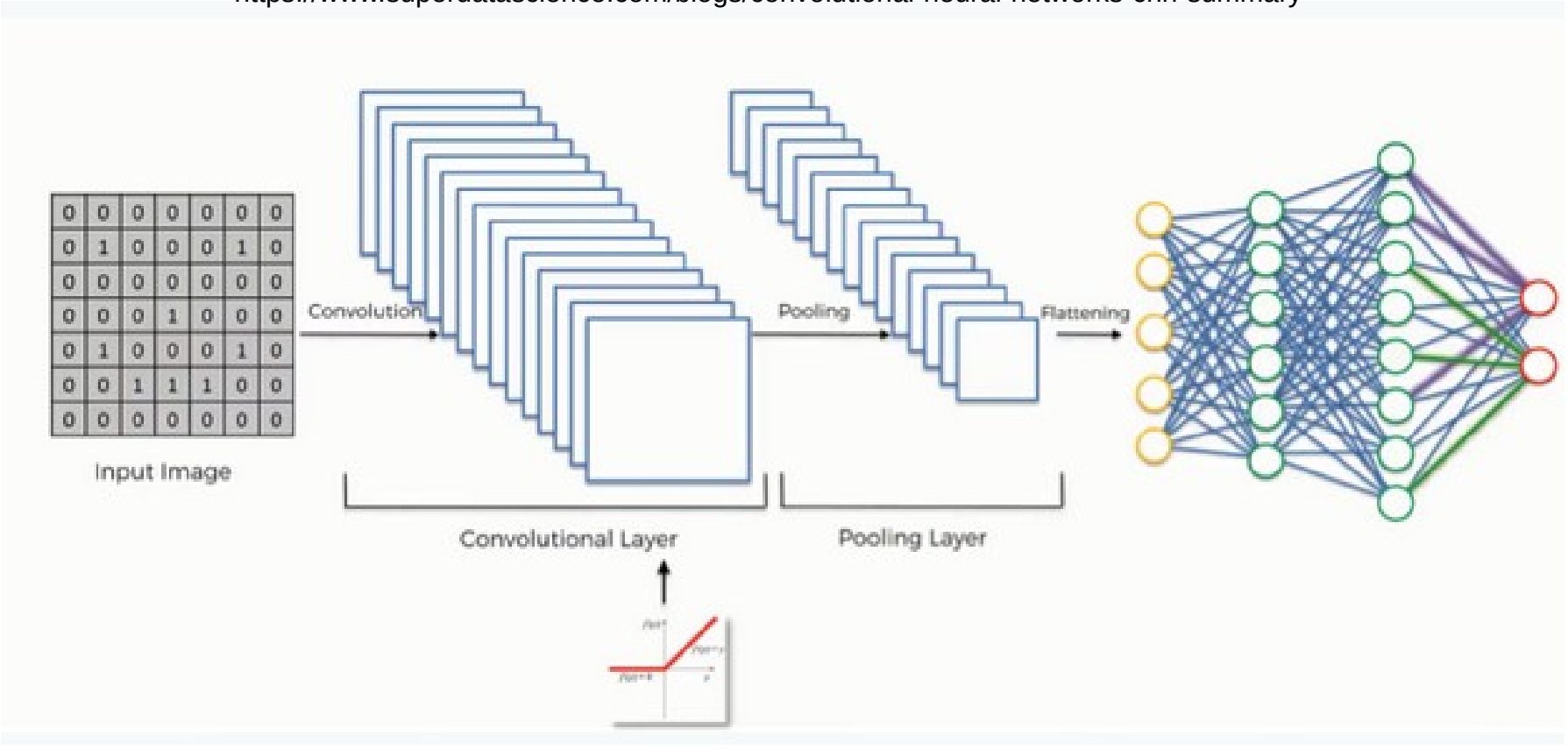
Illustration of A CNN for Digits Recognition

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53



Convolution Neural Networks

https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-summary



2D Convolution

Reference for the theoretical background: Chapter 6, Robot Vision, pp. 104 – 111, by BKP Horn, MIT Press

Definition:

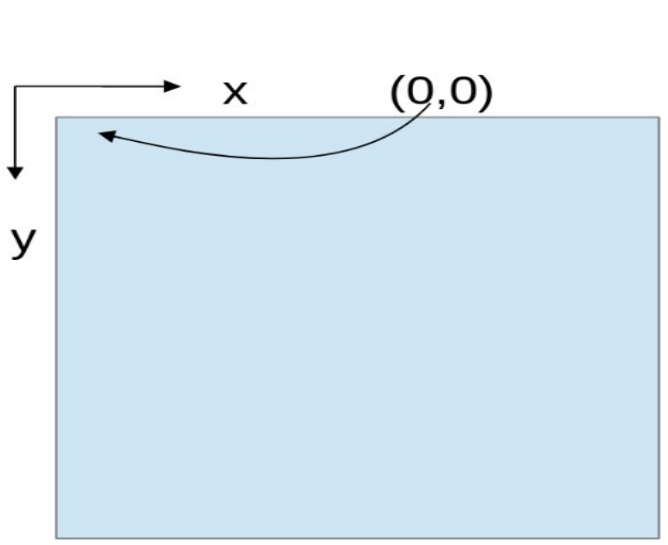
$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) \cdot g(x-\tau)\, d\tau$$

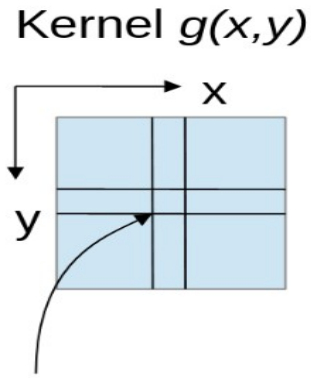
$$c(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2)$$

Image Kernel

Summation lower and upper bound in the case of M-by-N image f(x,y), should be adjusted to k1 = 0 to M-1, k2 = 0 to N-1

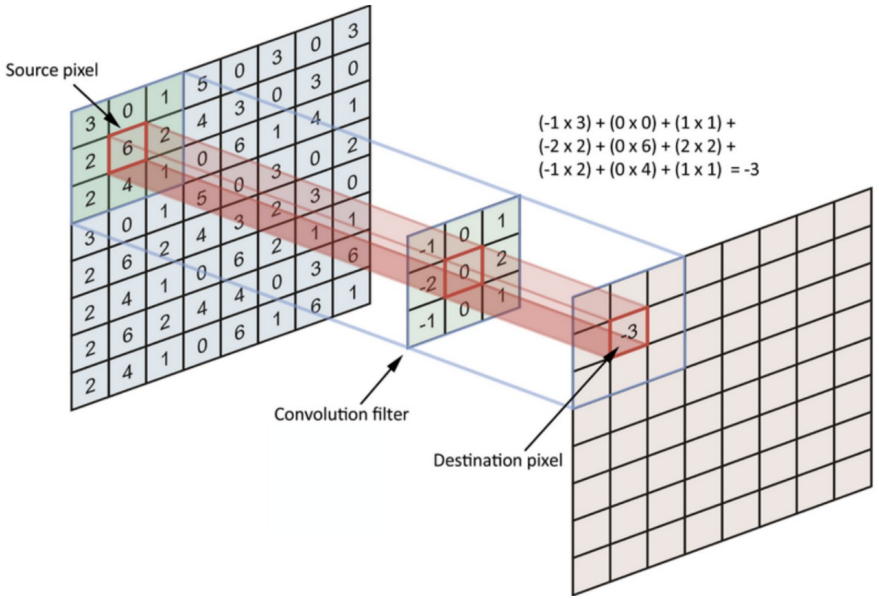
Reference for the OpenCV implementation: Learning OpenCV, Chapter 6, pp. 144 – 164.





Note: (1) 3 primitive computations: shift, multiplication, and addition;
(2) use discrete 2D convolution formula to compute 5x5 sample image with 3x3 kernels

2D Convolution Example



https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

2D Convolution with Matlab/Octave

C = conv2(A,B) computes the two-dimensional convolution of matrices A and B.

The size of C is determined as follows: if [ma,na] = size(A), [mb,nb] = size(B), Then [mc,nc] = size(C), Where mc = max([ma+mb-1,ma,mb]) and nc = max([na+nb-1,na,nb]).

Octave on Linux

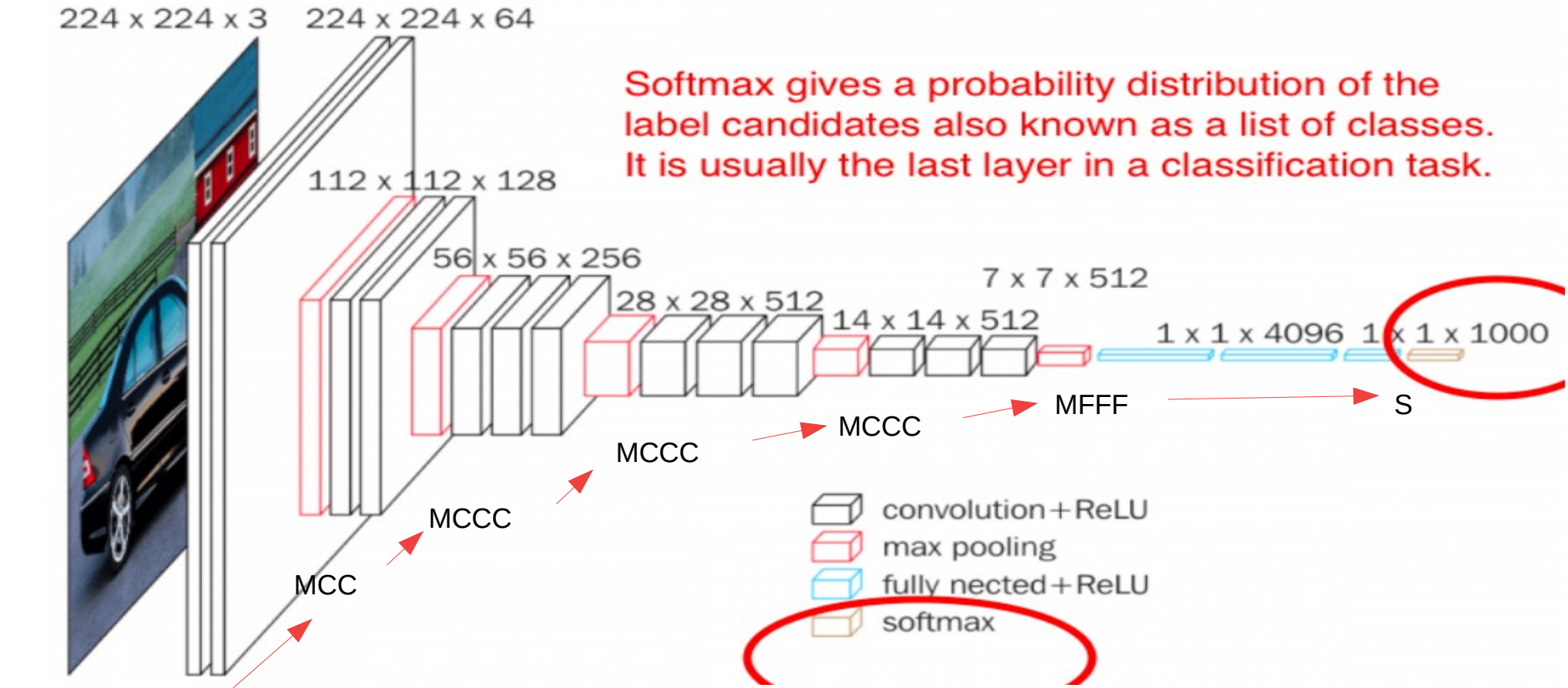
```
>> A = [ 0 0 100 100 100
         0 0 100 100 100
         0 0 100 100 100
         0 0 100 100 100
         0 0 100 100 100 ]


>> B = [ 1 0 -1
         1 0 -1
         1 0 -1 ]
```

```
C = conv2(A,B)
C =

    0    0  100  100    0 -100 -100
    0    0  200  200    0 -200 -200
    0    0  300  300    0 -300 -300
    0    0  300  300    0 -300 -300
    0    0  300  300    0 -300 -300
    0    0  200  200    0 -200 -200
    0    0  100  100    0 -100 -100
```

https://github.com/hualili/opencv/blob/master/deep-learning-2020S/1-2020S-%232019S-23-2DConvolution-2019-2-4.pdf

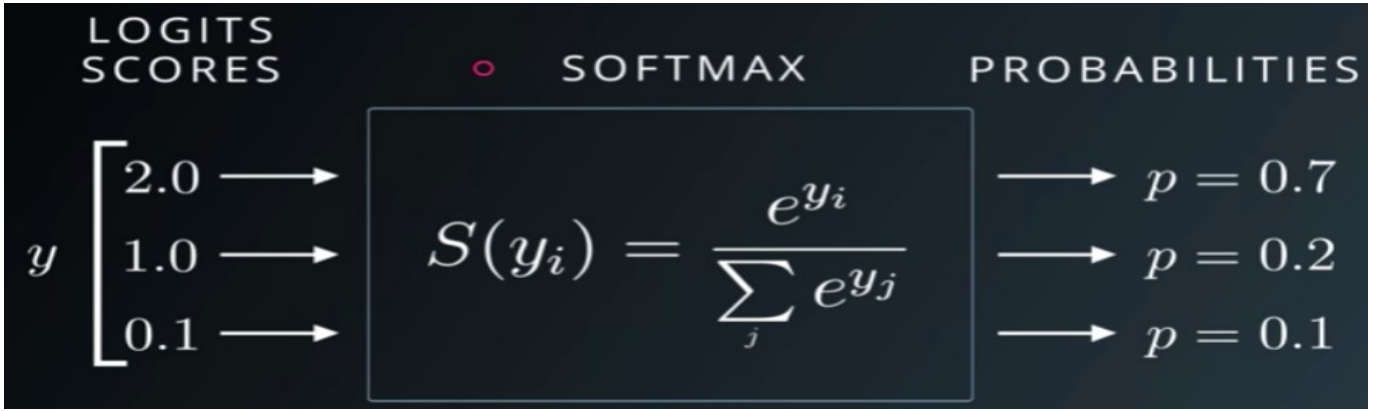
Architecture Example VGG16

https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d



Softmax Network

https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d



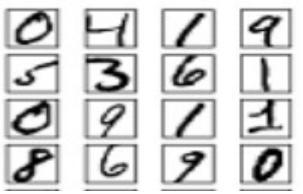
The above Udacity lecture slide shows that Softmax function turns logits [2.0, 1.0, 0.1] into probabilities [0.7, 0.2, 0.1], and the probabilities sum to 1.

“logits layer is popularly used for the last neuron layer for classification task which produces raw prediction values as real numbers ranging from [-infinity, +infinity]”

“Softmax’s input is the output of the fully connected layer immediately preceeding it, and it outputs the final output of the entire neural network. This output is a probability distribution of all the label class candidates.”

Output Layer

Example: NIST 10 digits (0-9) convnet Dense layer



Sample code : https://github.com/fchollet/deep-learning-with-python-notebooks

```
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'))
```

512 neurons in the layer

The network consists of 2 Dense layers, densely-connected ("fully-connected") neural layers. The output layer is a 10-way "softmax" layer, it returns an array of 10 probability scores (summing to 1). Each score will be the probability that the current digit image belongs to one of our 10 digit classes.

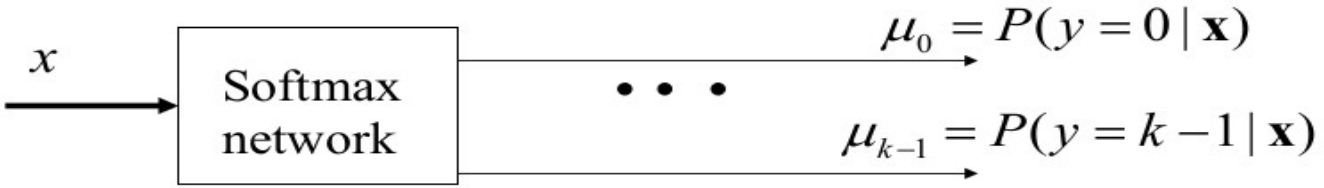
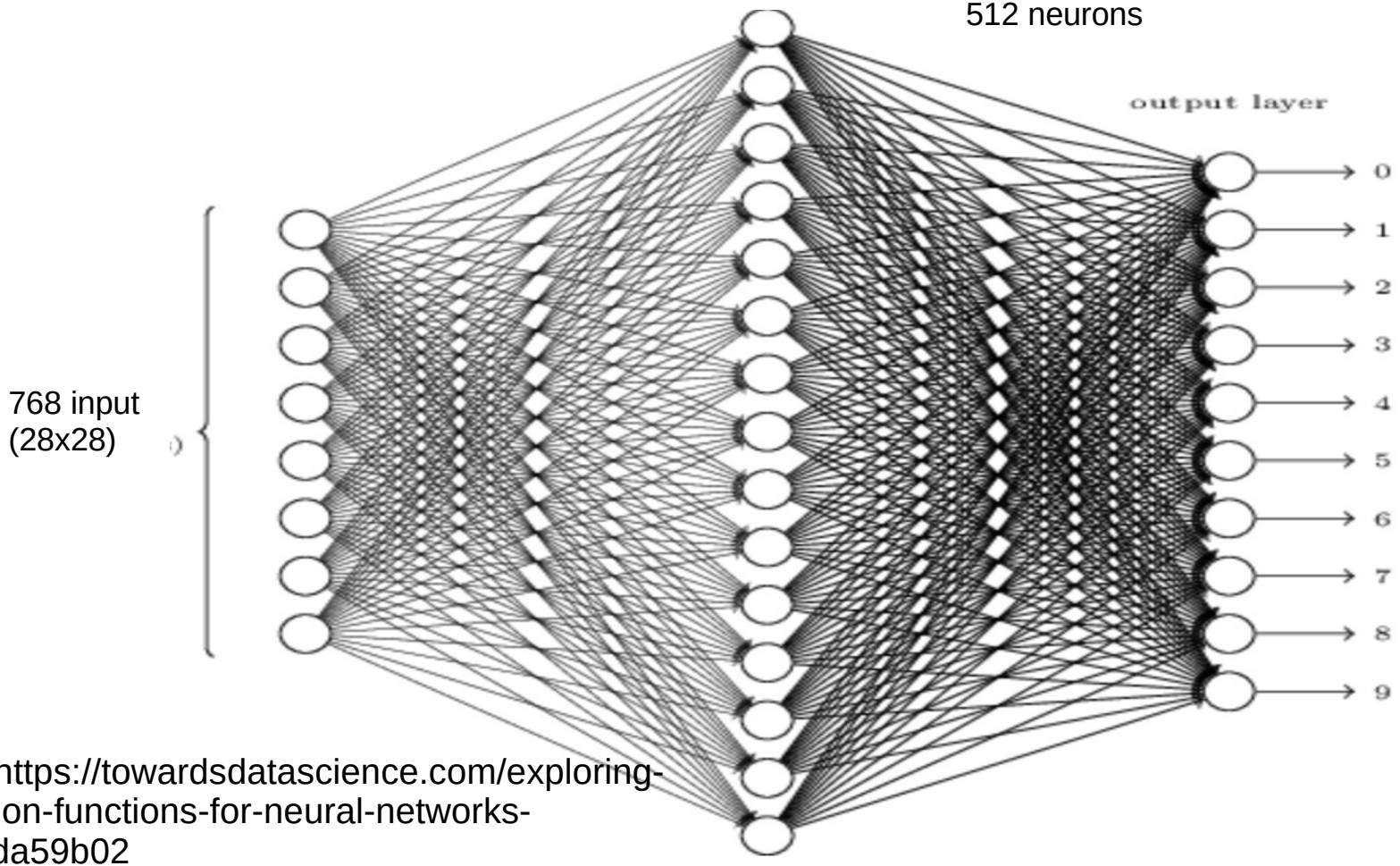


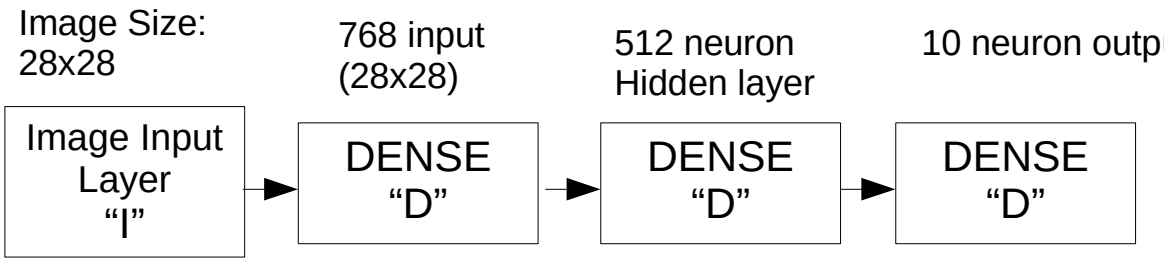
Illustration of the softmax block diagram from Milos Hauskrecht, milos@cs.pitt.edu,5329 Sennott Square

Dense Layer

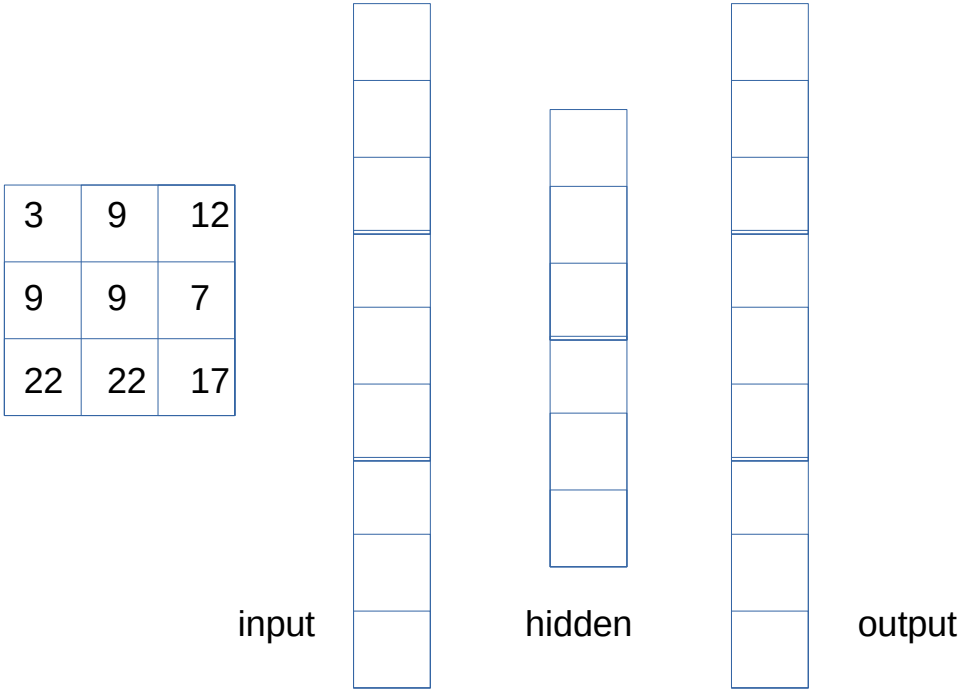


From: https://towardsdatascience.com/exploring-activation-functions-for-neural-networks-73498da59b02

Discussion with Feed Forward No Conv



From:
https://towardsdatascience.com/exploring-activation-functions-for-neural-networks-73498da59b02



Activation Functions

768 input
(28x28)

```
model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

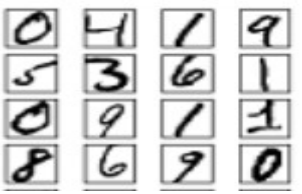
```
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

```
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

From: https://towardsdatascience.com/exploring-activation-functions-for-neural-networks-73498da59b02

Softmax

Example: NIST 10 digits (0-9) convnet Dense layer



Sample code : https://github.com/fchollet/deep-learning-with-python-notebooks

```
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'))
```

The network consists of 2 Dense layers, densely-connected ("fully-connected") neural layers. The output layer is a 10-way "softmax" layer, it returns an array of 10 probability scores (summing to 1). Each score will be the probability that the current digit image belongs to one of our 10 digit classes.

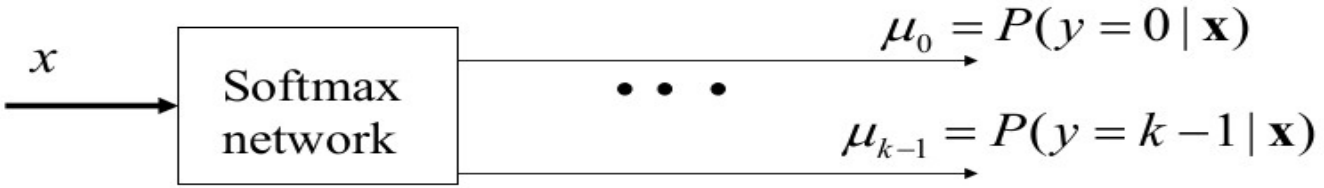


Illustration of the softmax block diagram from Milos Hauskrecht, milos@cs.pitt.edu,5329 Sennott Square

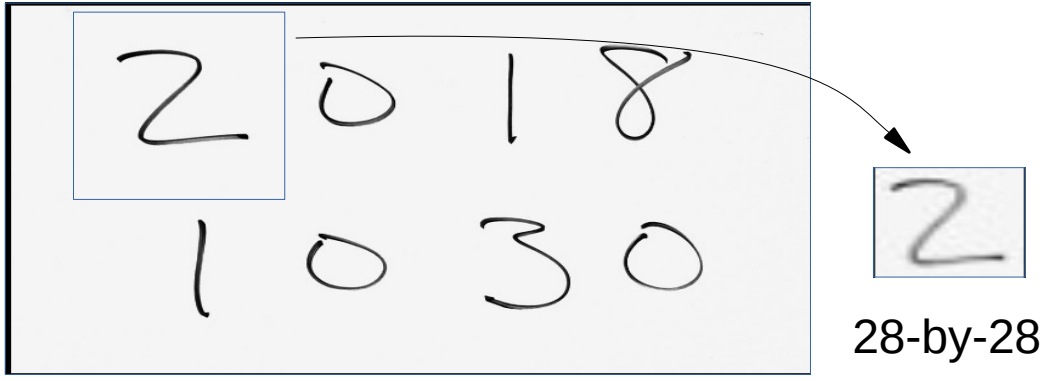
Prepare Image For Testing Trained MNIST convnet

Find out the input image size to MNIST convnet, from 2 sources: (1) from 7convnets-NumeralDetection-ch05.py (code below) (2) from the code, the dimension of the conv2d_1, kernel size is 3x3, from model.summary, the output shape is 26x26.

Example:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
```

So, (1) use screen capture program to locate ROI as shown below, and then (2) we wrote opencv program to take any size input image and convert it to 28-by-28



```
>>> model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

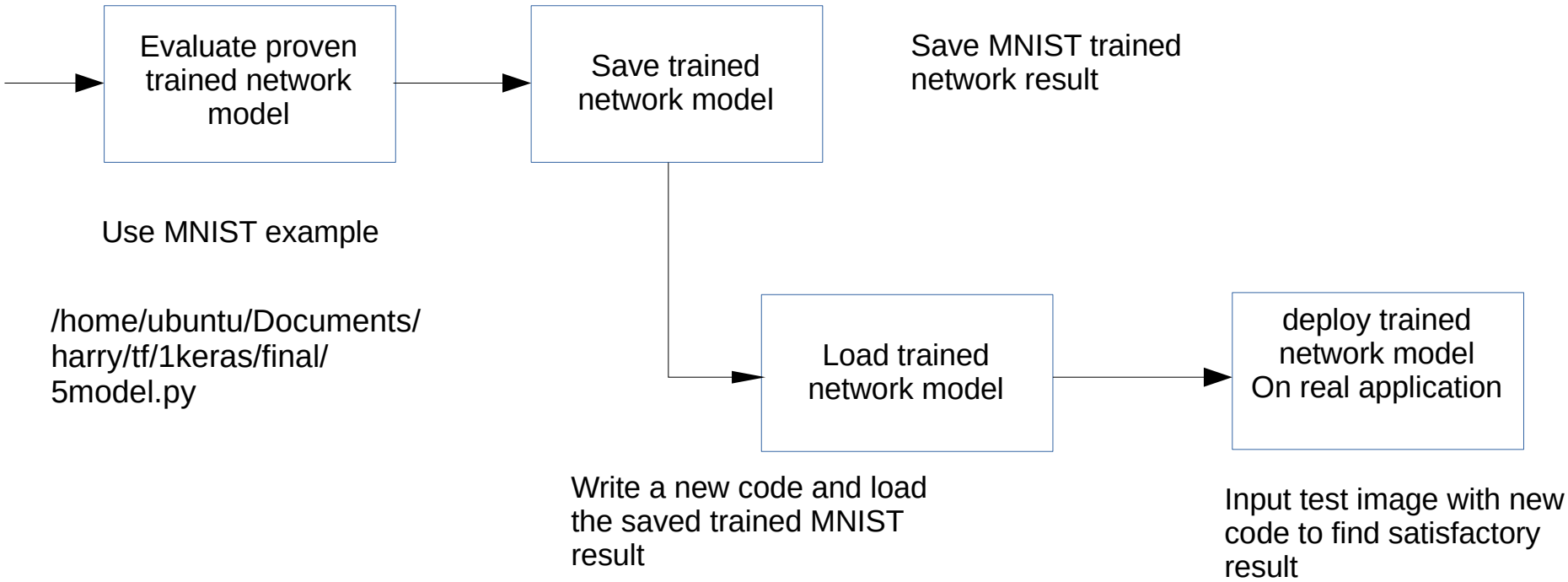
Convnet MNIST Architecture

Using TensorFlow backend.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_1 (MaxPooling2	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 64)	0
conv2d_3 (Conv2D)	(None, 3, 3, 64)	36928

Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0

10-29-2018 Road Map to Deploy CNN



10-29-2018 Save Keras Model

https://www.google.com/search?hl=en&ei=kFDZW__GM8SS0wLLhJSICg&q=save+model+json+keras&oq=save+Model+%28JSON%29&gs_l=psy-ab.1.0.0i22i30l7.4409.4409..7830...0.0..0.66.66.1.....0....1j2..gws-wiz.....0i71.fyoCDOfN8UU

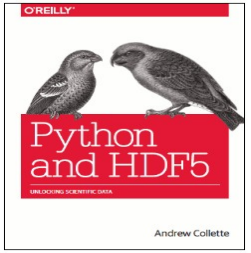
You can use model.save(filepath) to save a Keras model into a single HDF5 file which will contain: the architecture of the model, allowing to re-create the model. the weights of the model. the training configuration (loss, optimizer) the state of the optimizer, allowing to resume training exactly where you left off.

10-30-2018 Prepare h5py to Save Trained Network

https://www.quora.com/How-do-I-save-a-convolution-neural-network-model-after-training-I-am-working-in-Python

Keras for saving a model is just one line of code after training

Just make sure to have HDF5 for Python, use h5py package, which is a Pythonic interface to the HDF5 binary data format, which stores huge amounts of numerical data, and easily to be manipulated with NumPy. The files created can be exchanged including programs like IDL and MATLAB.



http://www.h5py.org/

First, install h5py, see the url link to the right and just follow the installation instructions, after the installation is done, be sure to use the following command to check:

```
import h5py
h5py.run_tests()
```

```
ubuntu@ubuntu-ThinkPad-Yoga-14:~/Documents$ python
Python 3.4.3 (default, Nov 28 2017, 16:41:13)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import h5py
>>> h5py.run_tests()
.....................................................x.................
........x.....................................s...s.....ss.............
.....................................................ssssss............
...........x....x.........................x......x.....................
.........
----------------------------------------------------------------------
Ran 457 tests in 0.632s

OK (skipped=14, expected failures=6)
<unittest.runner.TextTestResult run=457 errors=0 failures=0>
>>>
```

10-30-2018 Save Trained Network with h5py

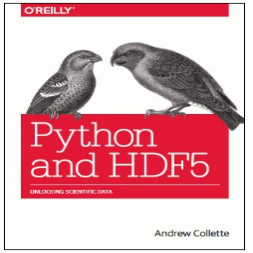
https://www.quora.com/How-do-I-save-a-convolution-neural-network-model-after-training-I-am-working-in-Python

Keras for saving a model is just one line of code after training

Model.save('harry_convnet_mnist.h5')

Example:

```
scores = model.evaluate(X_train,y_train,verbose=0)
print("Accuracy on train set: %.2f%%" % (scores[1]*100))
scores = model.evaluate(X_validation,y_validation,verbose=0)
print("Accuracy on validation set: %.2f%%" % (scores[1]*100))
scores = model.evaluate(X_test,y_test,verbose=0)
print("Accuracy on test set: %.2f%%" % (scores[1]*100))
model.save('ajits_model_d6_256.h5')
```



http://www.h5py.org/

Example: 7-1convnets-NumeralDet-saveTrained.py

```
#------------save trained model---------*
import h5py
model.save('harryTest.h5')
#-end
```

Just add 2 lines of code

10-30-2018 Load Trained Network

https://stackoverflow.com/questions/35074549/how-to-load-a-model-from-an-hdf5-file-in-keras

Keras for loading a model is just one line of code

If you stored the complete model, not only the weights, in the HDF5 file, then load is as simple as

Example:

```
#----------load trained model--------------------*
from keras.models import load_model
model = load_model('harryTest.h5')
model.summary() #check the model
```

```
>>> model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

OpenCV Program To Resize 28x28 Images

7-3ResizeImage.py

```
import cv2
import numpy as np

gray = cv2.imread('test.png',cv2.IMREAD_GRAYSCALE)
gray = cv2.resize(gray, (28,28))
cv2.imshow('image',gray)
cv2.imwrite('resized_harryTest.jpg',gray, [int(cv2.IMWRITE_JPEG_QUALITY),90])

k = cv2.waitKey(0)
if k == 27:         # wait for ESC key to exit
    cv2.destroyAllWindows()
```

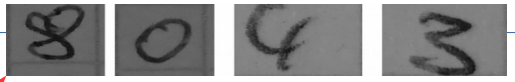
11-2-2018 Deploy MNIST with Test Image (1)

Example:

```
model = load_model('Numeral_detector.h5')
test_image= cv2.imread('resized_LCTest.jpg', cv2.IMREAD_GRAYSCALE)
test_image = np.reshape(test_image, [1, 28, 28, 1])
result_arr = model.predict(test_image)
result = model.predict_classes(test_image)
print(result_arr,' ', result)
```

10-29-2018 Deploy MNIST with Test Image (1)

https://medium.com/@o.kroeger/tensorflow-mnist-and-your-own-handwritten-digits-4d1cd32bbab4



8.png 0.png 4.png 3.png

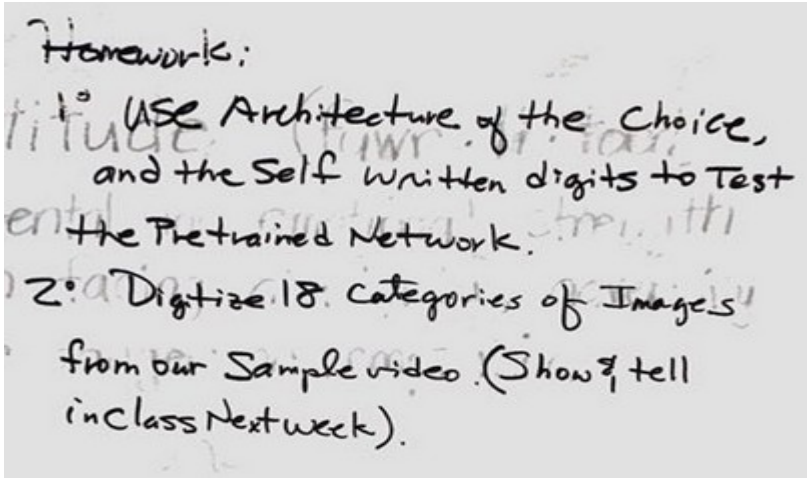
Image size: 28x28 = 784

```
# create an array where to store 4 pics with one numeral each
# each image consists of total 784 pixels
images = np.zeros((4,784))
# and the correct values
correct_vals = np.zeros((4,10))

# test images 8, 0, 4, 3
i = 0
for no in [8,0,4,3]:
    gray = cv2.imread("img/blog/own_"+str(no)+".png", cv2.CV_LOAD_IMAGE_GRAYSCALE)

    # resize the images and invert it (black background)
    gray = cv2.resize(255-gray, (28, 28))

    # save the processed images
    cv2.imwrite("pro-img/image_"+str(no)+".png", gray)
    """
    all images in the training set ranges from 0-1
    not from 0-255 so divide the flatten images
    (a one dimensional vector with 784 pixels)
    """
    flatten = gray.flatten() / 255.0
```

10-29-2018 Deploy MNIST with Test Image (2)

```
    """–---------------- store flatten image and generate----------------
    the correct_vals array
    correct_val for the first digit (9) would be
    [0,0,0,0,0,0,0,0,0,1]
       ----------------------------------------------------------------------"""
    images[i] = flatten
    correct_val = np.zeros((10))
    correct_val[no] = 1
    correct_vals[i] = correct_val
    i += 1
""" –-------the prediction will be an array with four values------------
    which show the predicted number
    –----------------------------------------------------------------------"""
prediction = tf.argmax(y,1)

"""–-------run the prediction and the accuracy function----------------
using our generated arrays (images and correct_vals)
    –----------------------------------------------------------------------"""
print sess.run(prediction, feed_dict={x: images, y_: correct_vals})
print sess.run(accuracy, feed_dict={x: images, y_: correct_vals})
```

11-2-2018 Architecture Aspects on Deployment



1
```
from keras import models
from keras import layers
from keras import optimizers
```

2
```
import matplotlib.pyplot as plt

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
                    plt.figure()
plt.show()
```

3
```
model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=4 * 4 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

4
```
model.compile(optimizer=optimizers.RMSprop(lr=2e-5),
              loss='binary_crossentropy',
              metrics=['acc'])
```

11-2-2018 Architecture Aspects on Deployment

1
```
from keras import models
from keras import layers
```

2
```
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
```

11-6-2018 Keras Supported CNN Architectures

https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

Xception
InceptionV3
ResNet50
VGG16
VGG19
MobileNet

ResNet architecture which can be successfully trained at depths of 50-200 for ImageNet and over 1,000 for CIFAR-10

Two major drawbacks with VGGNet: (1) painfully slow to train. (2) Network architecture weights themselves are quite large (in terms of disk/bandwidth). Due to its depth and number of fully-connected nodes, VGG16 is over 533MB and VGG19 is 574MB. This makes deploying VGG a tiresome task.

https://github.com/KaimingHe/resnet-1k-layers

11-6-2018 Use Keras Supported CNN

https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

```
# import the necessary packages
from keras.applications import ResNet50
from keras.applications import InceptionV3
from keras.applications import Xception # TensorFlow ONLY
from keras.applications import VGG16
from keras.applications import VGG19
from keras.applications import imagenet_utils
from keras.applications.inception_v3 import preprocess_input
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import numpy as np
import argparse
import cv2
```

Note: Weights for VGG16 and VGG19 are > 500MB. ResNet weights are ~100MB, while Inception and Xception weights are between 90-100MB. If this is the first time you are running this script for a given network, these weights will be (automatically) downloaded and cached to your local disk. Depending on your internet speed, this may take awhile. However, once the weights are downloaded, they will not need to be downloaded again, allowing subsequent runs of classify_image.py to be much faster.

11-6-2018 Deploy Cats Dogs Detection CNN

```
import time
import keras
keras.__version__
import numpy as np

from keras import models
import cv2
from keras.models import load_model
model = load_model('cats_and_dogs_small_1.h5')
#model.summary() #check the model

image =cv2.imread('1.jpg',cv2.IMWRITE_JPEG_QUALITY)
image = cv2.resize(image,(150,150))
image=np.reshape(image,[1,150,150,3])

time_curr=int(round(time.time()*1000))
output=model.predict_classes(image)
time_end=int(round(time.time()*1000))
time=time_end-time_curr
print("")
print(output)
print("")
print(time)
print("")
```

One image with resolution 150x150 with 3 channels (r, g, b) colors

11-9-2018 Python Imports for Augmentation To Create Train Data Set (1)

https://leemendelowitz.github.io/blog/how-does-python-find-packages.html

Python finds packages: Python imports by searching the directories listed in sys.path

```
>>> import sys
>>> print '\n'.join(sys.path)

/usr/lib/python2.7
/usr/lib/python2.7/plat-x86_64-linux-gnu
/usr/lib/python2.7/lib-tk
/usr/lib/python2.7/lib-old
/usr/lib/python2.7/lib-dynload
/usr/local/lib/python2.7/dist-packages
/usr/lib/python2.7/dist-packages
/usr/lib/python2.7/dist-packages/PILcompat
/usr/lib/python2.7/dist-packages/gtk-2.0
/usr/lib/python2.7/dist-packages/ubuntu-sso-client
>>>
```

sys.path.append() adds the path for the code to access.

10-26-2018 Keras API Functions

1
```
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
```

2
```
model.add(layers.MaxPooling2D((2, 2)))
```

3
```
model.add(layers.Flatten())
```

4
```
model.add(layers.Dense(64, activation='relu'))
```

5
```
tf.keras.layers.Dropout(0.2)(fc_1)
```

6
```
model.summary()
```

Oct-Nov-2018 Keras API Functions

1
```
import input_data
```

2
```
model.save('harryTest.h5')
```

3
```
from keras.models import load_model
```

4
```
model = load_model('harryTest.h5')
```

5
```
del model
```

6
```
import h5py
h5py.run_tests()
```

7
```
result_arr = model.predict(test_image)
```