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()
```

10-26-2018 ConvNet Architecture Design

Deep Learning Oct. 26 2018. 1/

Today's Topics:

1° Hands-On Programming ON Convnet.py.

Ref: github/hualili/opencv/IP120-AI-DL

7convnet.py. (Chollet's Style)

Compilation & Build this Program. Keras API Architect

Network Architecture → The Projects Assignment.

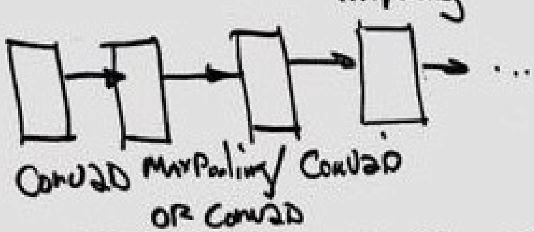
Objectives: Design And Prototype A DL Convnet.

Approach: (1) Network Diagram (from Examples) CAT I CAT II CAT III

Ref: CAT Video & Image 2018F-3-~ II & I

Basic Building Blocks (a) 2D Conv. Layer (c) Flatten (b) MaxPooling (d) Dense (layer)

maxpooling



Observation I: To Design Convnet for Deep Learning we start with Convolution Layer followed very often by MaxPooling Layer, then these Layers can be repeated...

Step 1. Repeating conv2D( ) layer and MaxPooling( ) layer

Example

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

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

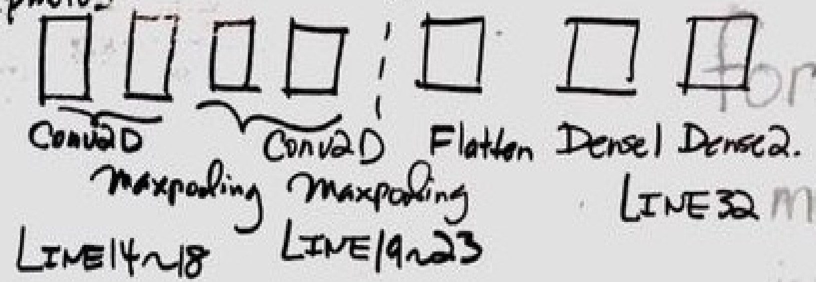
6
model.summary()

To display the network architecture

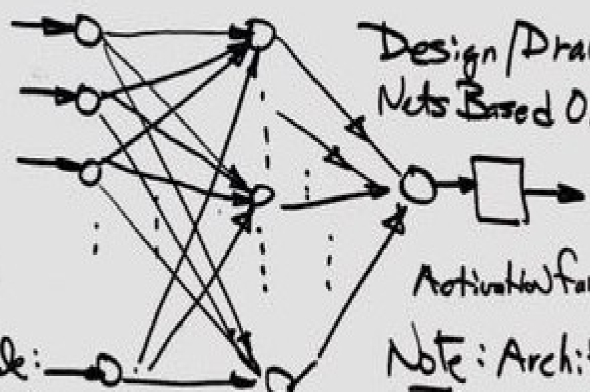
10-26-2018 3 ConvNet Examples

Example ①: from 5model.py Architecture

40k photos



Note:



Design/Draw Neural Nets Based On "Dense" layer.

Activation functions

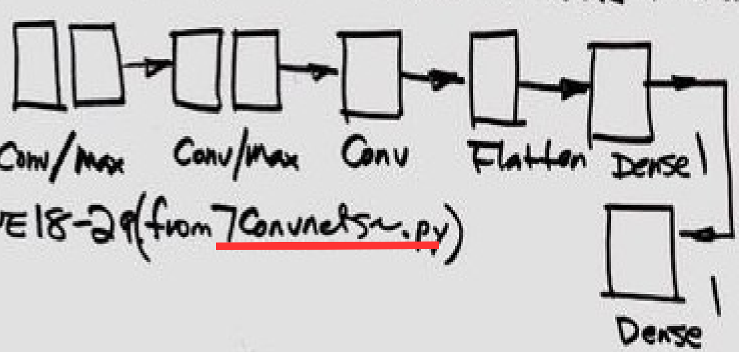
Example: ② NIST For Digits (0~9) Recognition.

60k photos

Note: Architecture

Conv/max → Conv/max →

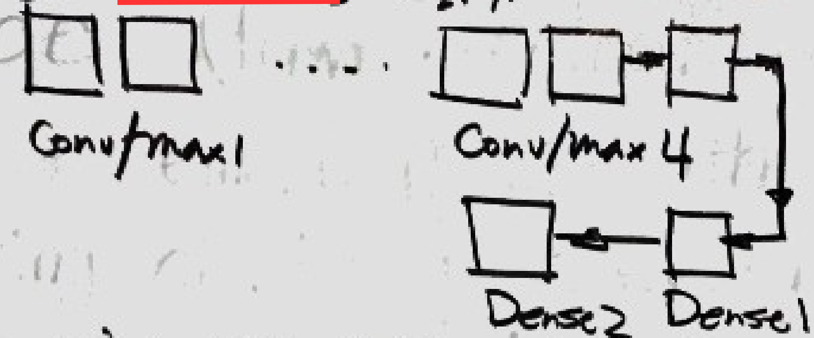
Dense ← Dense ← Flatten



LINE18-29 (from 7convnets~.py)

Example ③: For 1000 Cats & Dogs Photos.

Ref: 8convnets~.py. Flatten



Note: The tool to verify (Print) model is model.summary( )

Jerry On Python File Manipulations.

Sample Code: 8convnet~.py file 'manipulation'.

Observation II: Generally, Deep learning Architecture can be classified as convolutional Layer(s) and Neural Net layers.

Example: from the Summary of Example1 to Example3.

10-26-2018 Comparison of 3 ConvNets

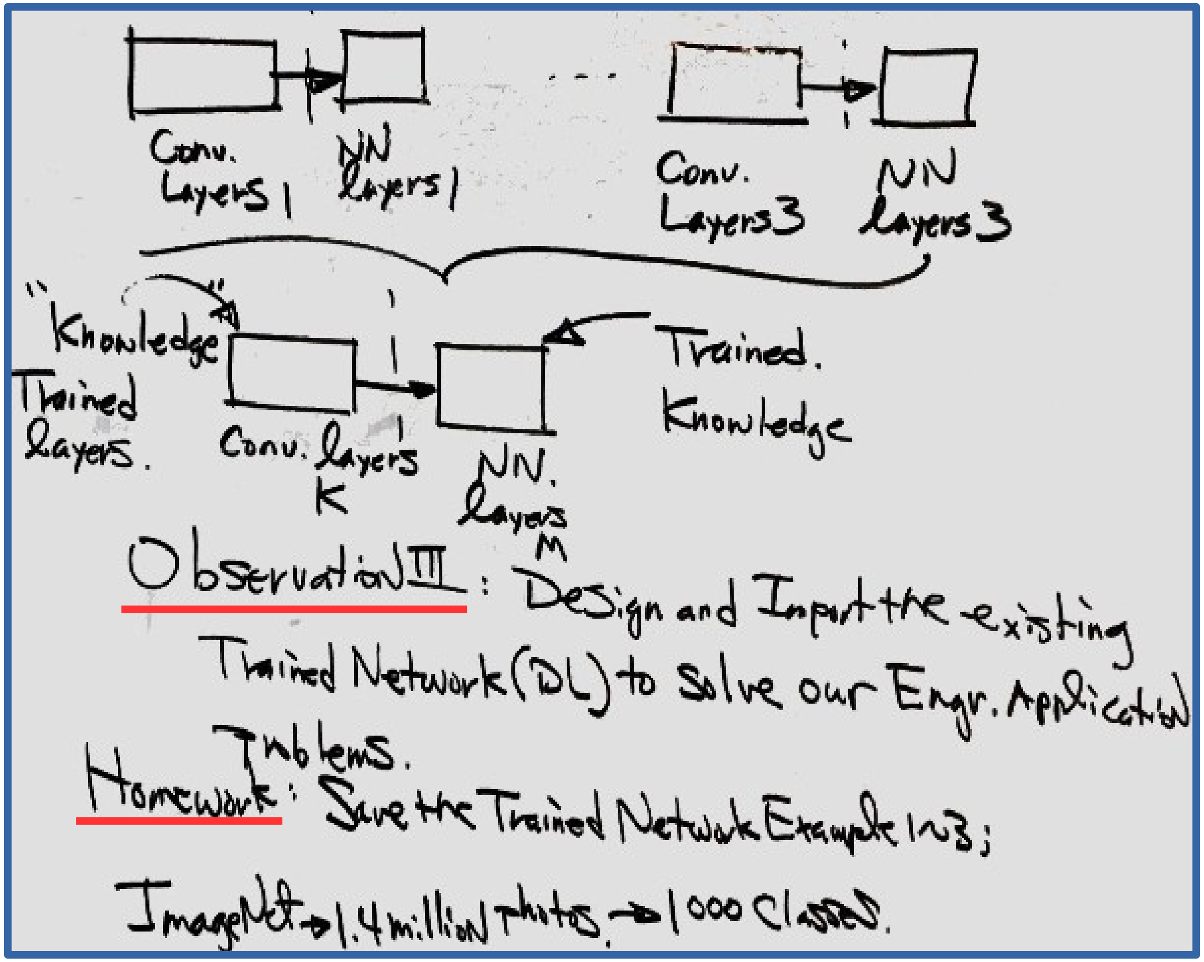
Table 1. Three well trained convnet examples

1 convnet1	Satellite imagery	40K images	ConvPool1+ConvPool2+Flatten+Den1+Den2
2 convnet2	NIST digits 0-9	60K images	ConvPool1+ConvPool2+Conv+Flatten+Den1+Den2
3 convnet3	Chollet cat-dog	1K images	ConvPool1+ConvPool2+ConvPool3 +ConvPool4 +Flatten+Den1+Den2

Table 2. Sample code for 3 well trained convnets

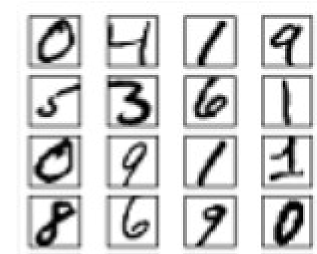
Network	Programs
convnet1	5model.py
convnet2	7convnets-NumeralDetection-ch05.py
convnet3	8convnets-SmallData-cats-dogs-ch05.py

10-26-2018 3 ConvNet Examples



10-26-2018 Numerals Output Design

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



```
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'))
```

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

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.

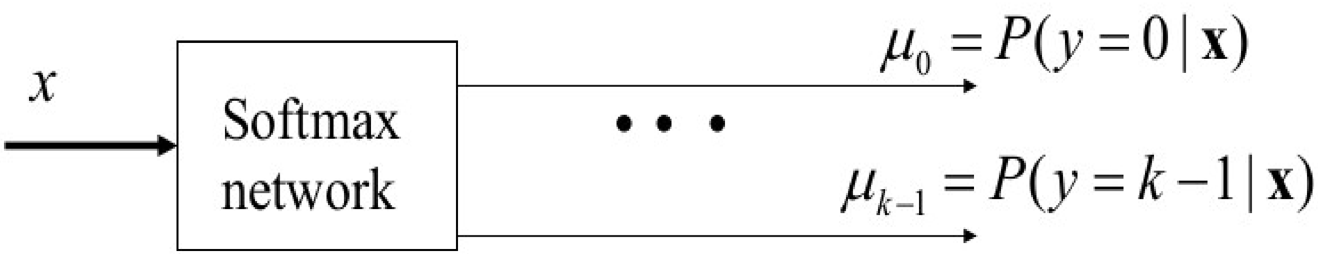


Illustration of the softmax block diagram from Milos Hauskrecht, milos@cs.pitt.edu,5329 Sennott Square

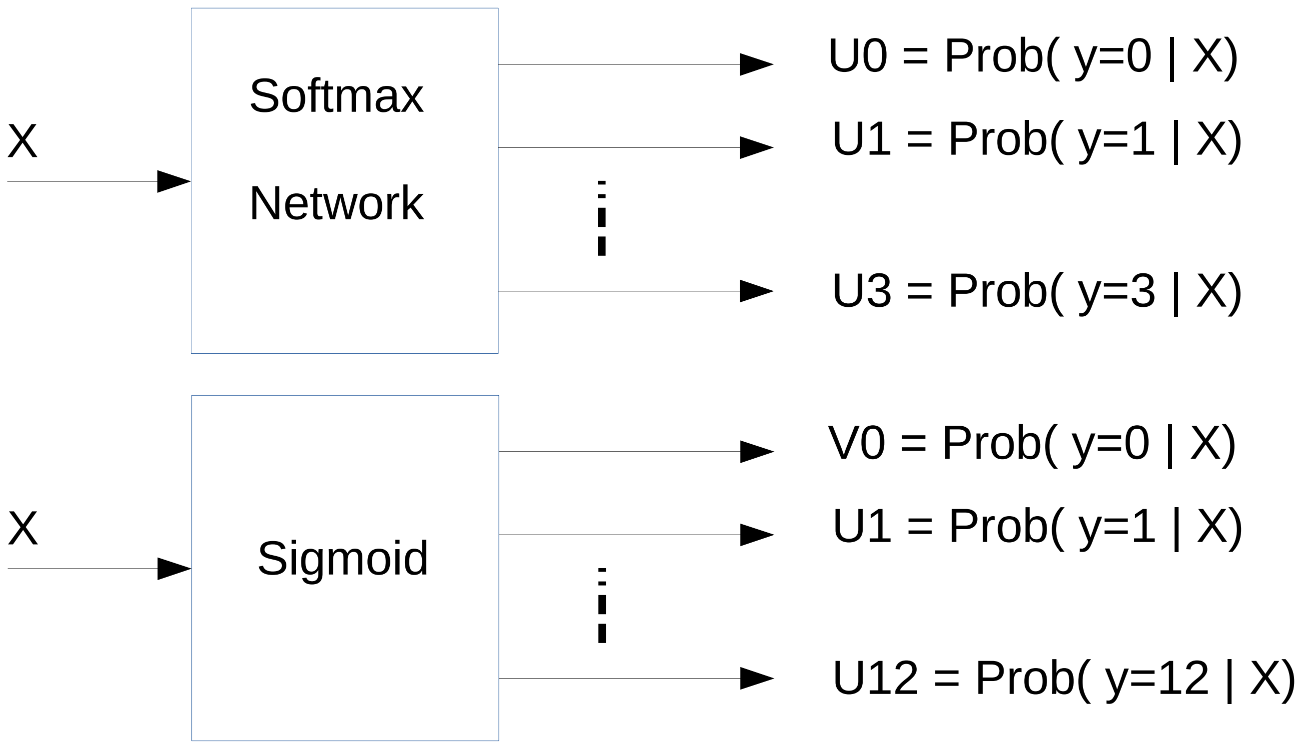
10-26-2018 Satellite Output Design

Example: Satellite Imagery Classification by convnet Dense layer

```
#separate outputs for the weather and the ground labels
weather_output = tf.keras.layers.Dense(4,
                                       activation='softmax',
                                       name='weather')(fc_2)
ground_output = tf.keras.layers.Dense(13,
                                      activation='sigmoid',
                                      name='ground')(fc_2)
```

Sample program: 5model.py

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

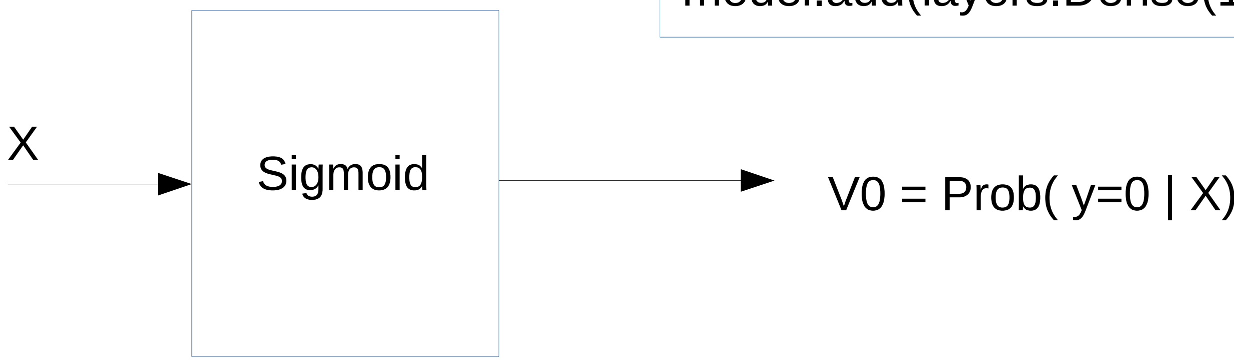


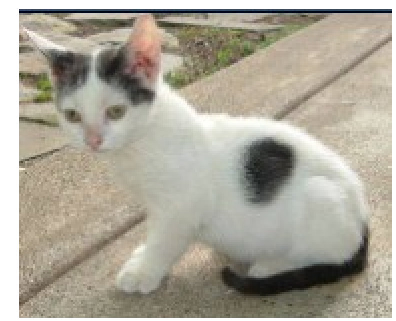
10-26-2018 Cats-Dogs Output Design

Example: Cats-dogs Imagery Classification by convnet Dense layer

Sample program: 8convnets-SmallData-cats-dogs-ch05

```
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```





10-26-2018 Kaggle Data and NIST Data

Example: Satellite Imagery Classification by convnet Dense layer

```
class KagglePlanetSequence(tf.keras.utils.Sequence):
```

Sample program: 5model.py

```
# original_dataset_dir = '/Users/fchollet/Downloads/kaggle_original_data'
original_dataset_dir = 'kaggle-cats-dogs'

# The directory to store smaller dataset
base_dir = 'kaggle-cats-dogs-small'
os.mkdir(base_dir)
```

Sample program: 8convnets-SmallData-cats-dogs-ch05

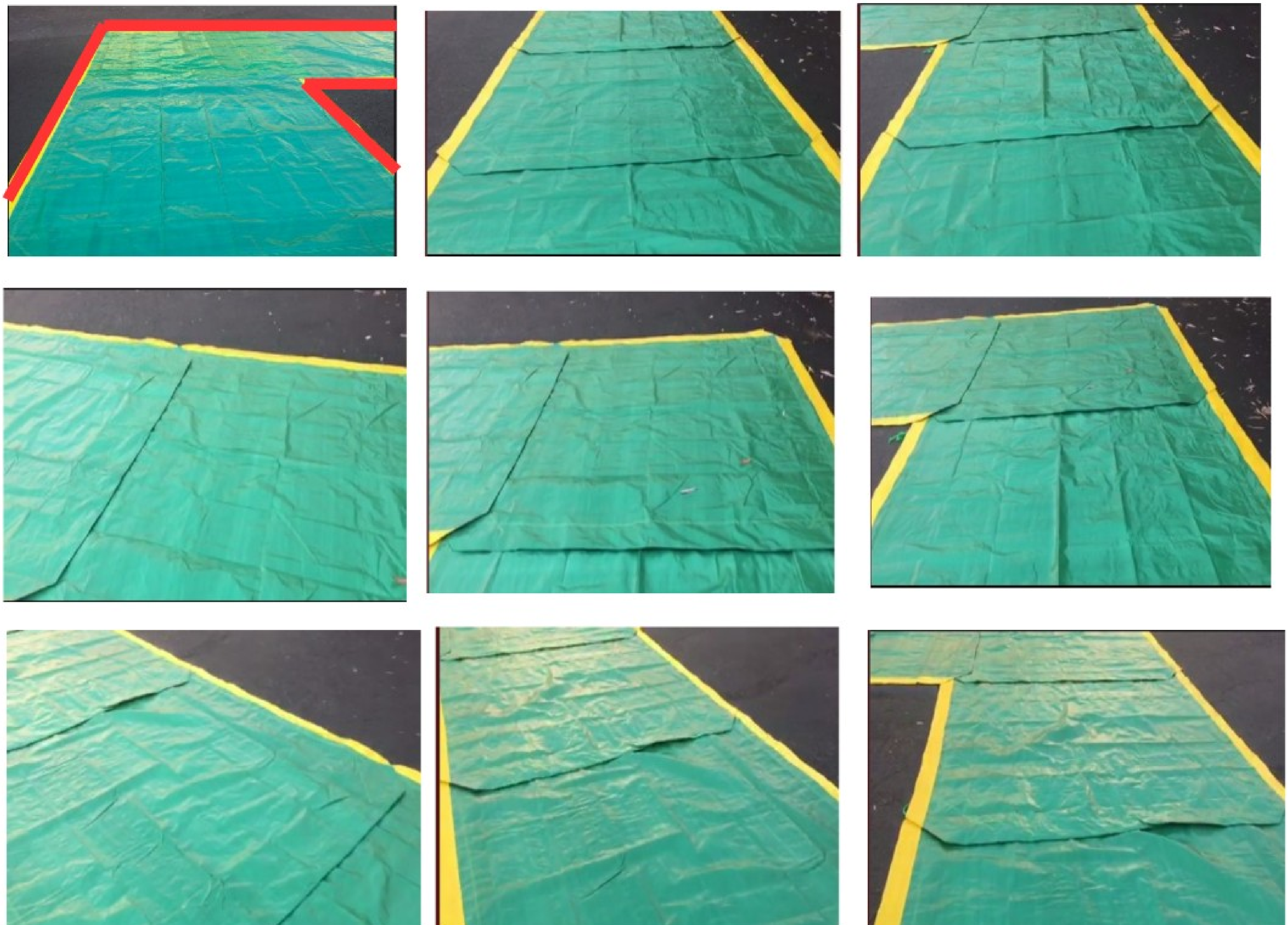
10-26-2018 Satellite Input Data Design

Example: Satellite Imagery Classification by convnet Dense layer

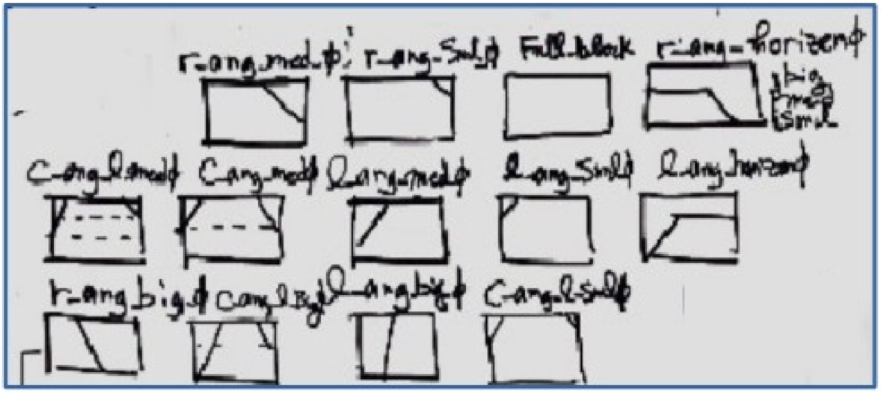
Sample program: 5model.py

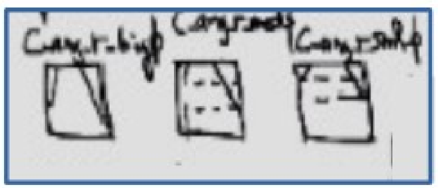
```
import tensorflow as tf
IM_SIZE = 128

image_input = tf.keras.Input(shape=(IM_SIZE, IM_SIZE, 3), name='input_layer')

# Some convolutional layers
conv_1 = tf.keras.layers.Conv2D(32,
                                kernel_size=(3, 3),
                                padding='same',
                                activation='relu')(image_input)
conv_1 = tf.keras.layers.MaxPooling2D(padding='same')(conv_1)
```

CAT-II Path Primitives



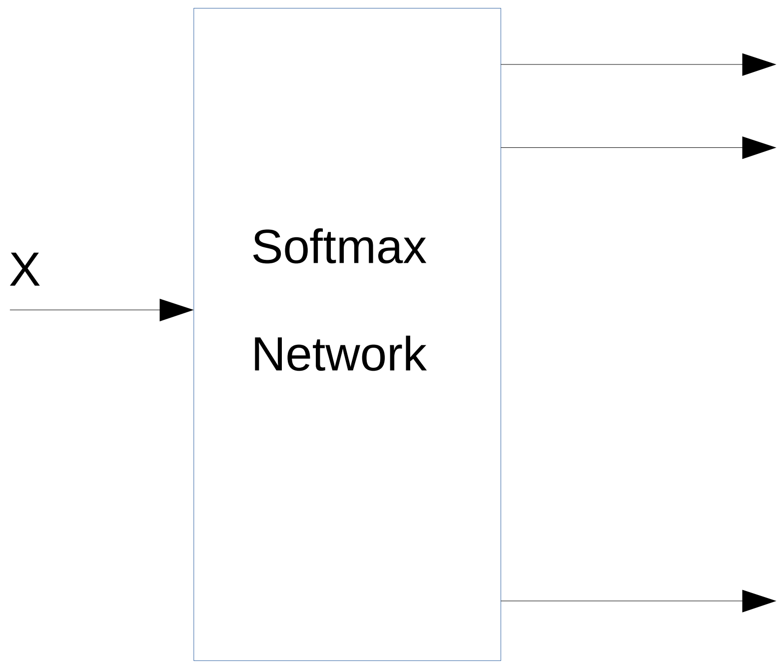
9-25-2018 Primitive Features for CAT-II Path Classification





1. r_ang_big	right angle big
2. r_ang_med	right angle medium
3. r_ang_sml	right angle small
4. r_ang_hor	right angle horizon
5. l_ang_big	left angle big
6. l_ang_med	left angle medium
7. l_ang_sml	left angle small
8. l_ang_hor	left angle horizon
9. cr_ang_big	centeral-right big
10. cr_ang_med	central angle medium
11. cr_ang_sml	central angle small
12. cl_ang_big	centeral-right angle
13. cl_ang_med	central angle medium
14. cl_ang_sml	central angle small
15. c_ang_big	centeral-angle big
16. c_ang_med	central-ang medium
17. c_ang_sml	central-ang small
18. full	full block

10-26-2018 Softmax Output Design for CATIII



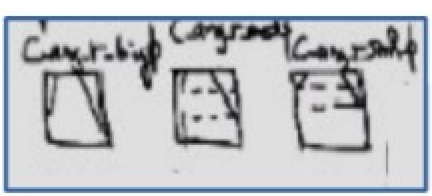
$U0 = Prob(y=0 | X)$

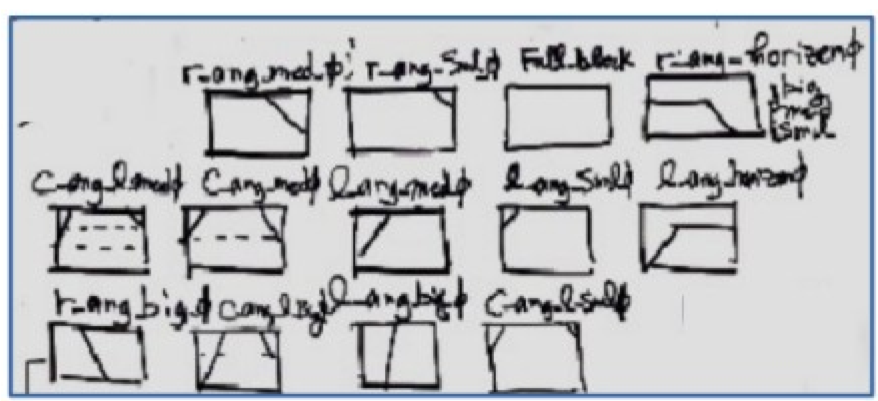
$U1 = Prob(y=1 | X)$

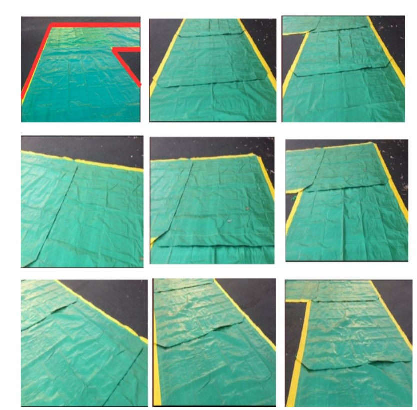
$U17 = Prob(y=17 | X)$

Where $Prob(y=0 | X) + Prob(y=1 | X) + \ldots + Prob(y=17 | X) = 1$

1. r_ang_big	right angle big
2. r_ang_med	right angle medium
3. r_ang_sml	right angle small
4. r_ang_hor	right angle horizon
5. l_ang_big	left angle big
6. l_ang_med	left angle medium
7. l_ang_sml	left angle small
8. l_ang_hor	left angle horizon
9. cr_ang_big	centeral-right big
10. cr_ang_med	central angle medium
11. cr_ang_sml	central angle small
12. cl_ang_big	centeral-right angle
13. cl_ang_med	central angle medium
14. cl_ang_sml	central angle small
15. c_ang_big	centeral-angle big
16. c_ang_med	central-ang medium
17. c_ang_sml	central-ang small
18. full	full block







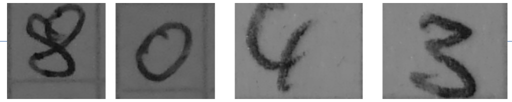
Harry Li, Ph.D. 2018

10-29-2018 Keras API Functions

1
import input_data

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

```
# 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})
```