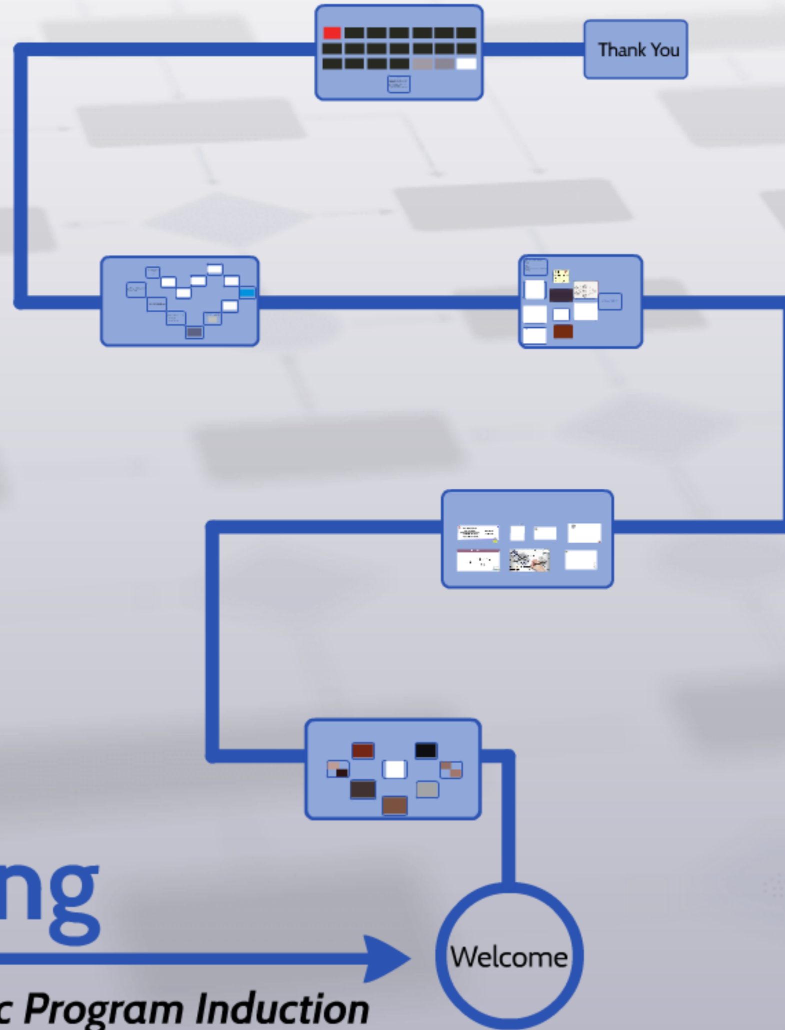
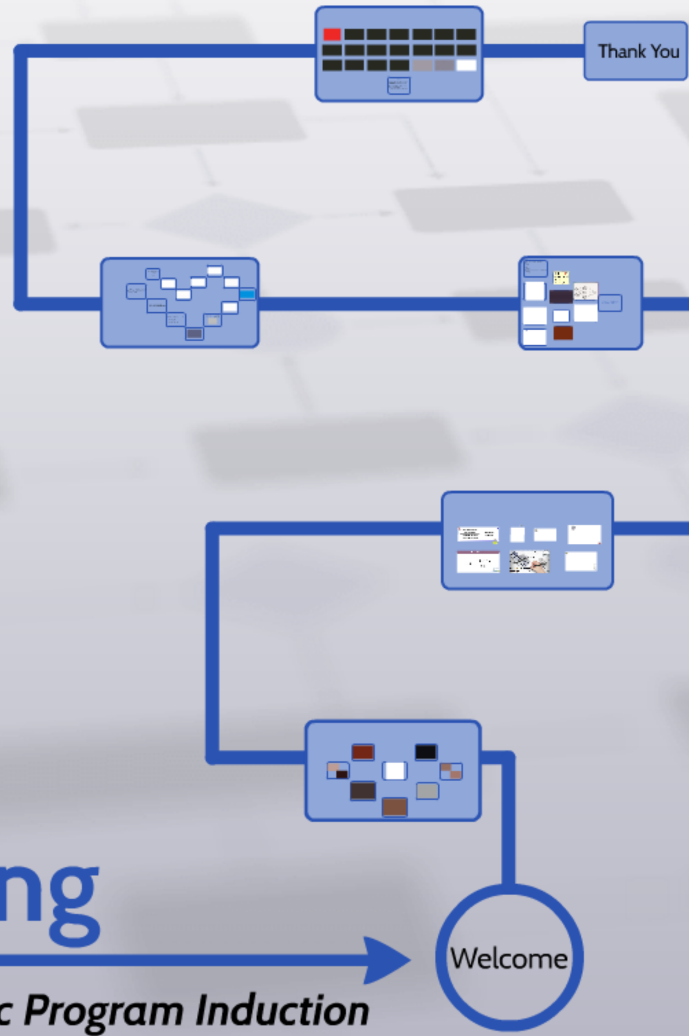
One Shot Learning

Human Level Concept Learning through Probabilistic Program Induction

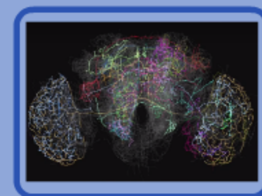


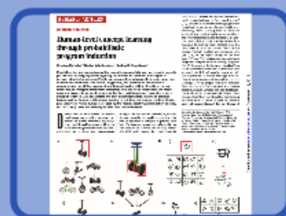
One Shot Learning

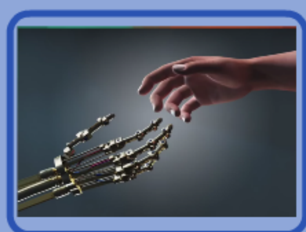
Human Level Concept Learning through Probabilistic Program Induction

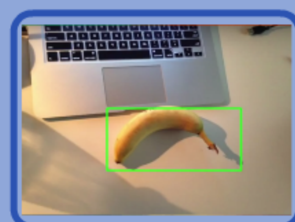


Welcome

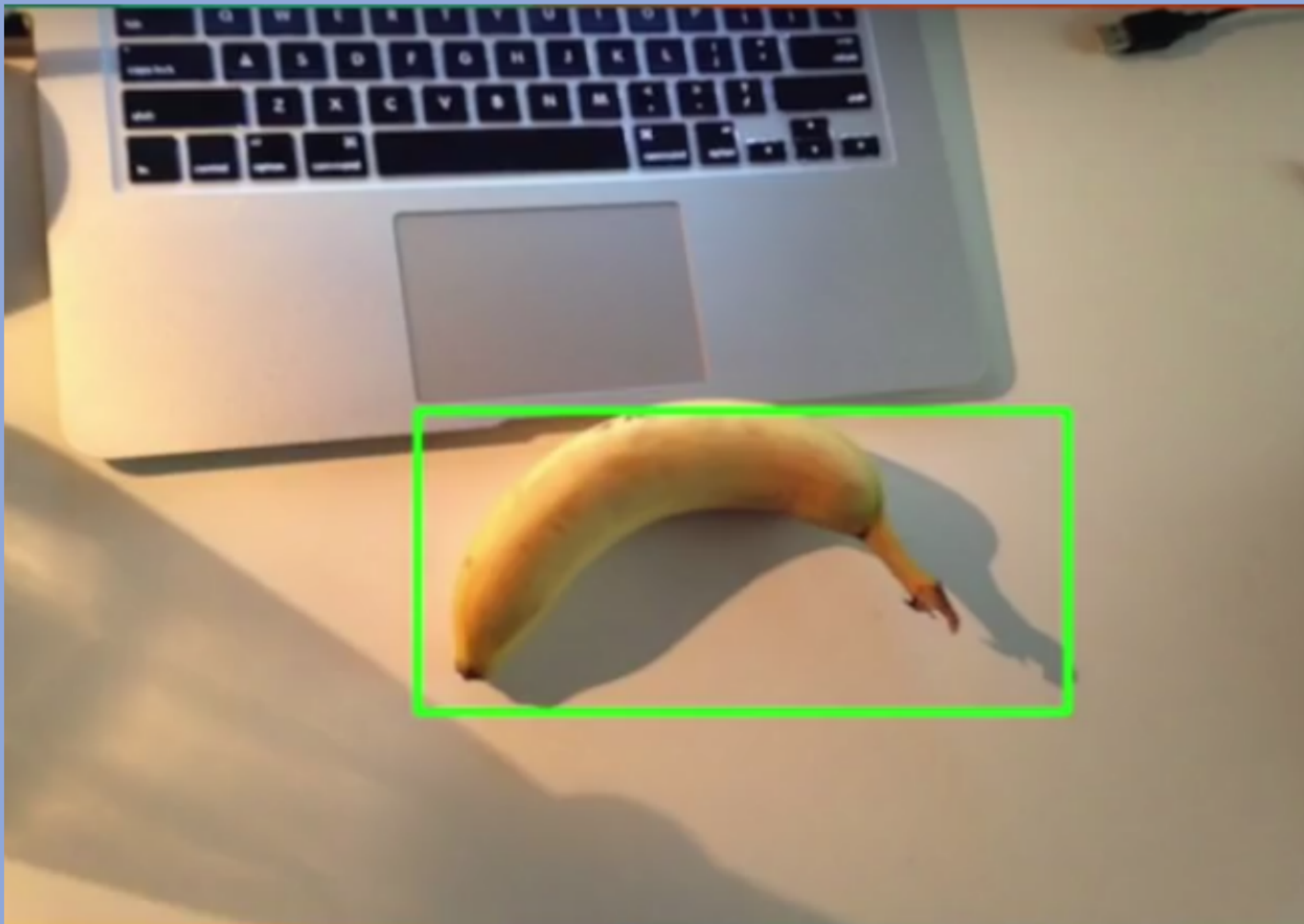







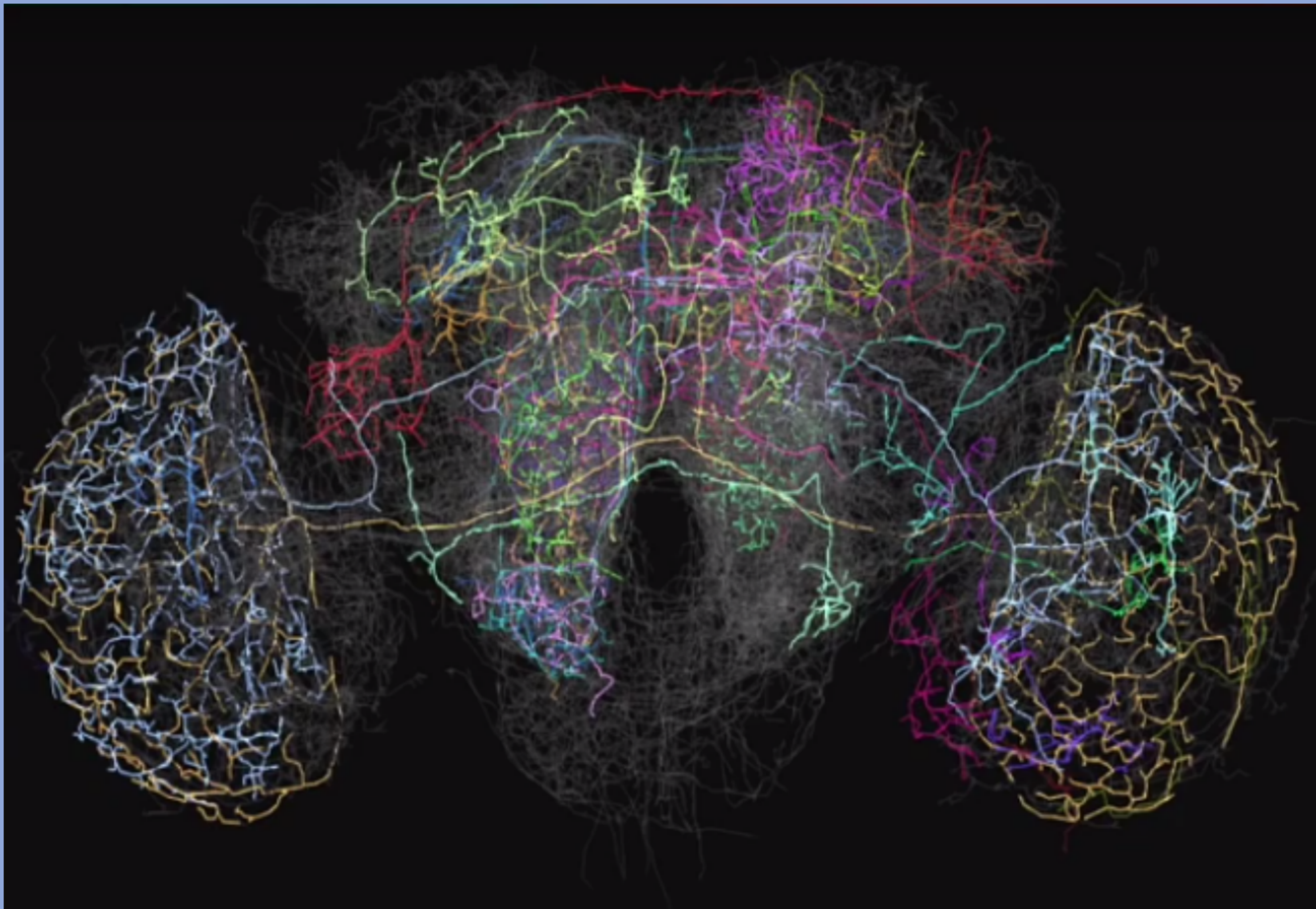
SCREW BATMAN
MEET BANANA MAN



abc
perceptron




SCREW BATMAN
MEET BANANA MAN


Posted at
BAJIROO.COM



RESEARCH ARTICLES

COGNITIVE SCIENCE

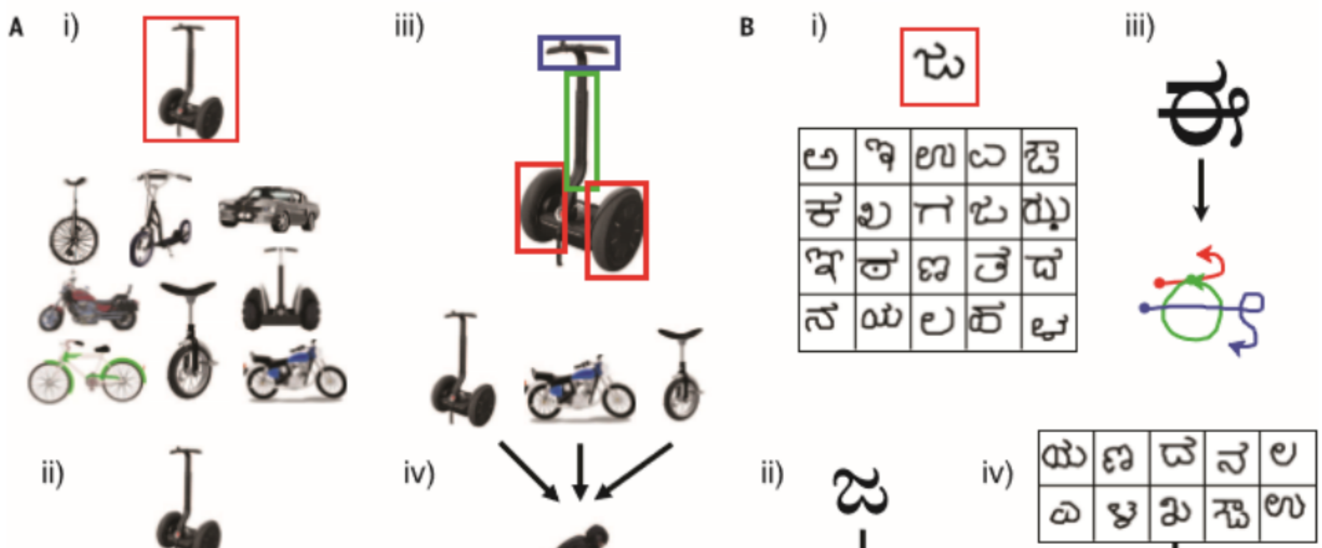
Human-level concept learning through probabilistic program induction

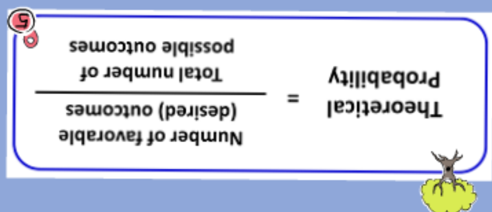
Brenden M. Lake,[1*] Ruslan Salakhutdinov,[2] Joshua B. Tenenbaum[3]

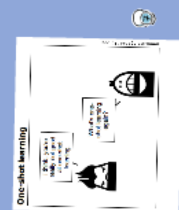
People learning new concepts can often generalize successfully from just a single example, yet machine learning algorithms typically require tens or hundreds of examples to perform with similar accuracy. People can also use learned concepts in richer ways than conventional algorithms—for action, imagination, and explanation. We present a computational model that captures these human learning abilities for a large class of simple visual concepts: handwritten characters from the world's alphabets. The model represents concepts as simple programs that best explain observed examples under a Bayesian criterion. On a challenging one-shot classification task, the model achieves human-level performance while outperforming recent deep learning approaches. We also present several "visual Turing tests" probing the model's creative generalization abilities, which in many cases are indistinguishable from human behavior.

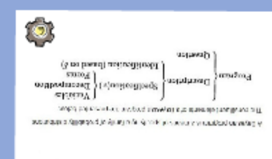
Despite remarkable advances in artificial intelligence and machine learning, two aspects of human conceptual knowledge have eluded machine systems. First, for most interesting kinds of natural and man-made categories, people can learn a new concept from just one or a handful of examples, whereas standard algorithms in machine learning require tens or hundreds of examples to perform similarly. For instance, people may only need to see one example of a novel two-wheeled vehicle (Fig. 1A) in order to grasp the boundaries of the new concept, and even children can make meaningful generalizations via "one-shot learning" (*1–3*). In contrast, many of the leading approaches in machine learning are also the most data-hungry, especially "deep learning" models that have achieved new levels of performance on object and speech recognition benchmarks (*4–9*). Second, people learn richer representations than machines do, even for simple concepts (Fig. 1B), using them for a wider range of functions, including (Fig. 1, ii) creating new exemplars (*10*), (Fig. 1, iii) parsing objects into parts and relations (*11*), and (Fig. 1, iv) creating new abstract categories of objects based on existing categories (*12, 13*). In contrast, the best machine classifiers do not perform these additional functions, which are rarely studied and usually require specialized algorithms. A central challenge is to explain these two aspects of human-level concept learning: How do people learn new concepts from just one or a few examples? And how do people learn such abstract, rich, and flexible representations? An even greater challenge arises when putting them together: How can learning succeed from such sparse data yet also produce such rich representations? For any theory of

[1]Center for Data Science, New York University, 726 Broadway, New York, NY 10003, USA. [2]Department of Computer Science and Department of Statistics, University of Toronto, 6 King's College Road, Toronto, ON M5S 3G4, Canada. [3]Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA.
*Corresponding author. E-mail: brenden@nyu.edu

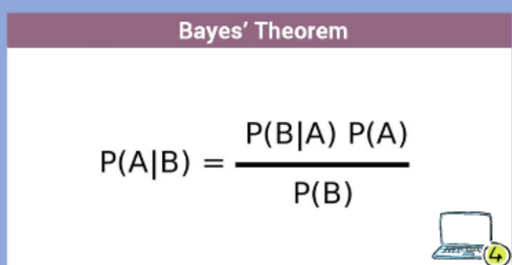



Theoretical Probability =
Number of favorable (desired) outcomes
Total number of possible outcomes





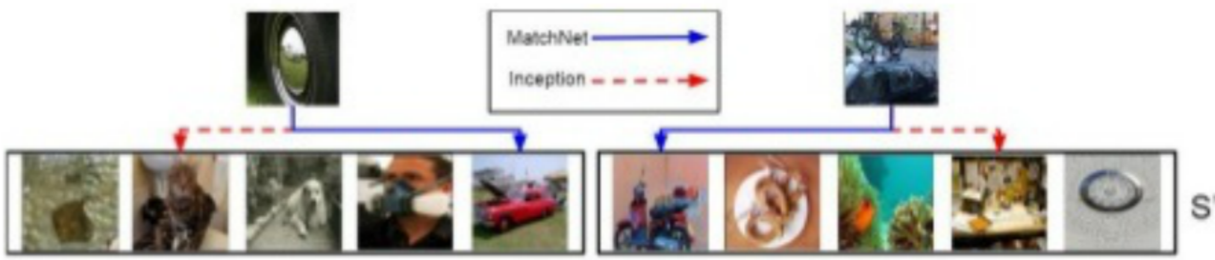

One-Shot Learning
How can we learn a novel concept from a few examples? (random guess will be 1/N)


Bayes' Theorem
$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$


*Machines learn like humans
Bayesian Program Learning


One-shot learning with attention and memory
Learn a concept from one or only a few training examples
Train a fully end-to-end nearest neighbor classifier: incorporating the best characteristics from both parametric and non-parametric models
Improved one-shot accuracy on Omniglot from 88.0% to 93.2% compared to competing approaches

One-Shot Learning



How can we learn a novel concept from a few examples? (random guess will be 1/N)

One-shot learning with attention and memory

- Learn a concept from one or only a few training examples
- Train a fully end-to-end nearest neighbor classifier: incorporating the best characteristics from both parametric and non-parametric models
- Improved one-shot accuracy on Omniglot from 88.0% to 93.2% compared to competing approaches

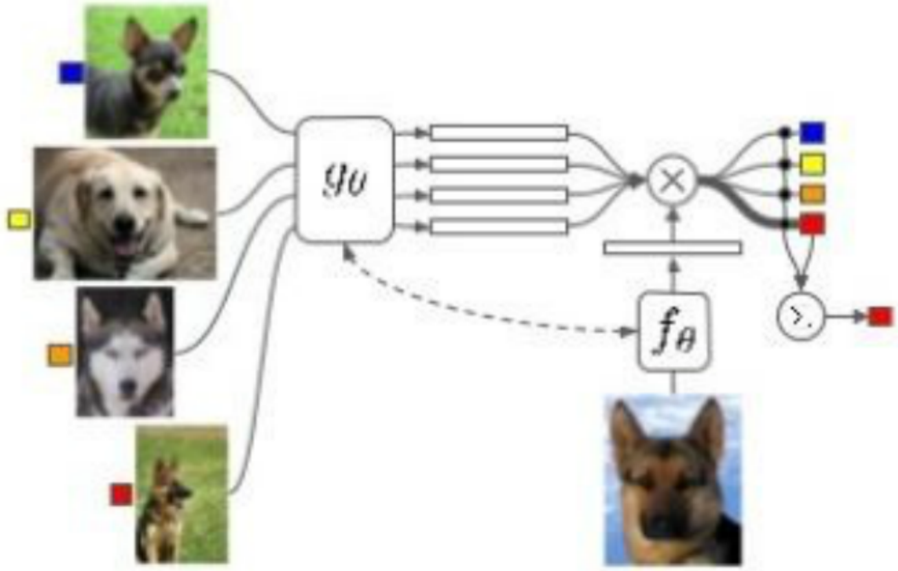


Figure 1: Matching Networks architecture

2


*Machines learn like humans
Bayesian Program
Learning
3

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

$$\text{Theoretical Probability} = \frac{\text{Number of favorable (desired) outcomes}}{\text{Total number of possible outcomes}}$$

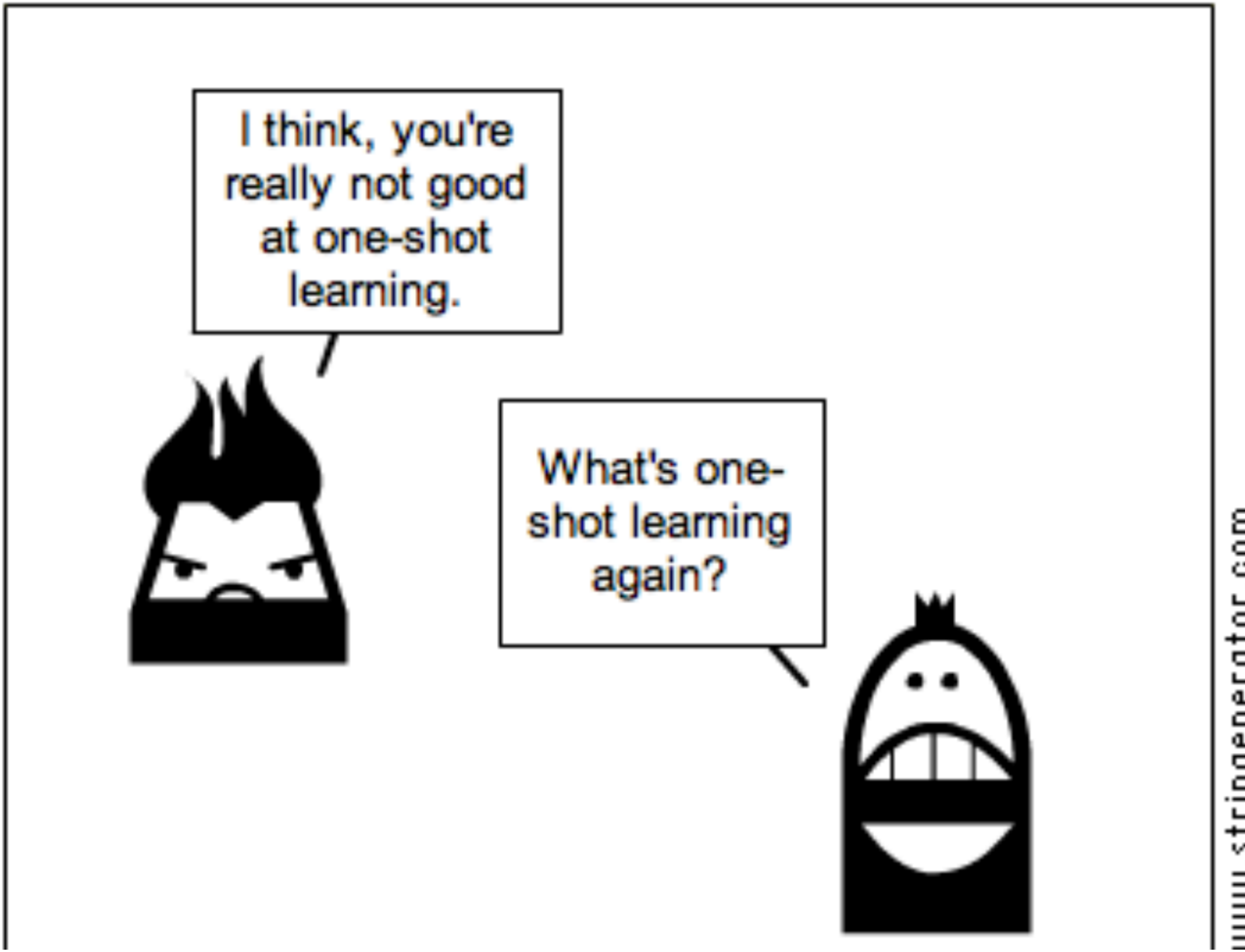
5

esian program is a means of specifying a family of probability distrib

onstituent elements of a Bayesian program are presented below:

$$\text{ogram}\begin{cases}\text{Description}\begin{cases}\text{Specification}(\pi)\begin{cases}\text{Variables}\\ \text{Decomposi}\\ \text{Forms}\end{cases}\\ \text{Identification (based on }\delta)\end{cases}\\ \text{Question}\end{cases}$$

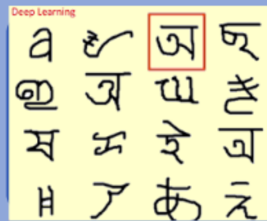
One-shot learning

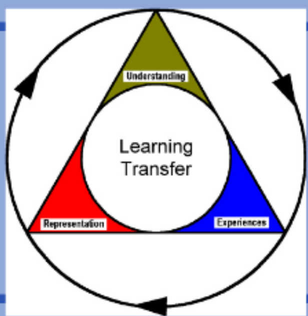


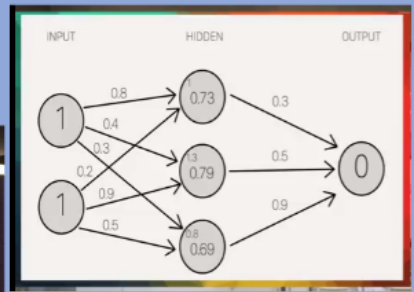
One-shot Learning with Memory-Augmented Neural Networks

Authors Name

1. **Adam Santoro**
   Google DeepMind
2. **Sergey Bartunov**
   Google DeepMind, National Research University Higher School of Economics (HSE)
3. **Matthew Botvinick**
4. **Daan Wierstra**
5. **Timothy Lillicrap**
   Google DeepMind

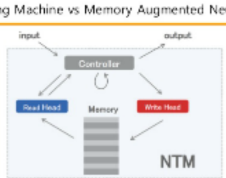




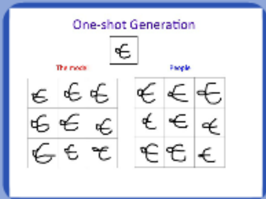


One-shot Learning with Memory-Augmented Neural Networks

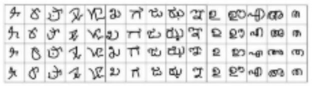
Neural Turing Machine vs Memory Augmented Neural Network



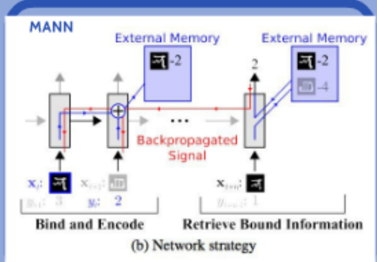
1. MANN은 NTM의 변형이다.
2. Controller는 Feed Forward NN or LSTM을 사용하였다.
3. 기존 NTM은 복사, 정렬등의 알고리즘을 학습하는데 사용하였지만 이 논문에서는 One shot Learing에 사용
4. 빠른 학습을 위해 Memory에 Write할 때 가장 적게 사용된 메모리 or 가장 최근에 사용된 메모리에 Write함 (LRUA)
5. 그렇다면 기본적인 질문? 왜 Augmented Memory가 필요한가?



Experiment – Data

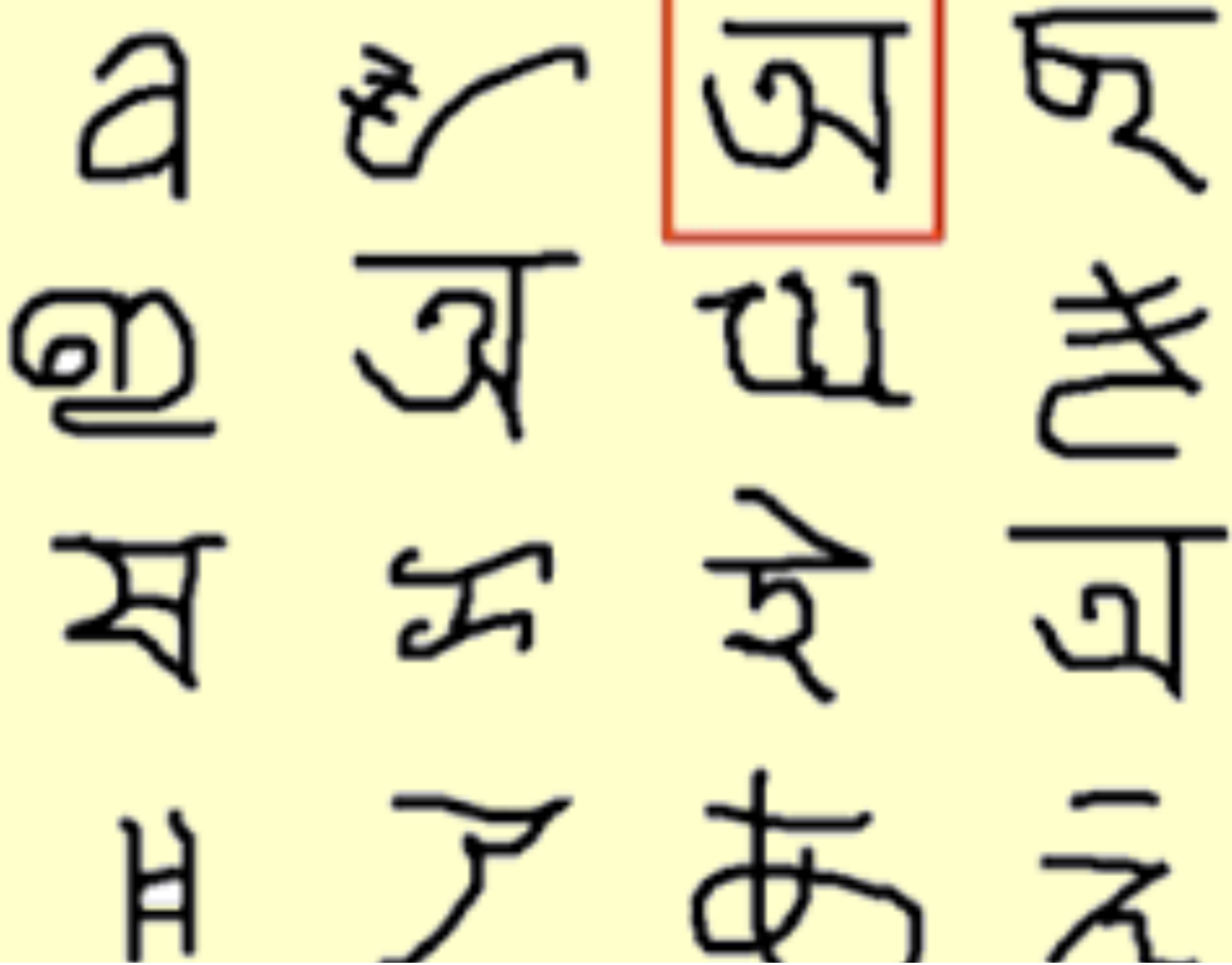


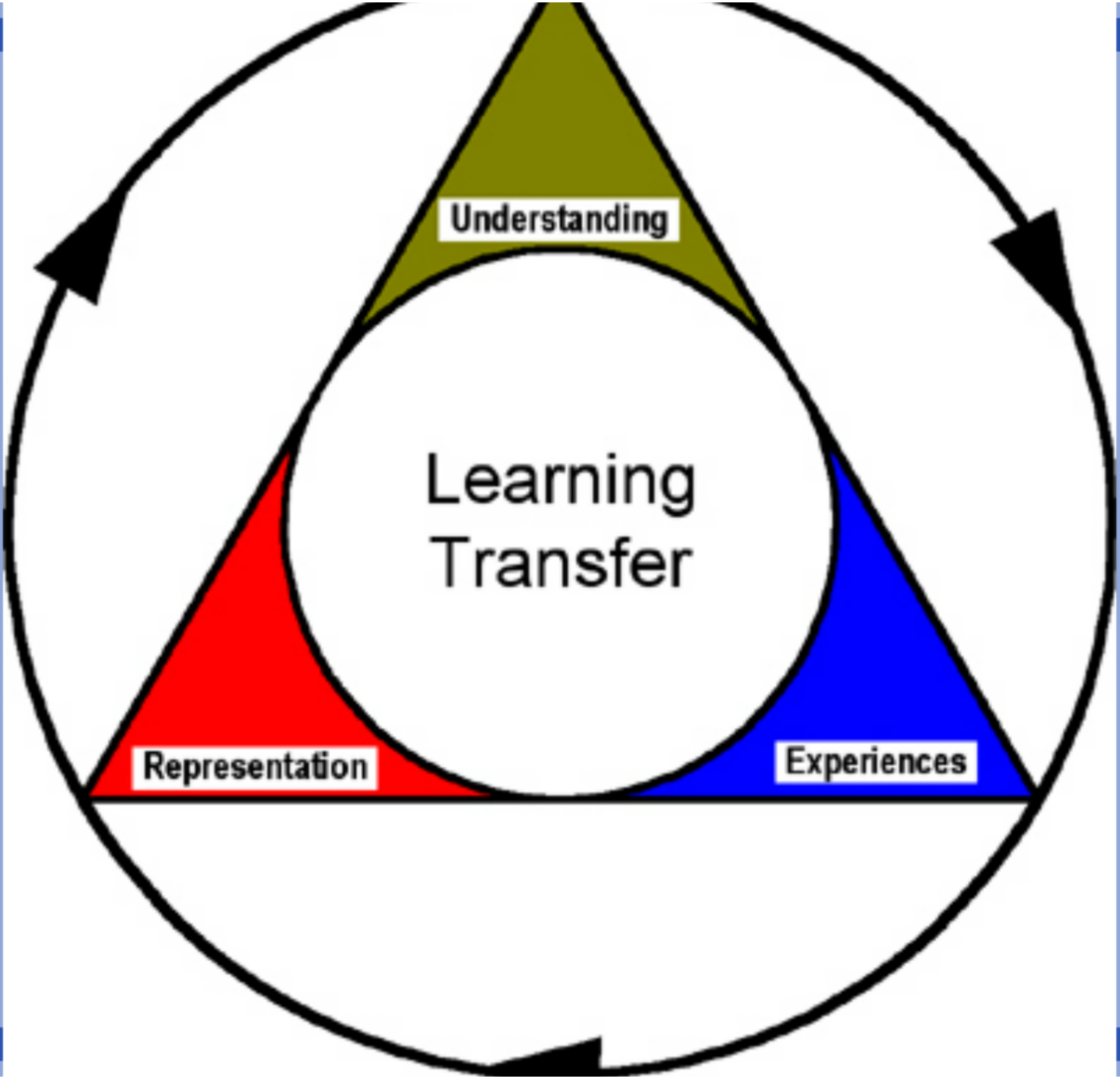
Omniglot Dataset : 1600 > classes
=> 1200 class train, 423 class test (downscale to 20x20)
+ plus rotate augmentation



One-shot Learning with Memory-Augmented Neural Networks

Deep Learning




Understanding
Learning
Transfer
Representation
Experiences


Bloomberg
Our view
The 'Existential Risk' framing has become
from the
core ethics & safety issues, and it overshadows the debate
AI is a hugely powerful
we control
There are
Google
Suleyman

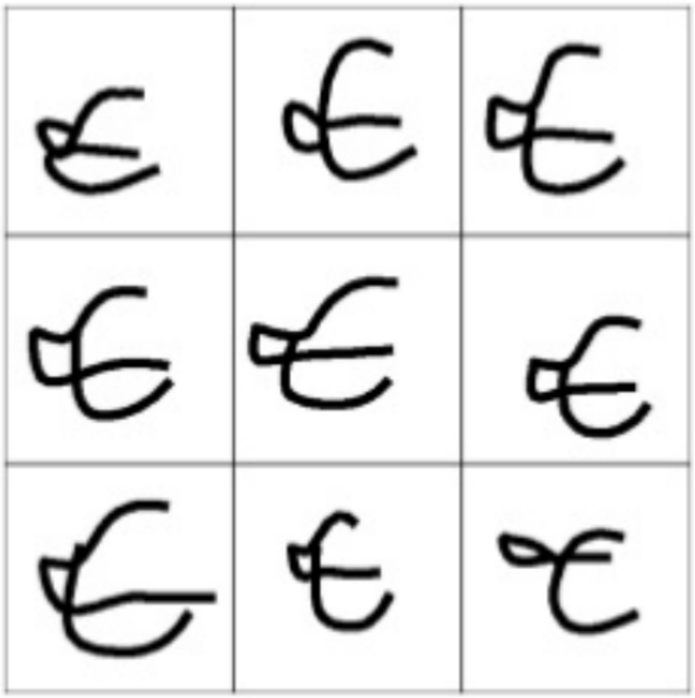
One-shot Learning with Memory-Augmented Neural Networks

Authors Name

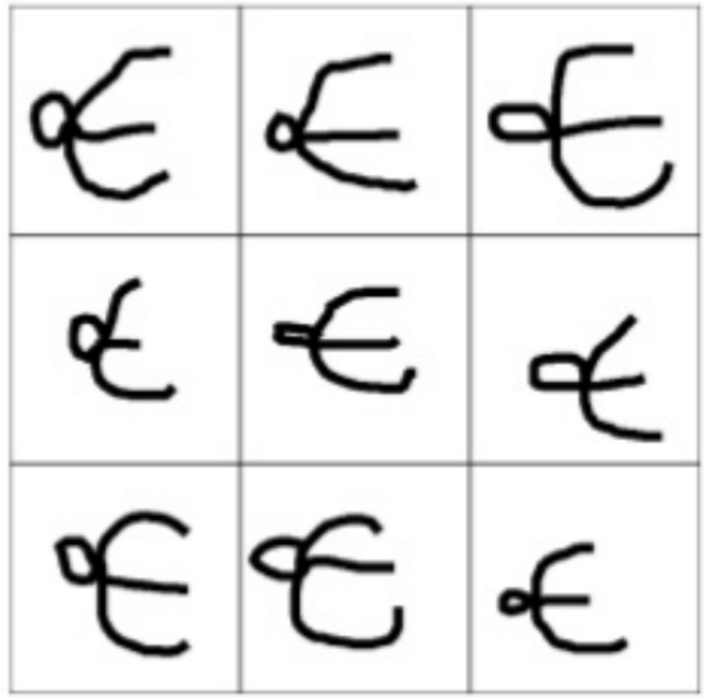
1. **Adam Santoro**
   Google DeepMind
2. **Sergey Bartunov**
   Google DeepMind, National Research University Higher School of Economics (HSE)
3. **Matthew Botvinick**
4. **Daan Wierstra**
5. **Timothy Lillicrap**
   Google DeepMind

One-shot Generation

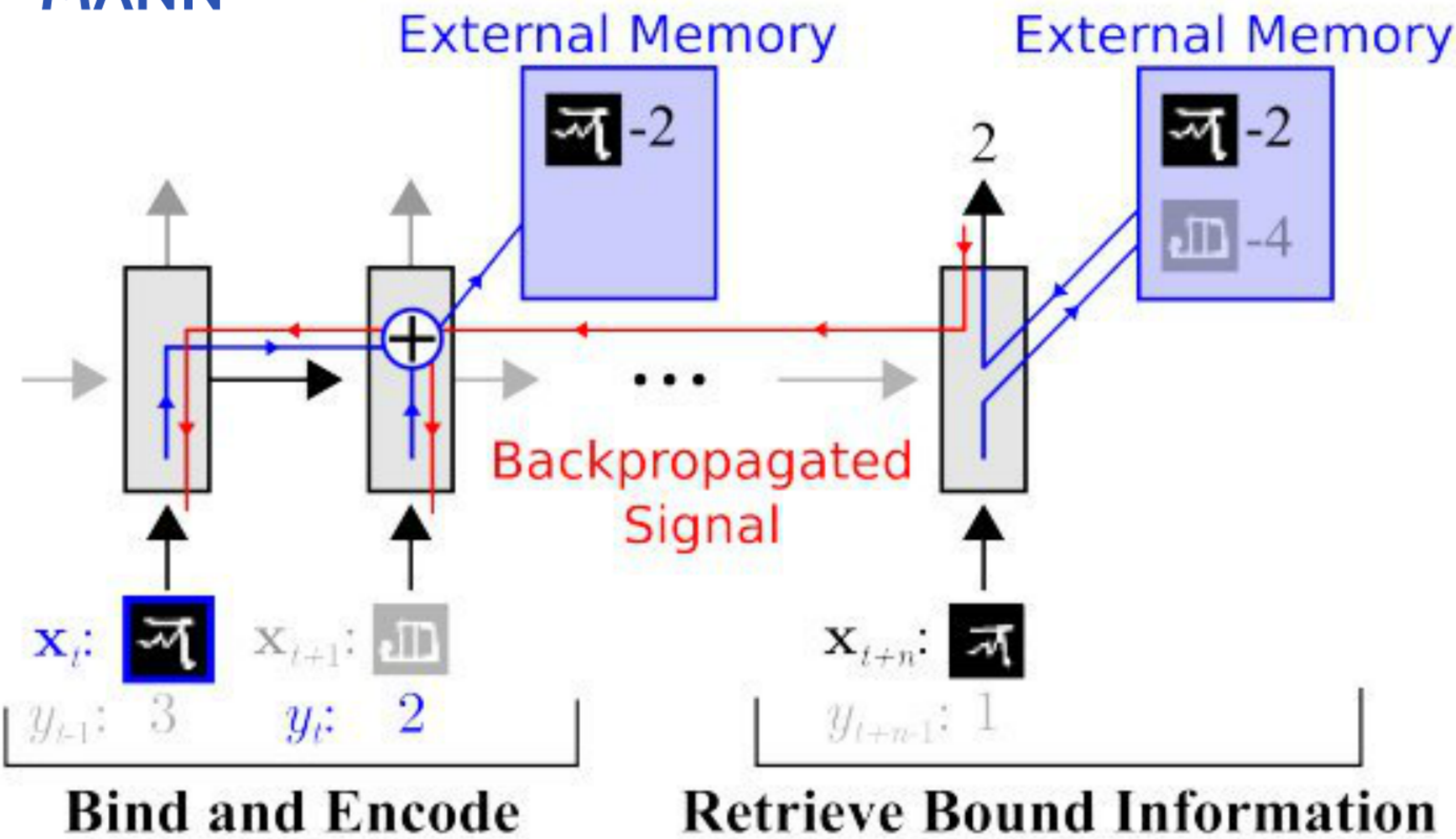
The model



People

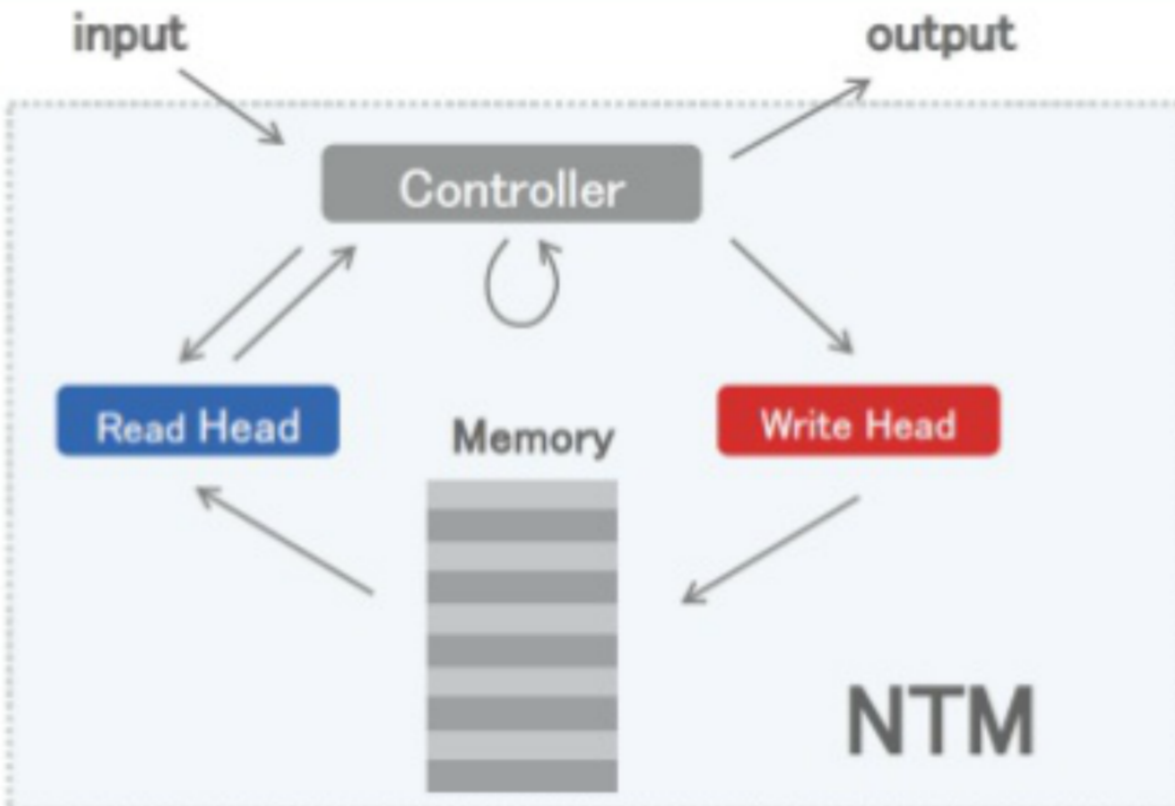


MANN

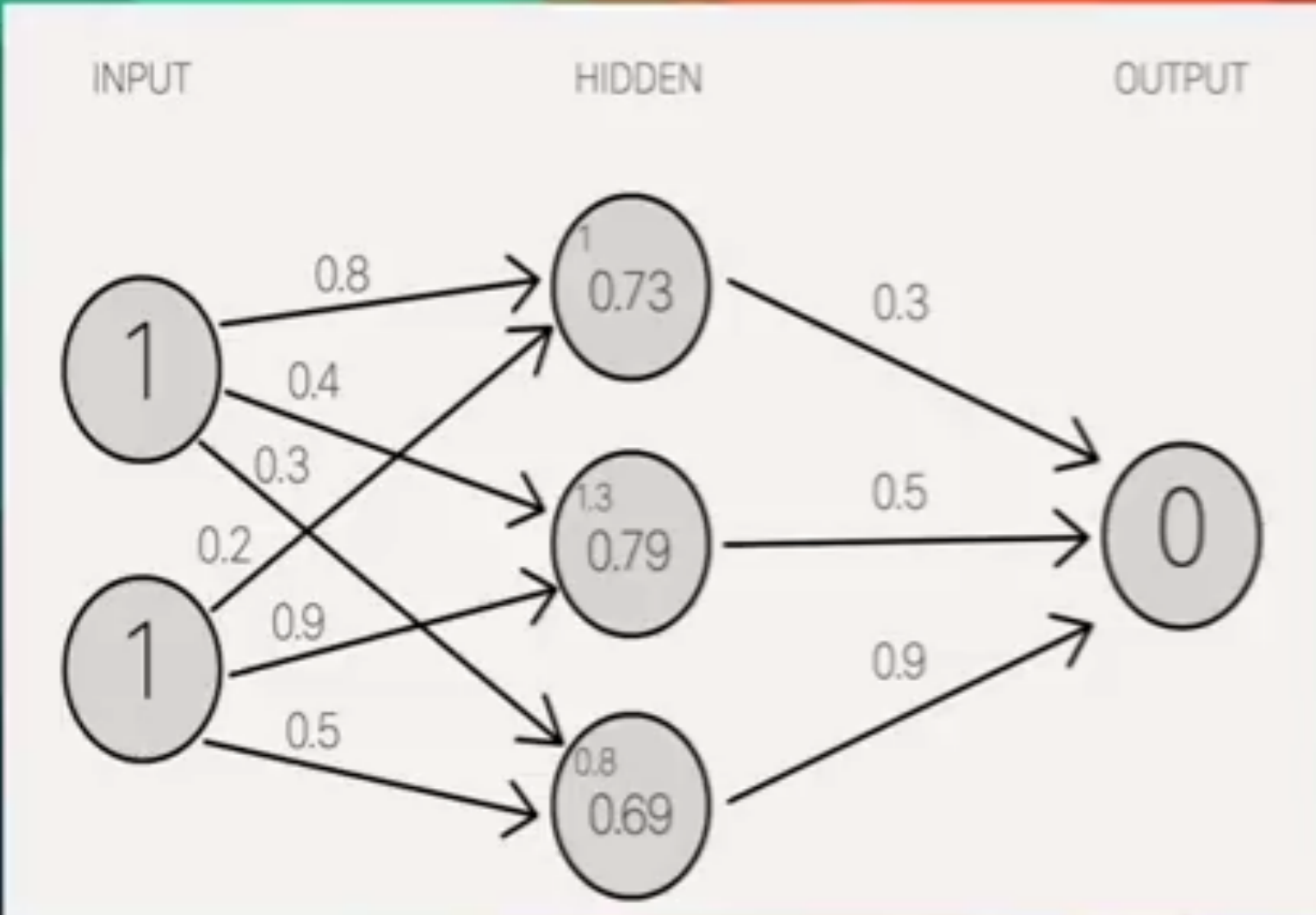


(b) Network strategy

Neural Turing Machine vs Memory Augmented Neural Network



1. MANN은 NTM의 변형이다.
2. Controller는 Feed Forward NN or LSTM을 사용하였다.
3. 기존 NTM은 복사, 정렬등의 알고리즘을 학습하는데 사용하였지만 이 논문에서는 One-shot Learing에 사용
4. 빠른 학습을 위해 Memory에 Write할 때 가장 적게 사용된 메모리 or 가장 최근에 사용된 메모리에 Write함 (LRUA)
5. 그렇다면 기본적인 질문?! 왜 Augmented Memory가 필요할까?


INPUT
HIDDEN
OUTPUT
1
1
0.8
0.4
0.3
0.2
0.9
0.5
1
0.73
1.3
0.79
0.8
0.69
0.3
0.5
0.9
0

Experiment – Data

Omniglot Dataset : 1600 > classes

⇒ 1200 class train, 423 class test (downscale to 20x20)

+ plus rotate augmentation

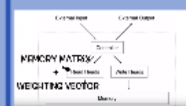
One shot learning is a type of learning that use deep learning networks to build the knowledge.

DEEP LEARNING

Memory Addressing

Content Based

Location Based

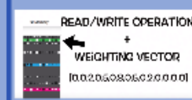


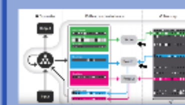
Design Pattern for Deep Learning

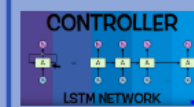
Meta Learning

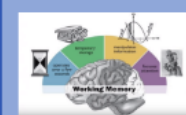
or

Learning to Learn

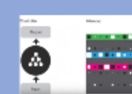


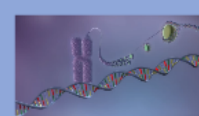






NURAL TURING MACHINE





One shot learning is a type of learning that use deep learning networks to build the knowledge.

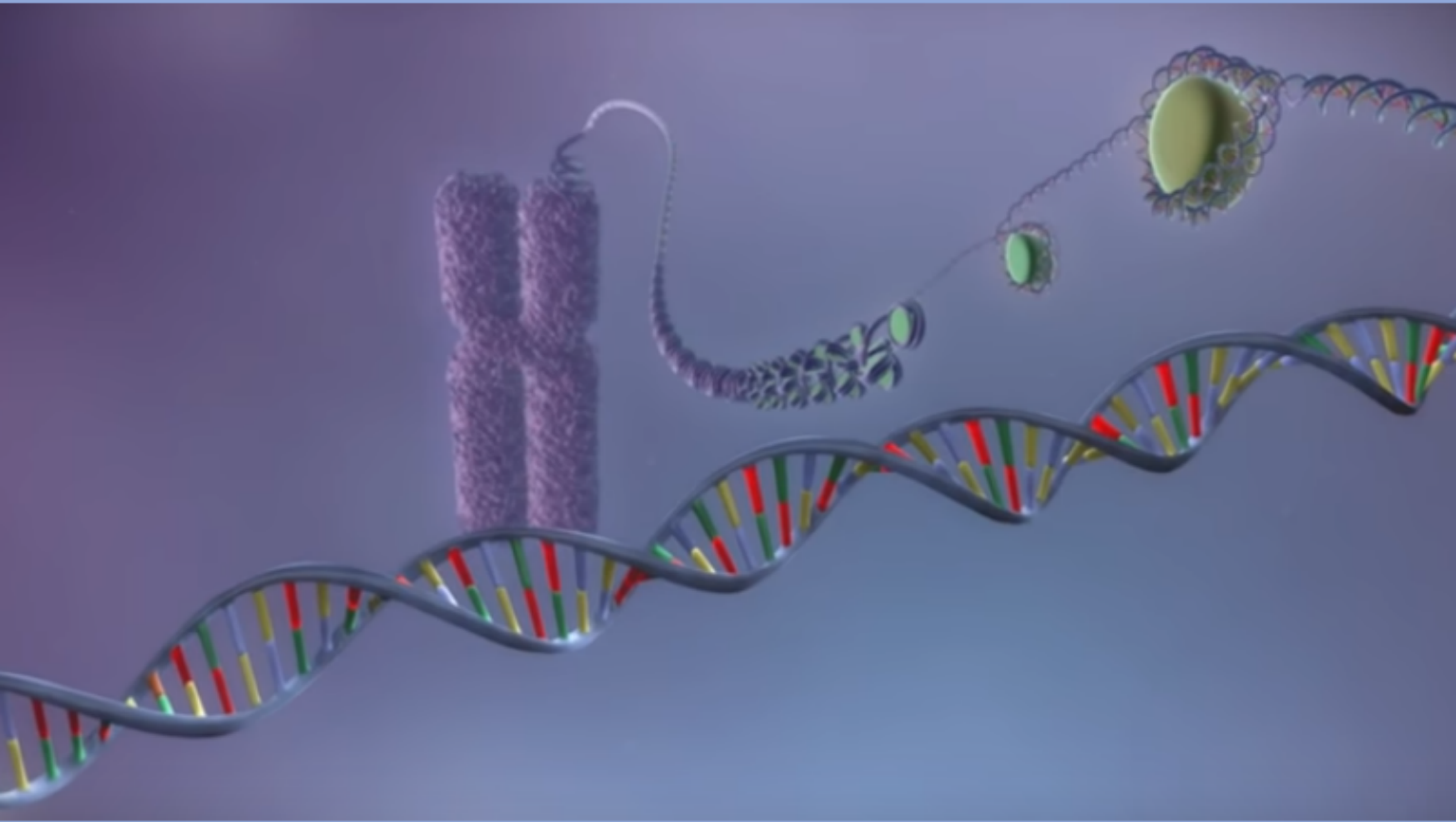
DEEP LEARNING

Design Pattern for Deep Learning

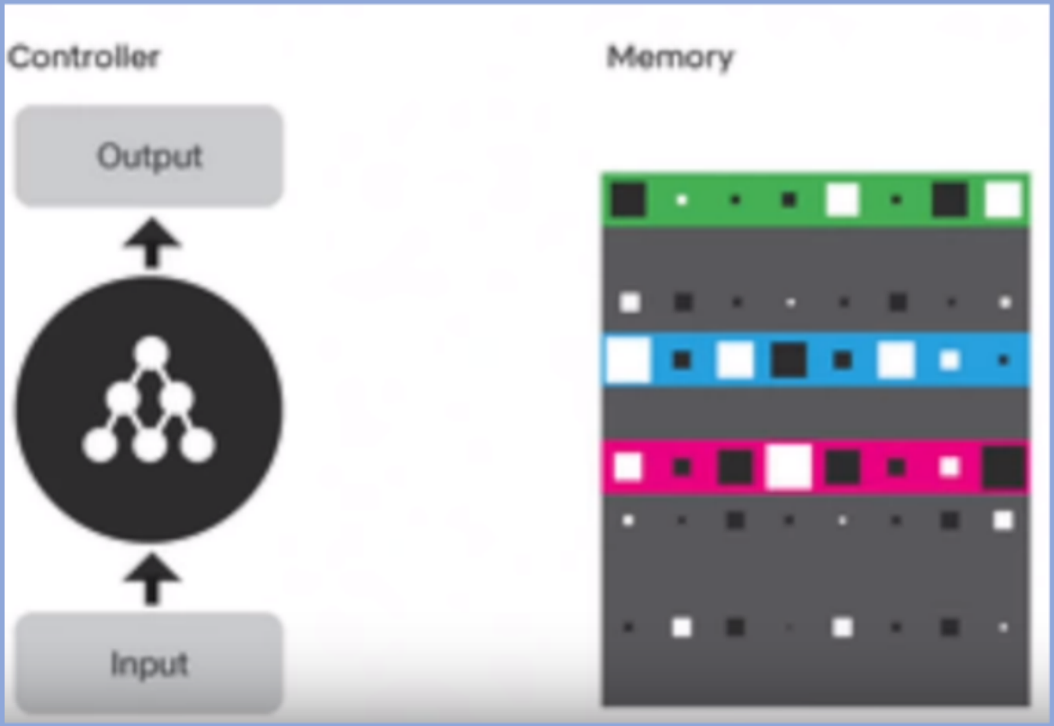
Meta Learning

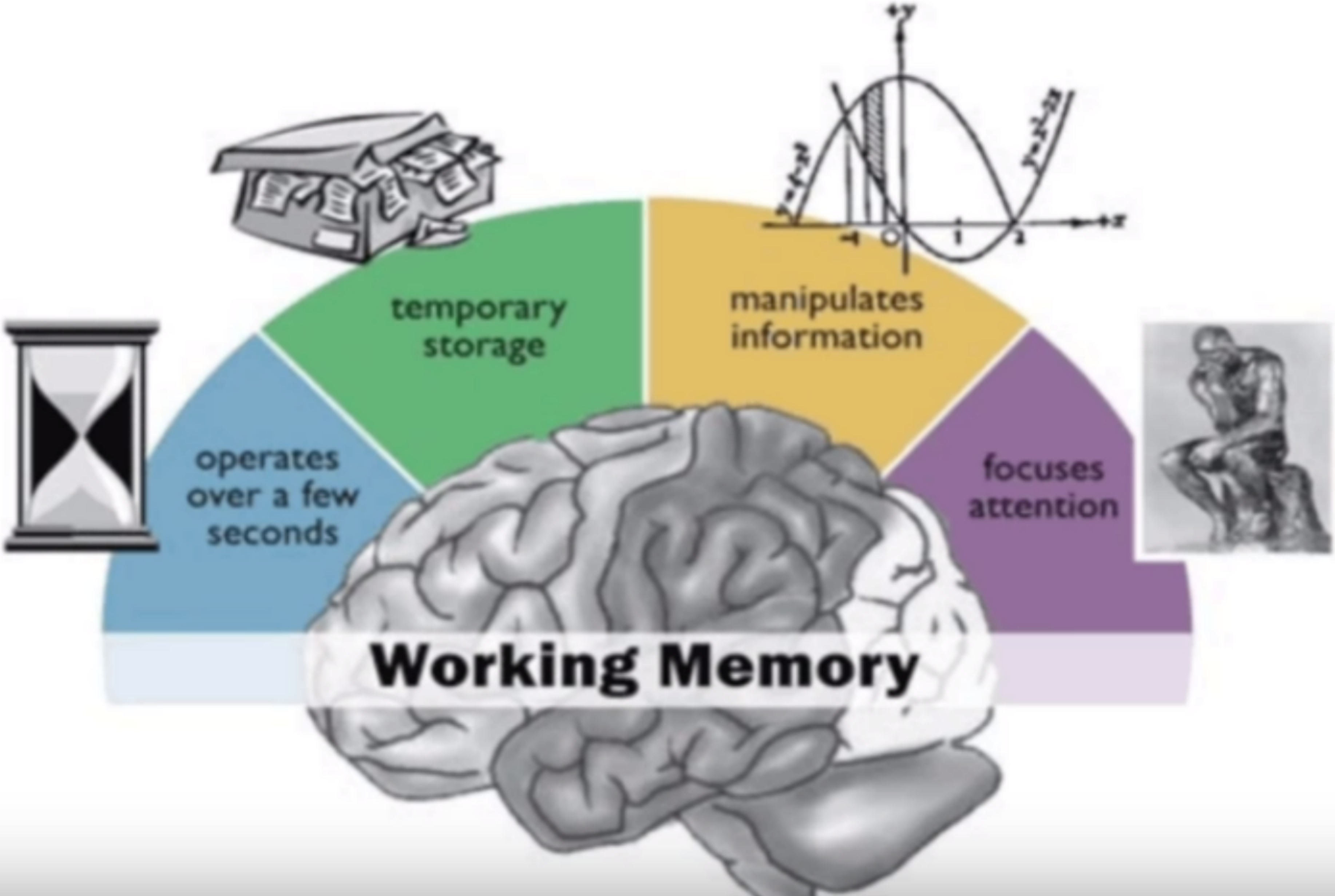
or

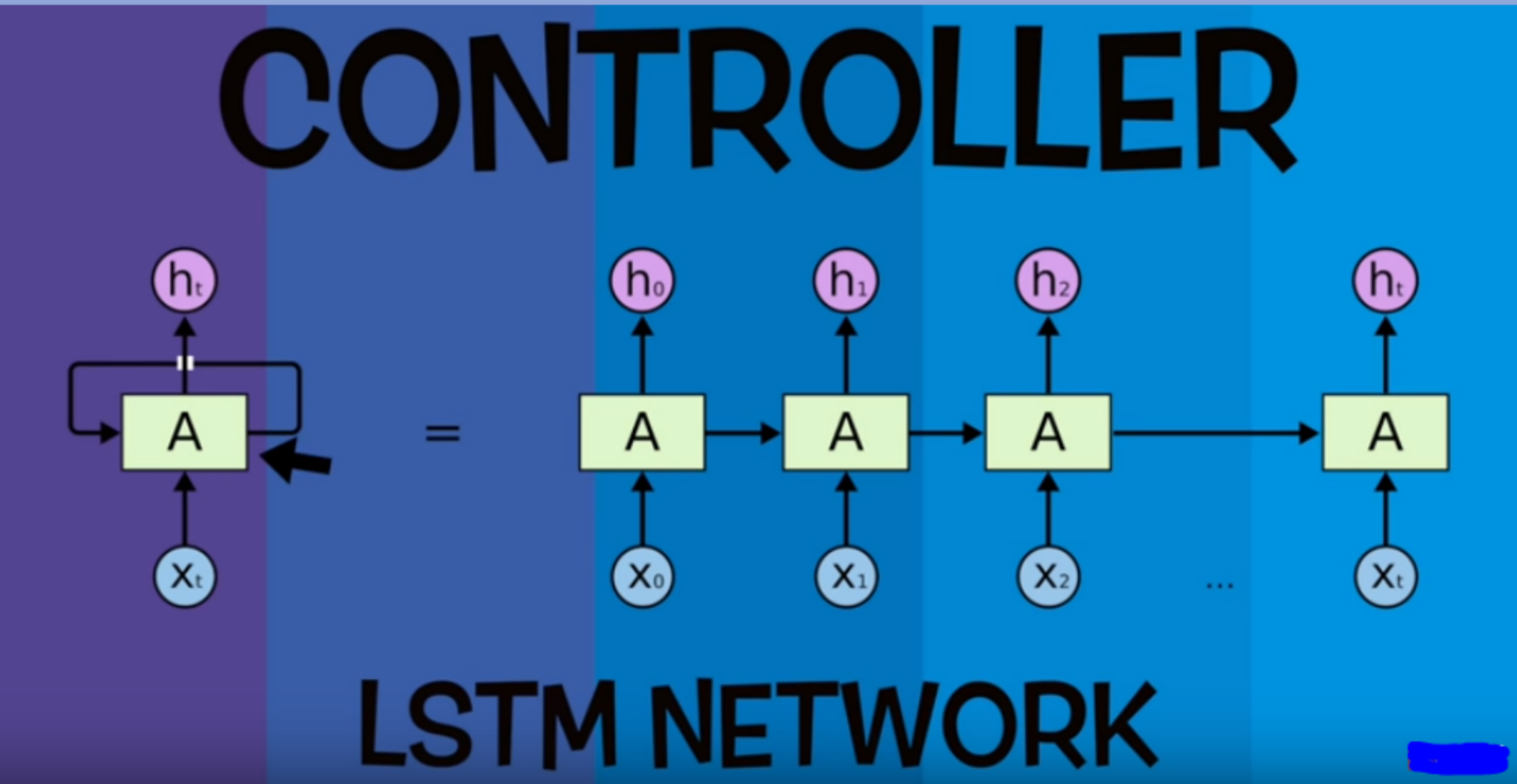
Learning to Learn

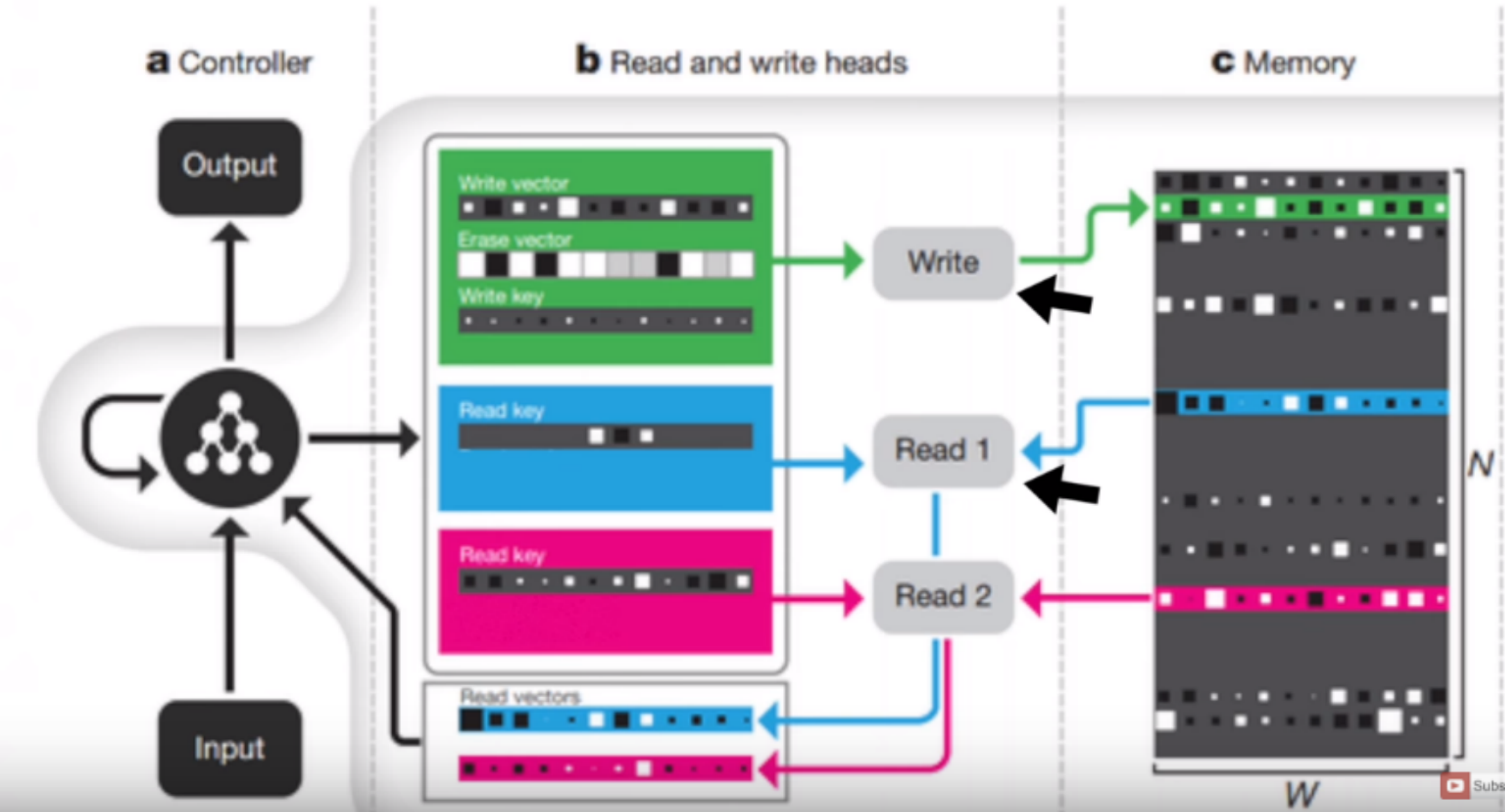


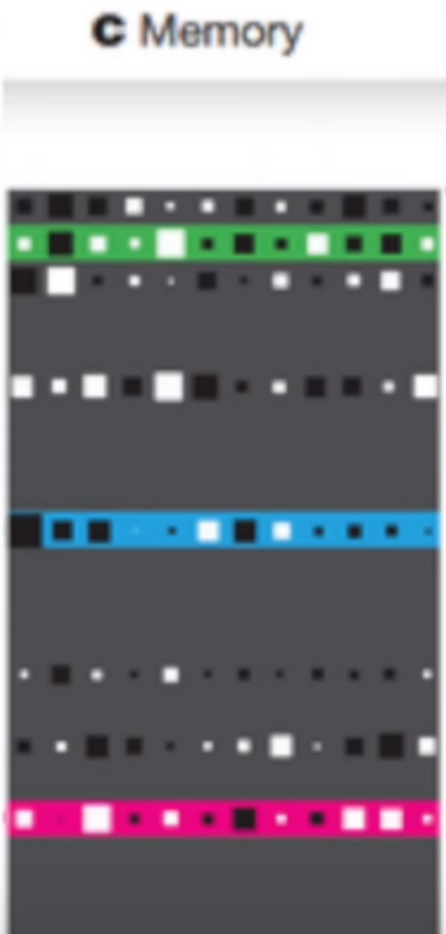
NURAL TURING MACHINE




temporary storage
manipulates information
operates over a few seconds
focuses attention
Working Memory


CONTROLLER
ht
A
Xt
=
h0
h1
h2
ht
A
A
A
A
X0
X1
X2
...
Xt
LSTM NETWORK


a Controller
b Read and write heads
c Memory
Output
Input
Write vector
Erase vector
Write key
Read key
Read key
Write
Read 1
Read 2
Read vectors
N
W

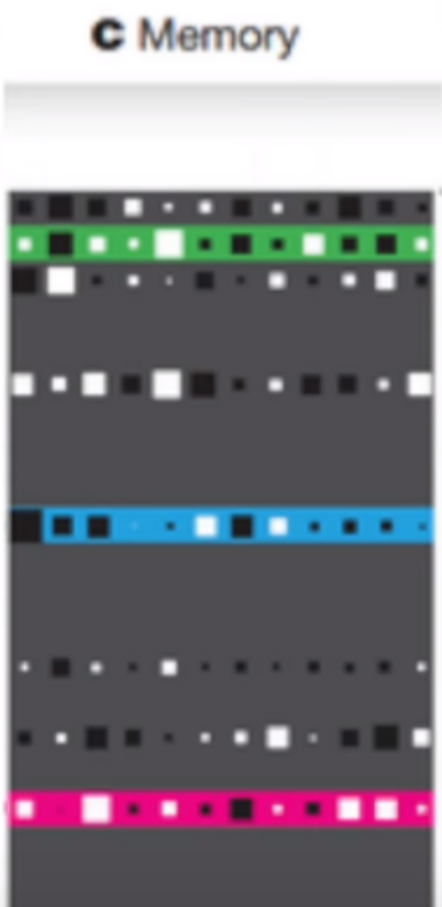


READ/WRITE OPERATION

+

WEIGHTING VECTOR

[0,0,0,1,0,0,0,0,0,0]

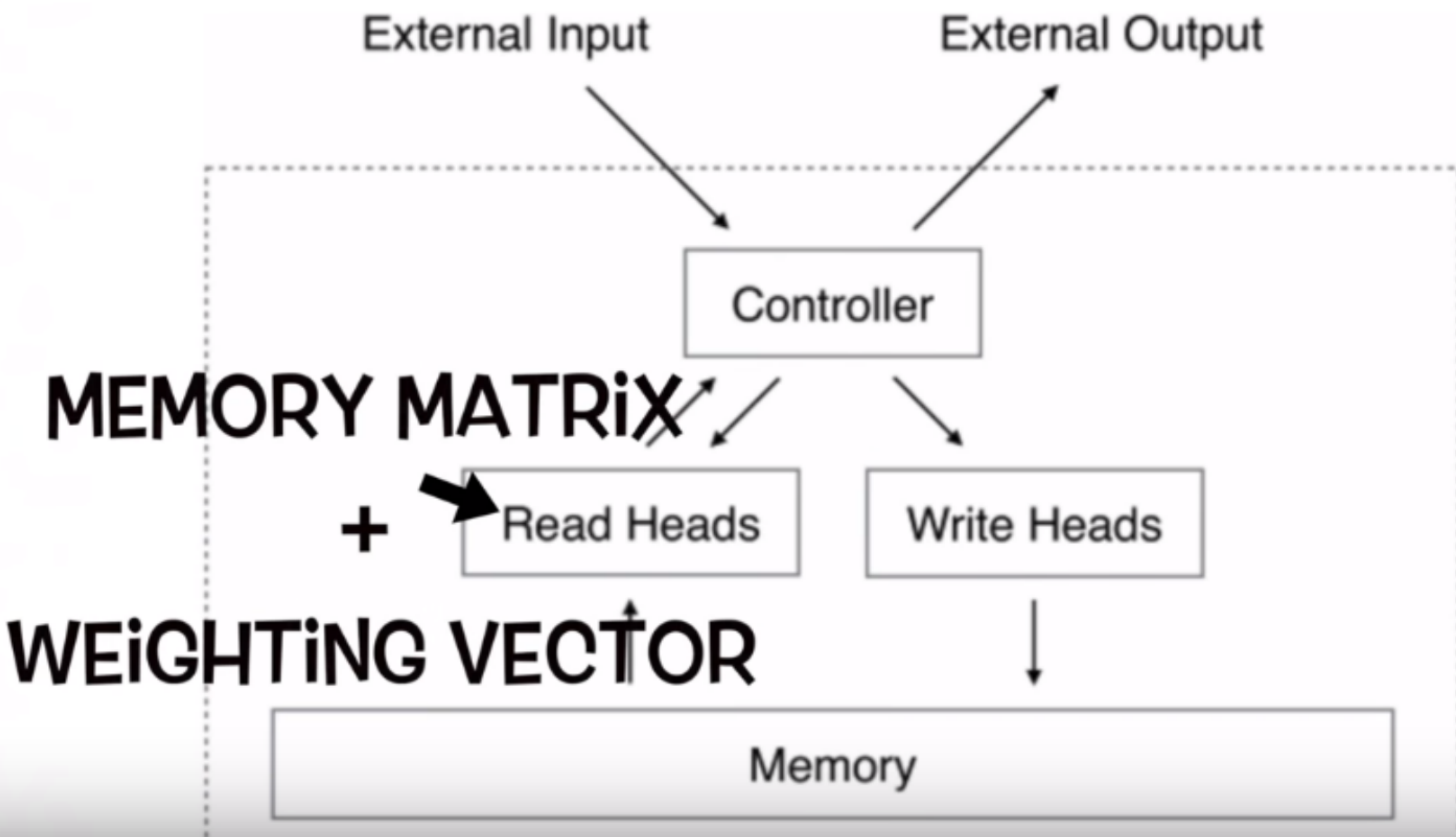


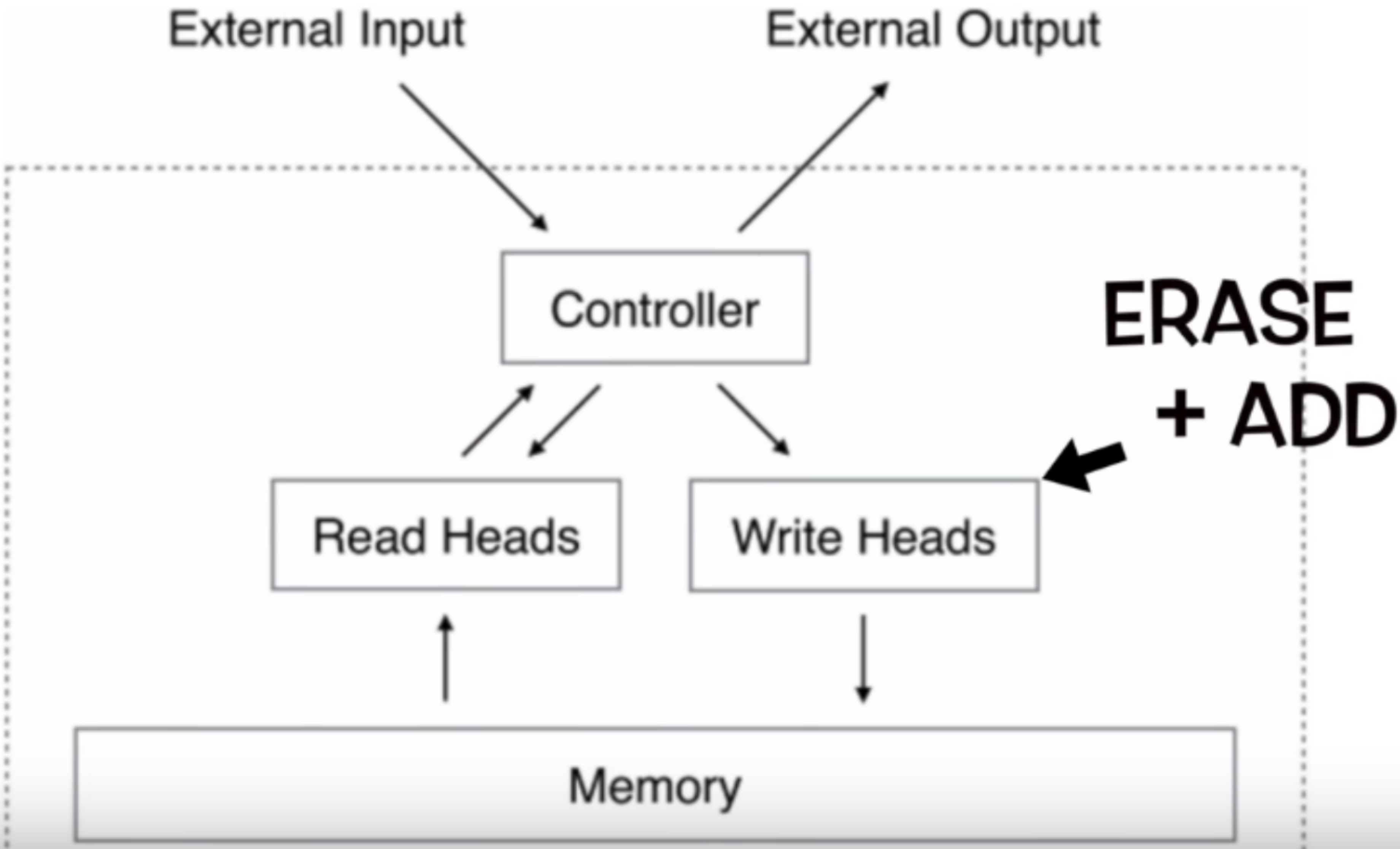
READ/WRITE OPERATION

+

WEiGHTiNG VECTOR

[0,0.2,0.5,0.8,0.5,0.2,0,0,0,0]

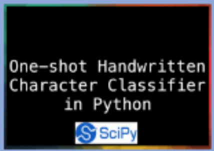

External Input
External Output
Controller
MEMORY MATRIX
+
WEIGHTING VECTOR
Read Heads
Write Heads
Memory

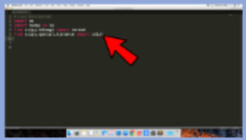
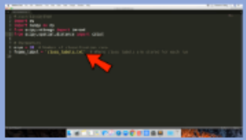

External Input
External Output
Controller
ERASE
+ ADD
Read Heads
Write Heads
Memory

Memory Addressing

Content Based

Location Based







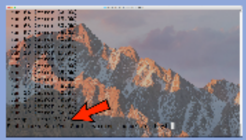
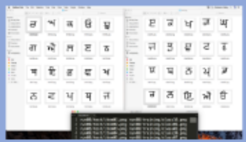










Source

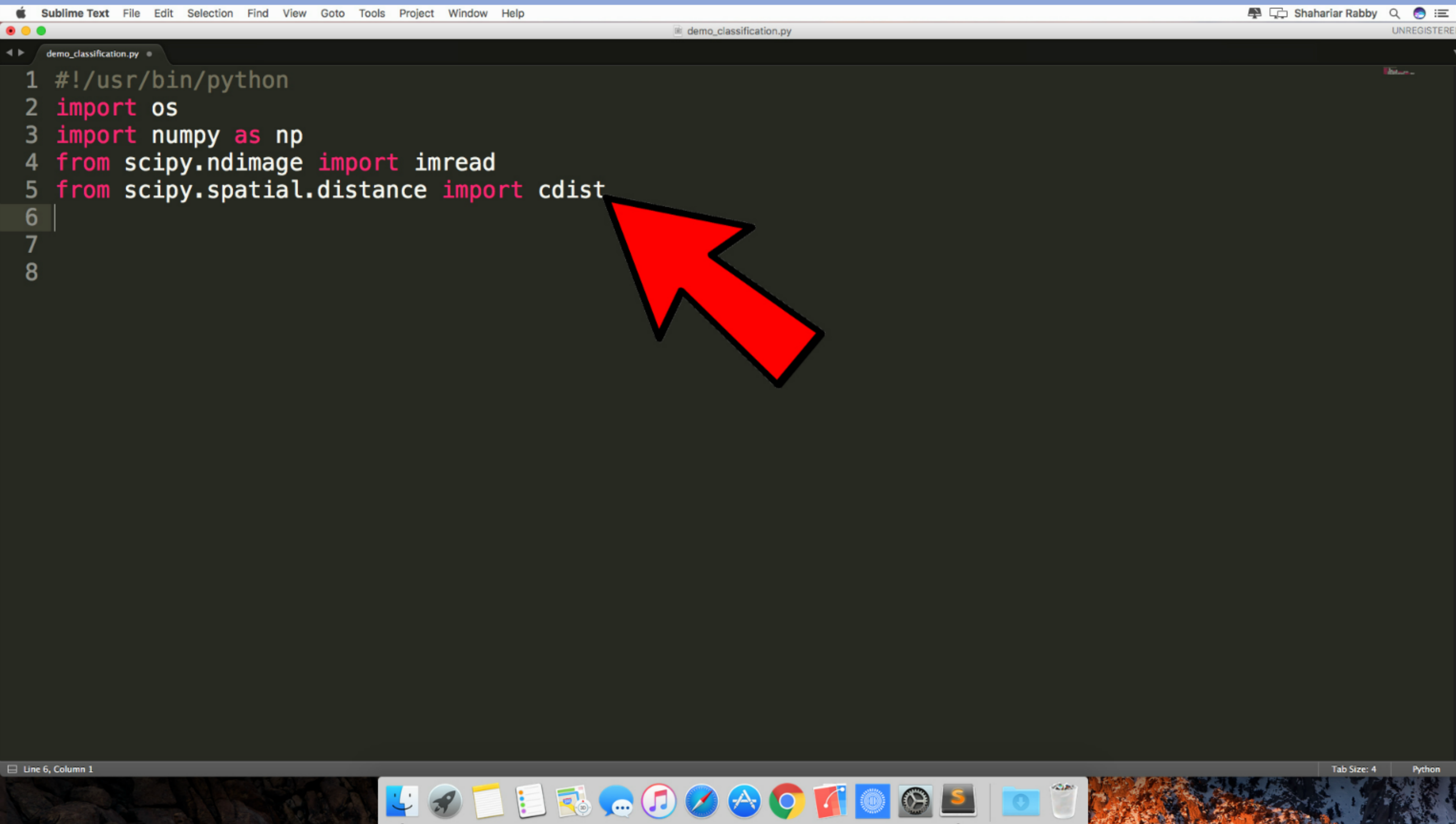
Human Level Concept Learning through Probabilistic Program Induction: http://web.mit.edu/cocosci/Papers/Science-2015-Lake-1332-8.pdf

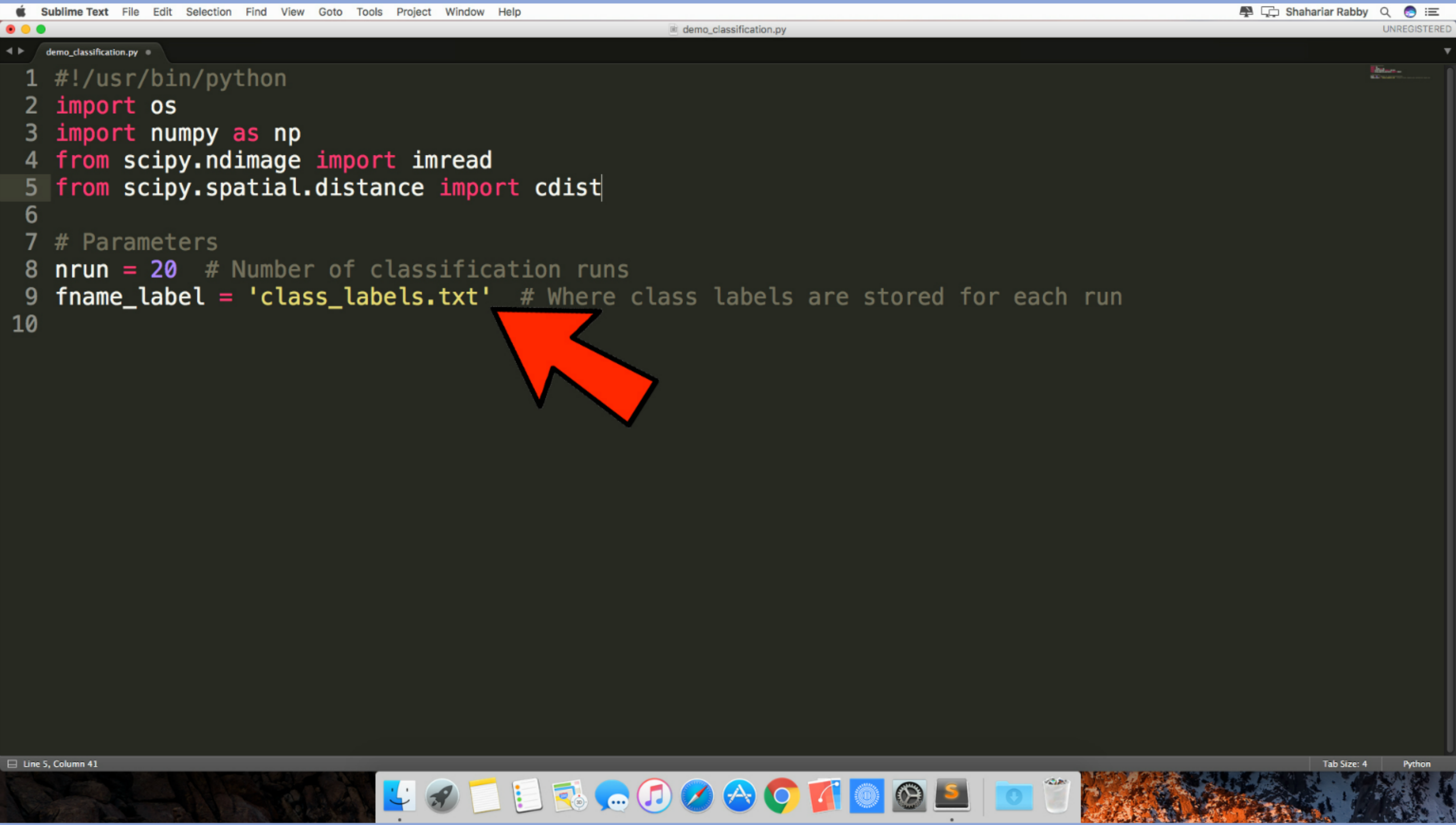
One-Shot Learning with Memory Augmented Neural Networks: https://arxiv.org/pdf/1605.06065v1.pdf

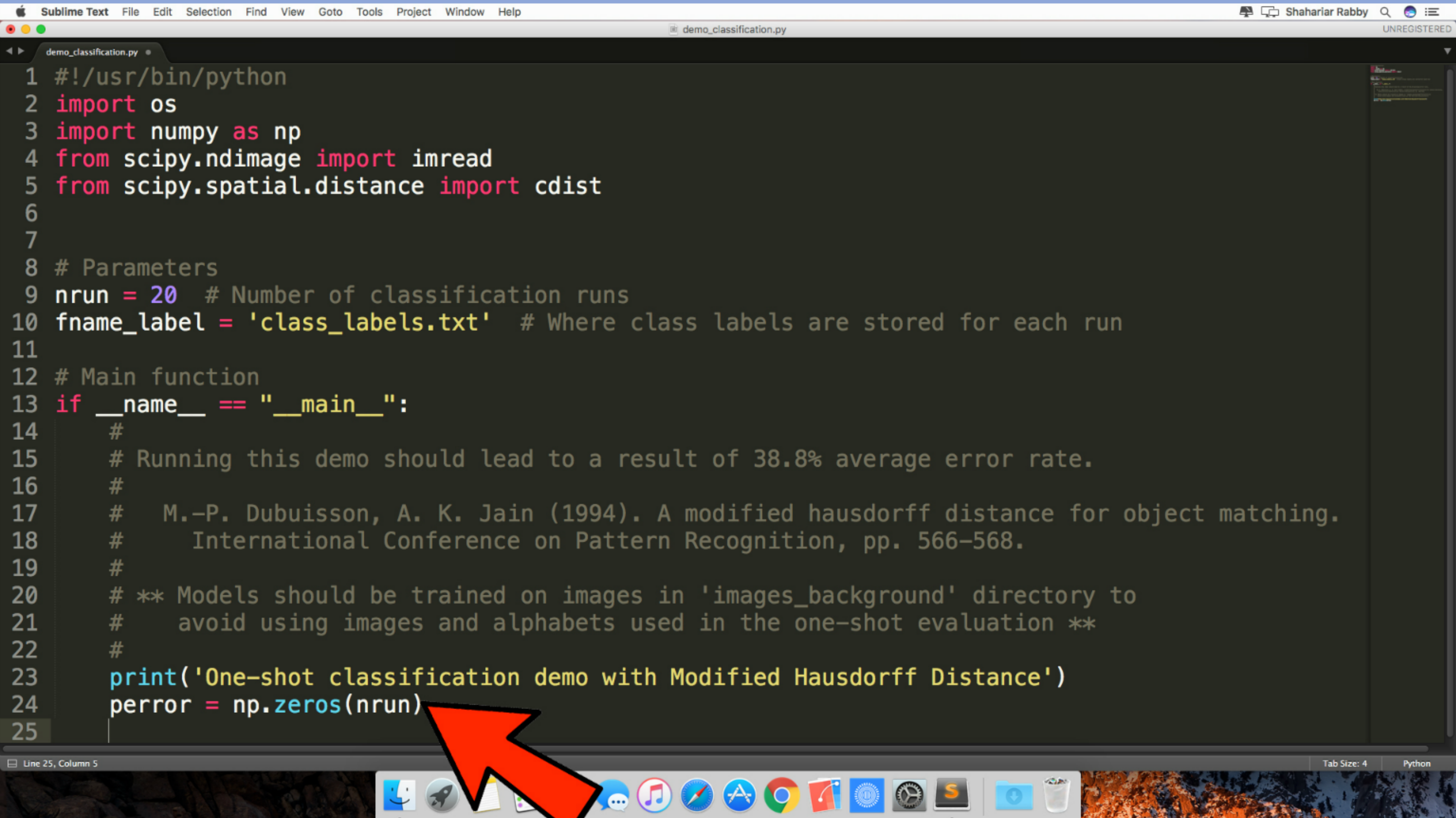
For Code & Papers: https://github.com/shahariarrabby/One-Shot

One-shot Handwritten Character Classifier in Python

SciPy


Sublime Text File Edit Selection Find View Goto Tools Project Window Help
Shahariar Rabby
demo_classification.py
UNREGISTERED
#!/usr/bin/python
import os
import numpy as np
from scipy.ndimage import imread
from scipy.spatial.distance import cdist
Line 6, Column 1
Tab Size: 4
Python


Sublime Text File Edit Selection Find View Goto Tools Project Window Help
Shahariar Rabby
demo_classification.py
UNREGISTERED
#!/usr/bin/python
import os
import numpy as np
from scipy.ndimage import imread
from scipy.spatial.distance import cdist
# Parameters
nrun = 20 # Number of classification runs
fname_label = 'class_labels.txt' # Where class labels are stored for each run
Line 5, Column 41
Tab Size: 4
Python

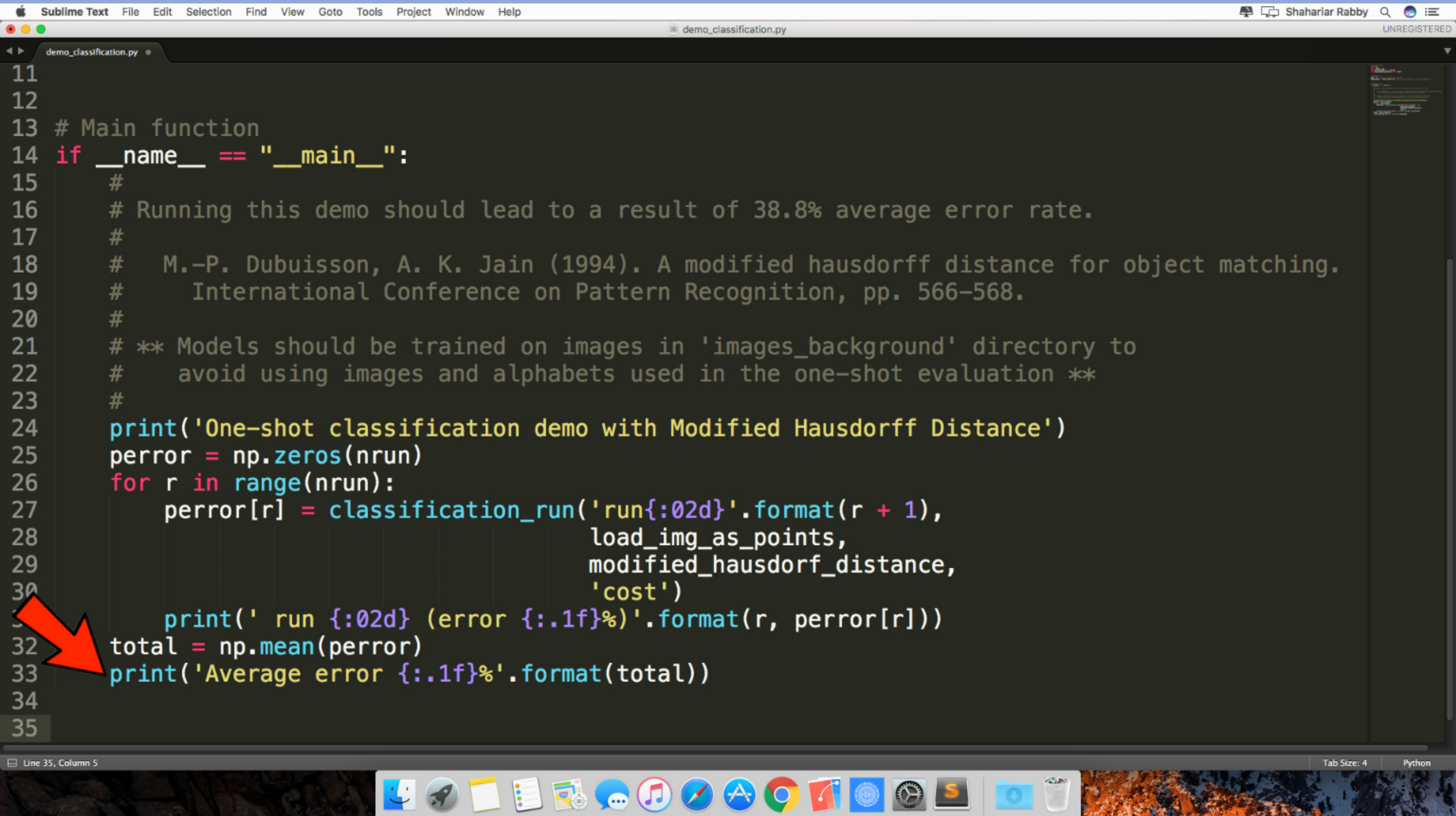

Sublime Text File Edit Selection Find View Goto Tools Project Window Help
Shahariar Rabby
demo_classification.py
UNREGISTERED
#!/usr/bin/python
import os
import numpy as np
from scipy.ndimage import imread
from scipy.spatial.distance import cdist
# Parameters
nrun = 20 # Number of classification runs
fname_label = 'class_labels.txt' # Where class labels are stored for each run
# Main function
if __name__ == "__main__":
    #
    # Running this demo should lead to a result of 38.8% average error rate.
    #
    #   M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #   International Conference on Pattern Recognition, pp. 566-568.
    #
    # ** Models should be trained on images in 'images_background' directory to
    #   avoid using images and alphabets used in the one-shot evaluation **
    #
    print('One-shot classification demo with Modified Hausdorff Distance')
    perror = np.zeros(nrun)
Line 25, Column 5
Tab Size: 4
Python

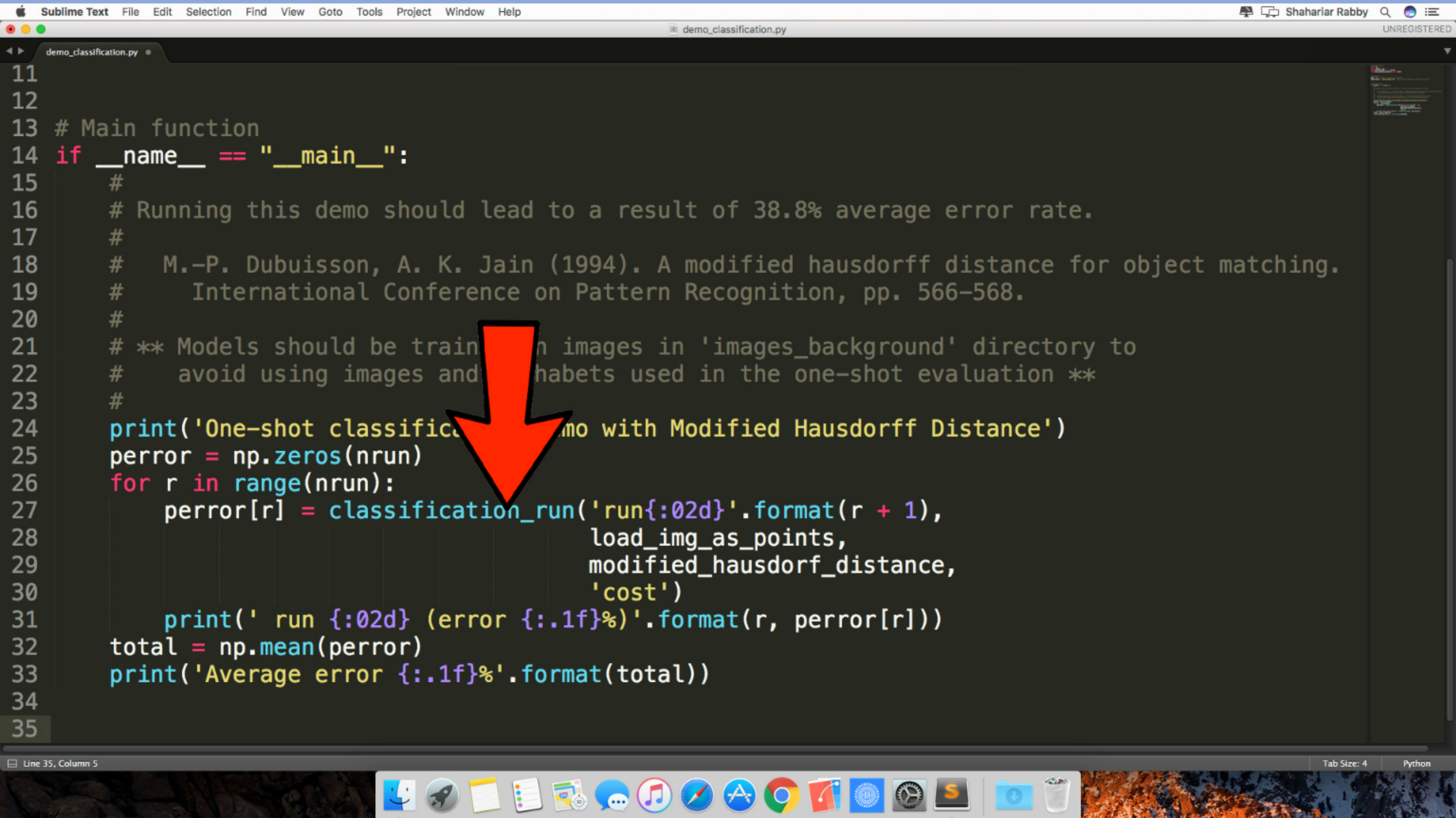
demo_classification.py

```python


# Main function
if __name__ == "__main__":
    #
    # Running this demo should lead to a result of 38.8% average error rate.
    #
    #   M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #     International Conference on Pattern Recognition, pp. 566-568.
    #
    # ** Models should be trained on images in 'images_background' directory to
    #    avoid using images and alphabets used in the one-shot evaluation **
    #
    print('One-shot classification demo with Modified Hausdorff Distance')
    perror = np.zeros(nrun)
    for r in range(nrun):
        perror[r] = classification_run('run{:02d}'.format(r + 1),
                                       load_img_as_points,
                                       modified_hausdorf_distance,
                                       'cost')
        print(' run {:02d} (error {:.1f}%)'.format(r, perror[r]))
    total = np.mean(perror)
    print('Average error {:.1f}%'.format(total))

```

```python


# Main function
if __name__ == "__main__":
    #
    # Running this demo should lead to a result of 38.8% average error rate.
    #
    #   M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #     International Conference on Pattern Recognition, pp. 566-568.
    #
    # ** Models should be trained on images in 'images_background' directory to
    #    avoid using images and alphabets used in the one-shot evaluation **
    #
    print('One-shot classification demo with Modified Hausdorff Distance')
    perror = np.zeros(nrun)
    for r in range(nrun):
        perror[r] = classification_run('run{:02d}'.format(r + 1),
                                       load_img_as_points,
                                       modified_hausdorff_distance,
                                       'cost')
        print(' run {:02d} (error {:.1f}%)'.format(r, perror[r]))
    total = np.mean(perror)
    print('Average error {:.1f}%'.format(total))

```

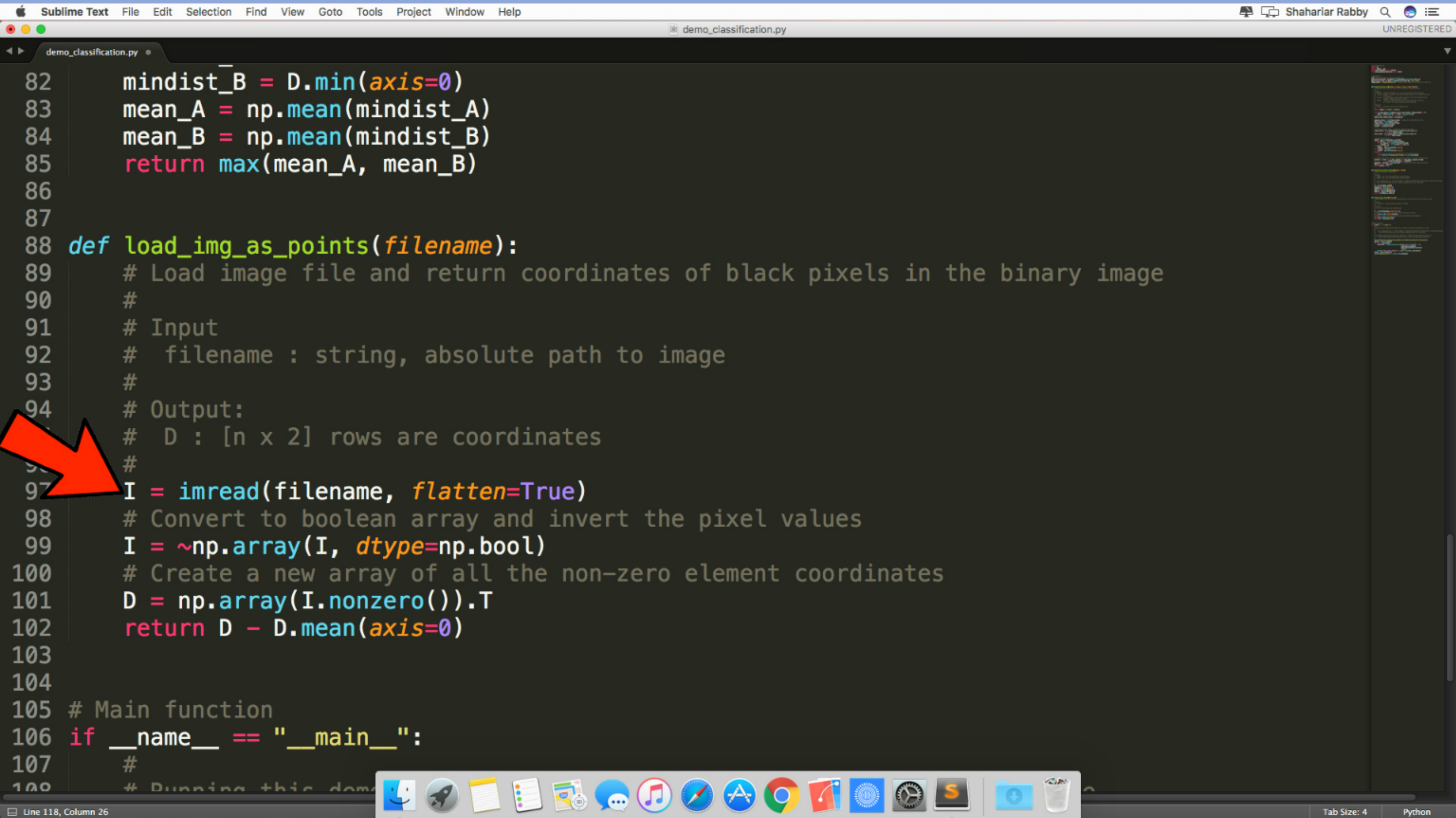
```python
# Main function
if __name__ == "__main__":
    #
    # Running this demo should lead to a result of 38.8% average error rate.
    #
    #   M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #     International Conference on Pattern Recognition, pp. 566-568.
    #
    # ** Models should be trained on images in 'images_background' directory to
    #    avoid using images and alphabets used in the one-shot evaluation **
    #
    print('One-shot classification demo with Modified Hausdorff Distance')
    perror = np.zeros(nrun)
    for r in range(nrun):
        perror[r] = classification_run('run{:02d}'.format(r + 1),
                                       load_img_as_points,
                                       modified_hausdorf_distance,
                                       'cost')
        print(' run {:02d} (error {:.1f}%)'.format(r, perror[r]))
    total = np.mean(perror)
    print('Average error {:.1f}%'.format(total))
```

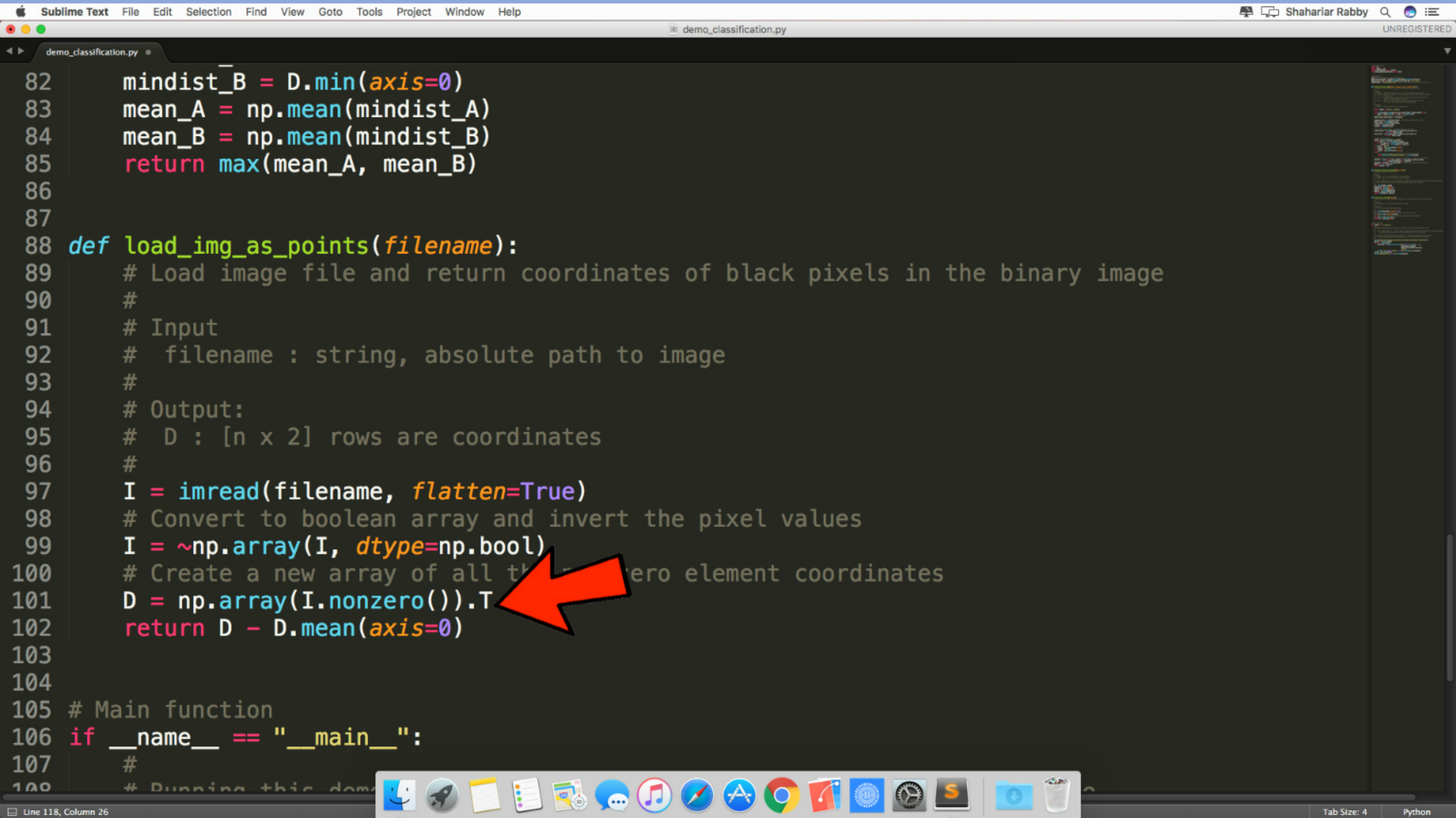

Sublime Text File Edit Selection Find View Goto Tools Project Window Help
Shahariar Rabby
demo_classification.py
UNREGISTERED
# Main function
if __name__ == "__main__":
    #
    # Running this demo should lead to a result of 38.8% average error rate.
    #
    #   M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #   International Conference on Pattern Recognition, pp. 566-568.
    #
    # ** Models should be trained on images in 'images_background' directory to
    #   avoid using images and alphabets used in the one-shot evaluation **
    #
    print('One-shot classification demo with Modified Hausdorff Distance')
    perror = np.zeros(nrun)
    for r in range(nrun):
        perror[r] = classification_run('run{:02d}'.format(r + 1),
                                       load_img_as_points,
                                       modified_hausdorf_distance,
                                       'cost')
        print(' run {:02d} (error {:.1f}%)'.format(r, perror[r]))
    total = np.mean(perror)
    print('Average error {:.1f}%'.format(total))
Line 35, Column 5
Tab Size: 4
Python


Sublime Text File Edit Selection Find View Goto Tools Project Window Help
Shahariar Rabby
demo_classification.py
UNREGISTERED
# Main function
if __name__ == "__main__":
#
Running this demo should lead to a result of 38.8% average error rate.
#
M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
International Conference on Pattern Recognition, pp. 566-568.
#
** Models should be train
n images in 'images_background' directory to
avoid using images and
habets used in the one-shot evaluation **
#
print('One-shot classific
mo with Modified Hausdorff Distance')
perror = np.zeros(nrun)
for r in range(nrun):
perror[r] = classification_run('run{:02d}'.format(r + 1),
load_img_as_points,
modified_hausdorf_distance,
'cost')
print(' run {:02d} (error {:.1f}%)'.format(r, perror[r]))
total = np.mean(perror)
print('Average error {:.1f}%'.format(total))
Line 35, Column 5
Tab Size: 4
Python

demo_classification.py

```python

# Main function
if __name__ == "__main__":
    #
    # Running this demo should lead to a result of 38.8% average error rate.
    #
    #   M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #     International Conference on Pattern Recognition, pp. 566-568.
    #
    # ** Models should be trained on images in 'images_background' directory to
    #    avoid using images and alphabets used in the one-shot evaluation **
    #
    print('One-shot classification demo with Modified Hausdorff Distance')
    perror = np.zeros(nrun)
    for r in range(nrun):
        perror[r] = classification_run('run{:02d}'.format(r + 1),
                                       load_img_as_points,
                                       modified_hausdorff_distance,
                                       'cost')
        print(' run {:02d} (error {:.1f}%)'.format(r, perror[r]))
    total = np.mean(perror)
    print('Average error {:.1f}%'.format(total))

```


Sublime Text File Edit Selection Find View Goto Tools Project Window Help
Shahariar Rabby
demo_classification.py
UNREGISTERED
demo_classification.py
    mindist_B = D.min(axis=0)
    mean_A = np.mean(mindist_A)
    mean_B = np.mean(mindist_B)
    return max(mean_A, mean_B)
def load_img_as_points(filename):
    # Load image file and return coordinates of black pixels in the binary image
    #
    # Input
    # filename : string, absolute path to image
    #
    # Output:
    # D : [n x 2] rows are coordinates
    #
    I = imread(filename, flatten=True)
    # Convert to boolean array and invert the pixel values
    I = ~np.array(I, dtype=np.bool)
    # Create a new array of all the non-zero element coordinates
    D = np.array(I.nonzero()).T
    return D - D.mean(axis=0)
# Main function
if __name__ == "__main__":
    #
Line 118, Column 26
Tab Size: 4
Python


Sublime Text File Edit Selection Find View Goto Tools Project Window Help
Shahariar Rabby
demo_classification.py
UNREGISTERED
mindist_B = D.min(axis=0)
mean_A = np.mean(mindist_A)
mean_B = np.mean(mindist_B)
return max(mean_A, mean_B)
def load_img_as_points(filename):
# Load image file and return coordinates of black pixels in the binary image
#
Input
filename : string, absolute path to image
#
Output:
D : [n x 2] rows are coordinates
#
I = imread(filename, flatten=True)
# Convert to boolean array and invert the pixel values
I = ~np.array(I, dtype=np.bool)
# Create a new array of all th
ero element coordinates
D = np.array(I.nonzero()).T
return D - D.mean(axis=0)
# Main function
if __name__ == "__main__":
#
Line 118, Column 26
Tab Size: 4
Python

Sublime Text File Edit Selection Find View Goto Tools Project Window Help Shahariar Rabby

demo_classification.py UNREGISTERED

demo_classification.py

```
    I = ~np.array(I, dtype=np.bool)
    # Create a new array of all the non-zero element coordinates
    D = np.array(I.nonzero()).T
    return D - D.mean(axis=0)

def modified_hausdorf_distance(itemA, itemB):
    # Modified Hausdorff Distance
    #
    # Input
    #  itemA : [n x 2] coordinates of black pixels
    #  itemB : [m x 2] coordinates of black pixels
    #
    #  M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #  International Conference on Pattern Recognition, pp. 566-568.
    #
    D = cdist(itemA, itemB)
    mindist_A = D.min(axis=1)
    mindist_B = D.min(axis=0)
    mean_A = np.mean(mindist_A)
    mean_B = np.mean(mindist_B)
    return max(mean_A, mean_B)


```

Line 69, Column 5 Tab Size: 4 Python

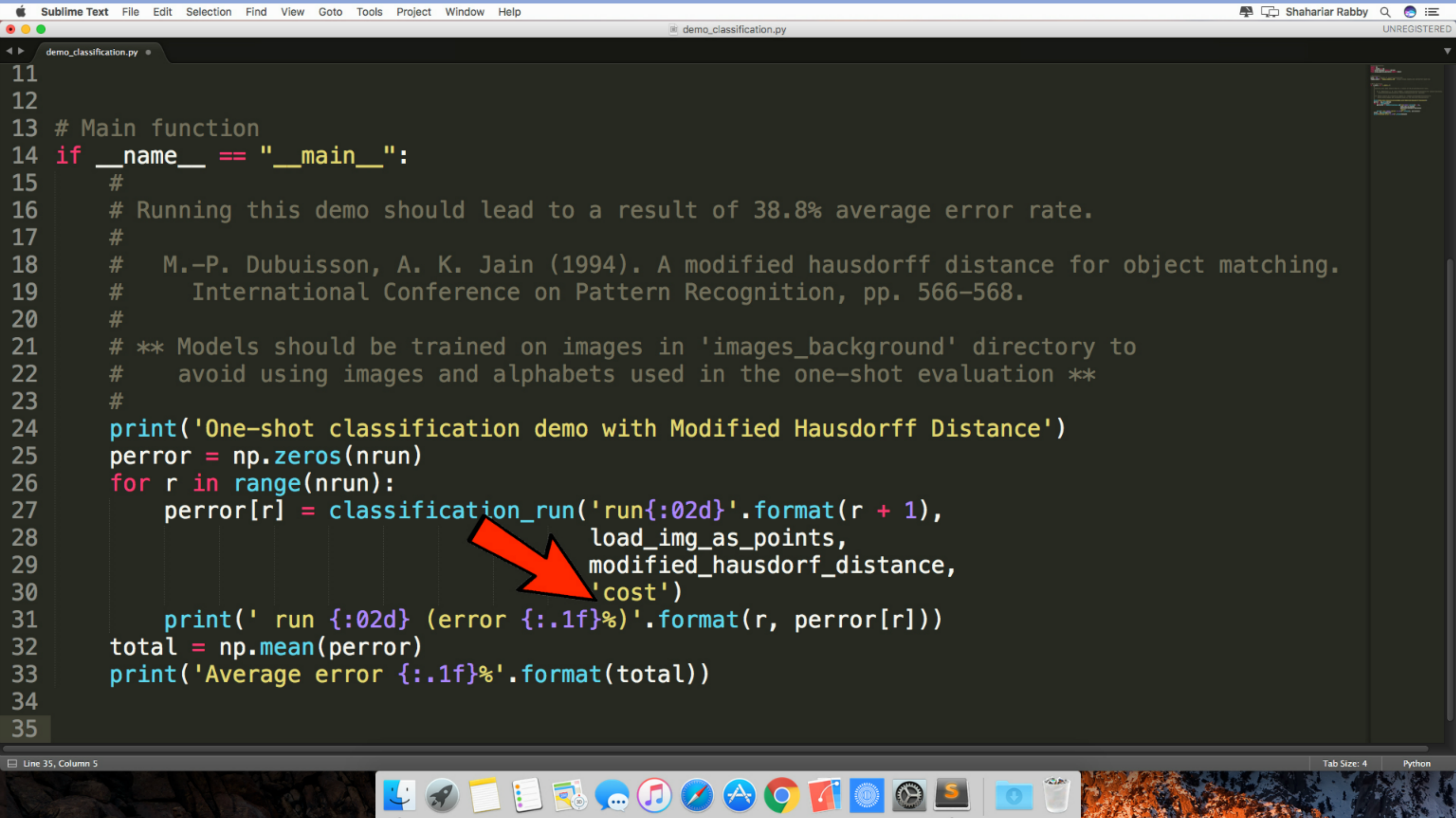
Sublime Text File Edit Selection Find View Goto Tools Project Window Help Shahariar Rabby

demo_classification.py UNREGISTERED

demo_classification.py

```python
    I = ~np.array(I, dtype=np.bool)
    # Create a new array of all the non-zero element coordinates
    D = np.array(I.nonzero()).T
    return D - D.mean(axis=0)

def modified_hausdorff_distance(itemA, itemB):
    # Modified Hausdorff Distance
    #
    # Input
    #  itemA : [n x 2] coordinates of black pixels
    #  itemB : [m x 2] coordinates of black pixels
    #
    #  M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #  International Conference on Pattern Recognition, pp. 566-568.
    #
    D = cdist(itemA, itemB)
    mindist_A = D.min(axis=1)
    mindist_B = D.min(axis=0)
    mean_A = np.mean(mindist_A)
    mean_B = np.mean(mindist_B)
    return max(mean_A, mean_B)
```

Line 69, Column 5 Tab Size: 4 Python



```python


# Main function
if __name__ == "__main__":
    #
    # Running this demo should lead to a result of 38.8% average error rate.
    #
    #   M.-P. Dubuisson, A. K. Jain (1994). A modified hausdorff distance for object matching.
    #     International Conference on Pattern Recognition, pp. 566-568.
    #
    # ** Models should be trained on images in 'images_background' directory to
    #    avoid using images and alphabets used in the one-shot evaluation **
    #
    print('One-shot classification demo with Modified Hausdorff Distance')
    perror = np.zeros(nrun)
    for r in range(nrun):
        perror[r] = classification_run('run{:02d}'.format(r + 1),
                                       load_img_as_points,
                                       modified_hausdorf_distance,
                                       'cost')
        print(' run {:02d} (error {:.1f}%)'.format(r, perror[r]))
    total = np.mean(perror)
    print('Average error {:.1f}%'.format(total))

```

Sublime Text File Edit Selection Find View Goto Tools Project Window Help   Shahariar Rabby

demo_classification.py   UNREGISTERED

demo_classification.py

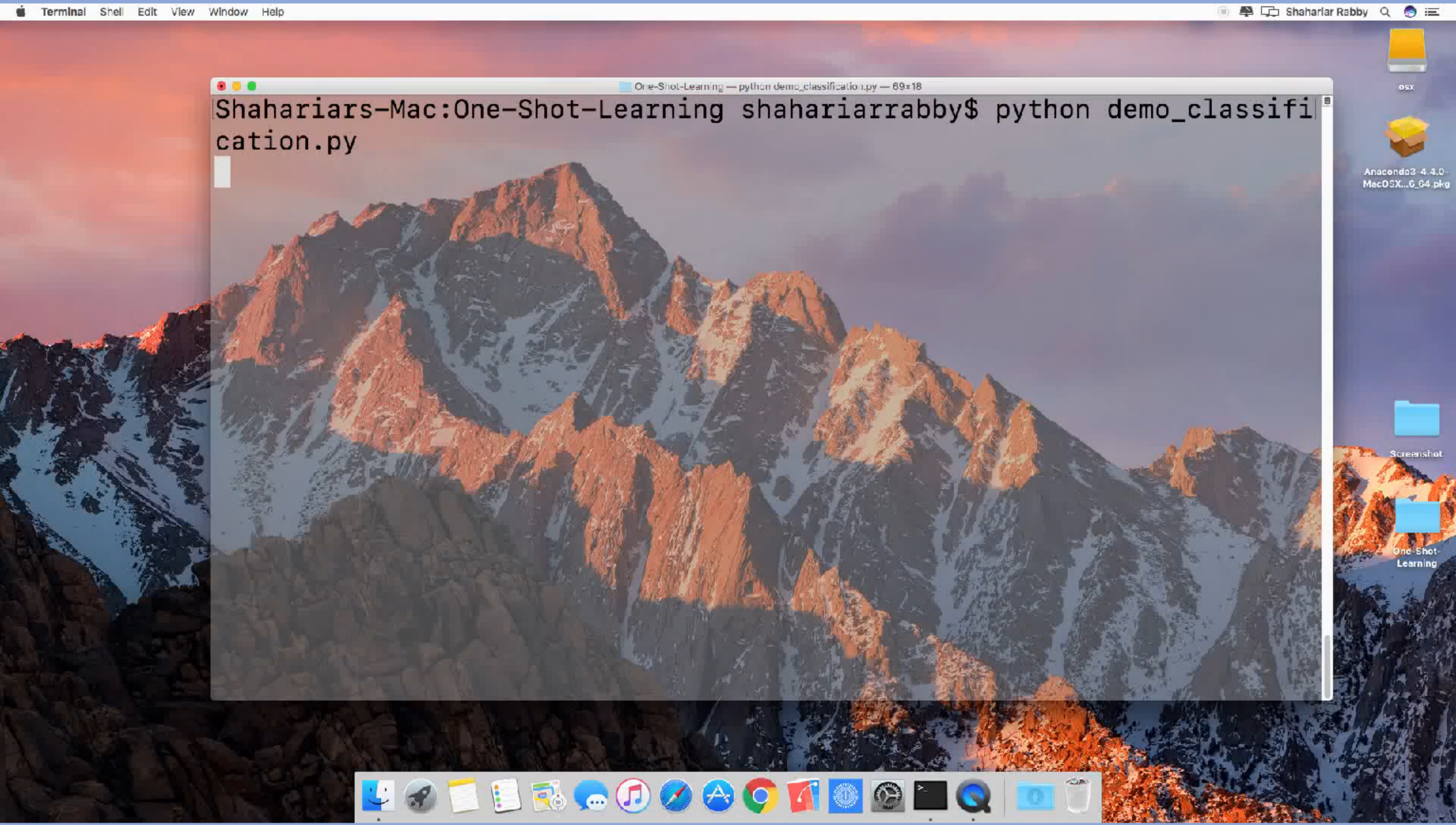
```python
def classification_run(folder, f_load, f_cost, ftype='cost'):
    # Compute error rate for one run of one-shot classification
    #
    # Input
    #  folder : contains images for a run of one-shot classification
    #  f_load : itemA = f_load('file.png') should read in the image file and
    #           process it
    #  f_cost : f_cost(itemA,itemB) should compute similarity between two
    #           images, using output of f_load
    #  ftype  : 'cost' if small values from f_cost mean more similar,
    #           or 'score' if large values are more similar
    #
    # Output
    #  perror : percent errors (0 to 100% error)
    #
    assert ftype in {'cost', 'score'}

    with open(os.path.join(path_to_all_runs, folder, fname_label)) as f:
        pairs = [line.split() for line in f.readlines()]
    # Unzip the pairs into two sets of tuples
    test_files, train_files = zip(*pairs)

    answers_files = list(train_files)  # Copy the training file list
    test_files = sorted(test_files)
    train_files = sorted(train_files)
```

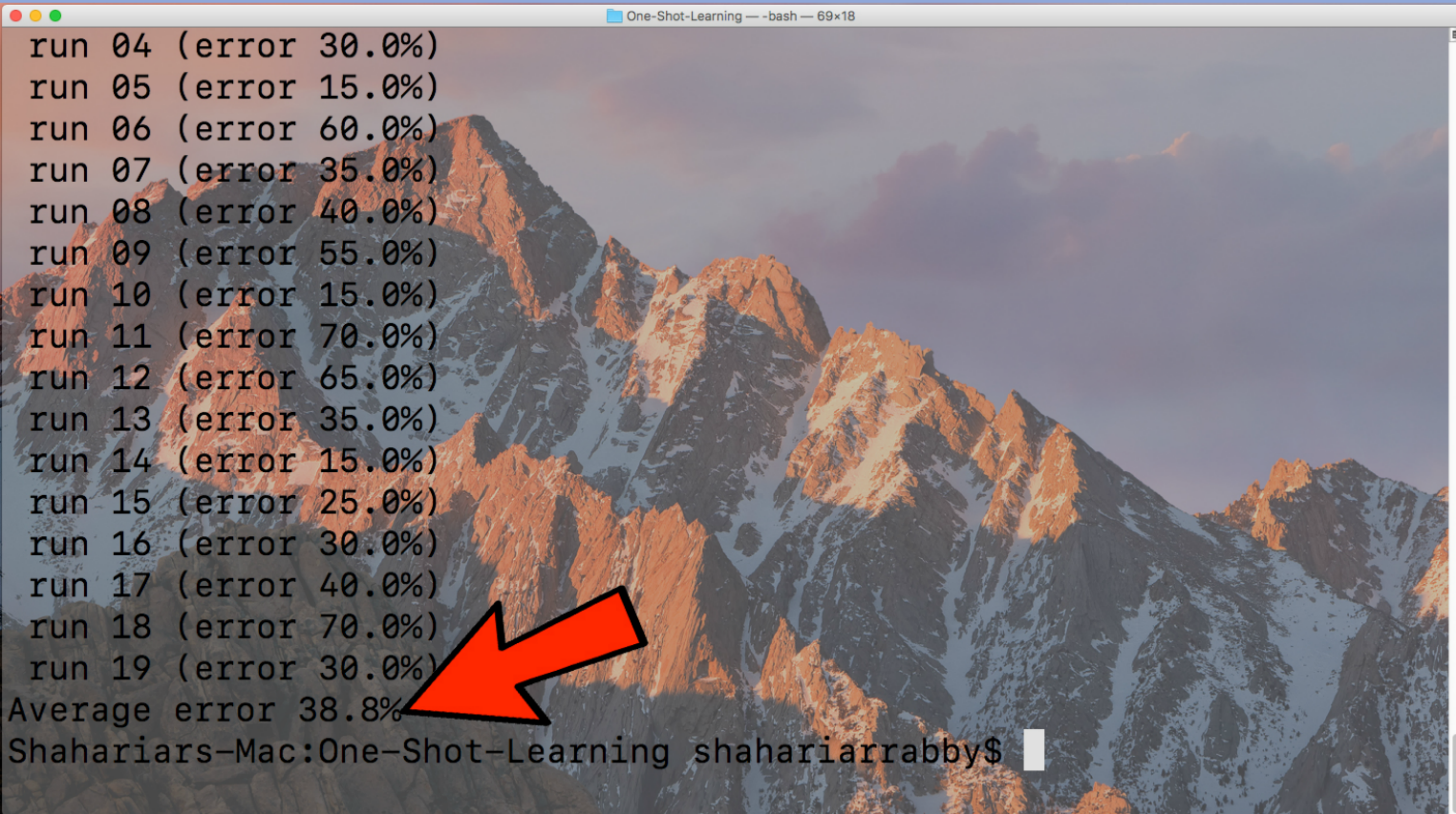
Line 123, Column 1   Tab Size: 4   Python

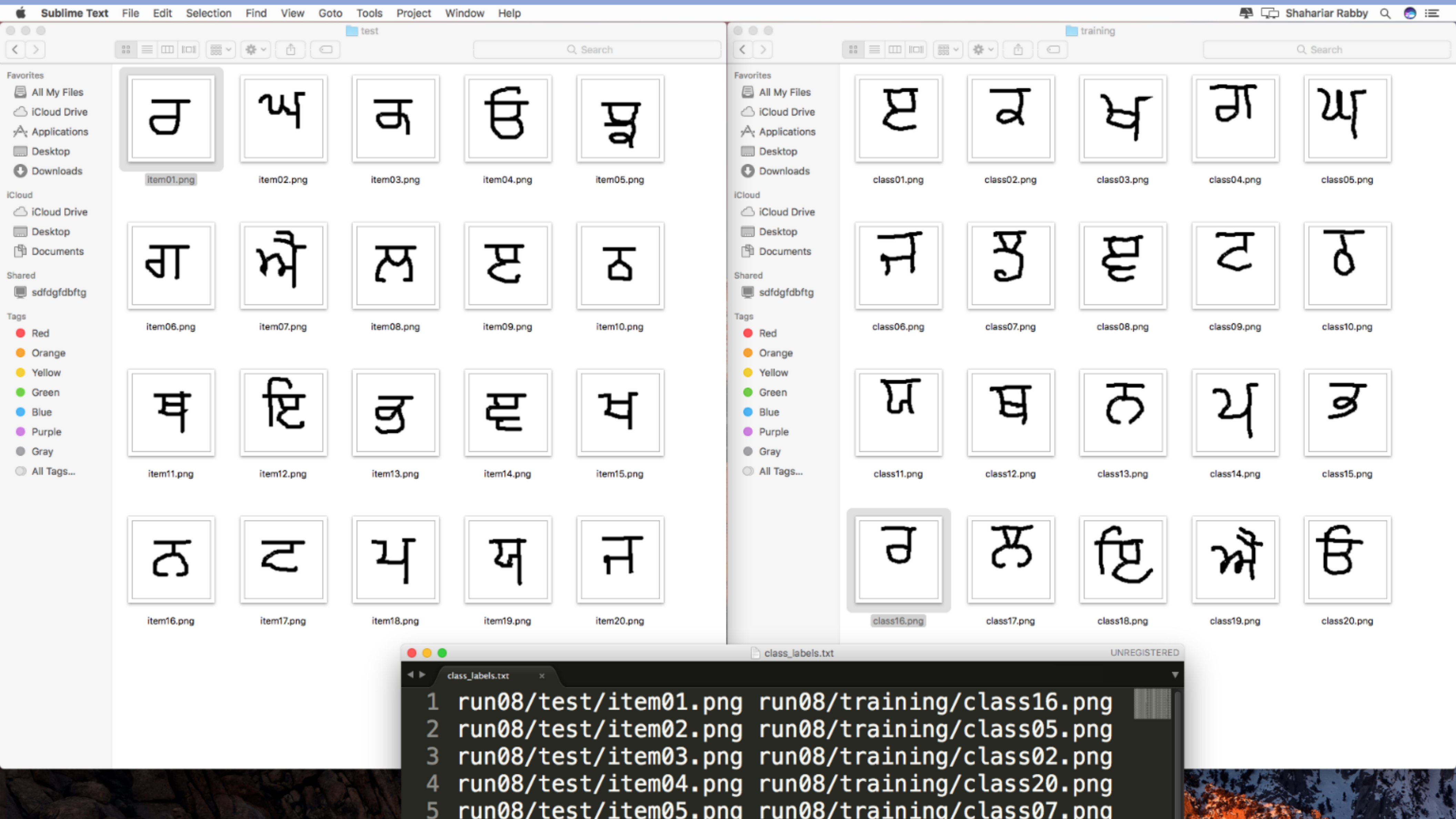
demo_classification.py

```python
    # Load the images (and, if needed, extract features)
    train_items = [f_load(os.path.join(path_to_all_runs, f))
                   for f in train_files]
    test_items = [f_load(os.path.join(path_to_all_runs, f))
                  for f in test_files]

    # Compute cost matrix
    costM = np.zeros((n_test, n_train))
    for i, test_i in enumerate(test_items):
        for j, train_j in enumerate(train_items):
            costM[i, j] = f_cost(test_i, train_j)
    if ftype == 'cost':
        y_hats = np.argmin(costM, axis=1)
    elif ftype == 'score':
        y_hats = np.argmax(costM, axis=1)
    else:
        # This should never be reached due to the assert above
        raise ValueError('Unexpected ftype: {}'.format(ftype))

    # compute the error rate by counting the number of correct predictions
    correct = len([1 for y_hat, answer in zip(y_hats, answers_files)
                   if train_files[y_hat] == answer])
    pcorrect = correct / float(n_test)  # Python 2.x ensure float division
    perror = 1.0 - pcorrect
    return perror * 100
```

```python
    # Load the images (and, if needed, extract features)
    train_items = [f_load(os.path.join(path_to_all_runs, f))
                   for f in train_files]
    test_items = [f_load(os.path.join(path_to_all_runs, f))
                  for f in test_files]

    # Compute cost matrix
    costM = np.zeros((n_test, n_train))
    for i, test_i in enumerate(test_items):
        for j, train_j in enumerate(train_items):
            costM[i, j] = f_cost(test_i, train_j)
    if ftype == 'cost':
        y_hats = np.argmin(costM, axis=1)
    elif ftype == 'score':
        y_hats = np.argmax(costM, axis=1)
    else:
        # This should never be reached due to the assert above
        raise ValueError('Unexpected ftype: {}'.format(ftype))

    # compute the error rate by counting the number of correct predictions
    correct = len([1 for y_hat, answer in zip(y_hats, answers_files)
                   if train_files[y_hat] == answer])
    pcorrect = correct / float(n_test)  # Python 2.x ensure float division
    perror = 1.0 - pcorrect
    return perror * 100
```


Terminal Shell Edit View Window Help
Shahariar Rabby
One-Shot-Learning — python demo_classification.py — 69×18
Shahariars-Mac:One-Shot-Learning shahariarrabby$ python demo_classifi
cation.py
osx
Anaconda3-4.4.0-MacOSX...6_64.pkg
Screenshot
One-Shot-Learning


One-Shot-Learning — -bash — 69×18
run 04 (error 30.0%)
run 05 (error 15.0%)
run 06 (error 60.0%)
run 07 (error 35.0%)
run 08 (error 40.0%)
run 09 (error 55.0%)
run 10 (error 15.0%)
run 11 (error 70.0%)
run 12 (error 65.0%)
run 13 (error 35.0%)
run 14 (error 15.0%)
run 15 (error 25.0%)
run 16 (error 30.0%)
run 17 (error 40.0%)
run 18 (error 70.0%)
run 19 (error 30.0%)
Average error 38.8%
Shahariars-Mac:One-Shot-Learning shahariarrabby$



Sublime Text File Edit Selection Find View Goto Tools Project Window Help
Shahariar Rabby
test
Search
Favorites
All My Files
iCloud Drive
Applications
Desktop
Downloads
iCloud
iCloud Drive
Desktop
Documents
Shared
sdfdgfdbftg
Tags
Red
Orange
Yellow
Green
Blue
Purple
Gray
All Tags...
ਰ
item01.png
ਘ
item02.png
ਕ
item03.png
ਓ
item04.png
ਙ
item05.png
ਗ
item06.png
ਐ
item07.png
ਲ
item08.png
ੲ
item09.png
ਠ
item10.png
ਥ
item11.png
ਇ
item12.png
ਭ
item13.png
ਵ
item14.png
ਖ
item15.png
ਨ
item16.png
ਟ
item17.png
ਪ
item18.png
ਯ
item19.png
ਜ
item20.png
training
Search
Favorites
All My Files
iCloud Drive
Applications
Desktop
Downloads
iCloud
iCloud Drive
Desktop
Documents
Shared
sdfdgfdbftg
Tags
Red
Orange
Yellow
Green
Blue
Purple
Gray
All Tags...
class01.png
class02.png
class03.png
class04.png
class05.png
class06.png
class07.png
class08.png
class09.png
class10.png
class11.png
class12.png
class13.png
class14.png
class15.png
class16.png
class17.png
class18.png
class19.png
class20.png
class_labels.txt
UNREGISTERED
class_labels.txt
1 run08/test/item01.png run08/training/class16.png
2 run08/test/item02.png run08/training/class05.png
3 run08/test/item03.png run08/training/class02.png
4 run08/test/item04.png run08/training/class20.png
5 run08/test/item05.png run08/training/class07.png

Source

Human Level Concept Learning through Probabilistic Program Induction: http://web.mit.edu/cocosci/Papers/Science-2015-Lake-1332-8.pdf

One-Shot Learning with Memory Augmented Neural Networks: https://arxiv.org/pdf/1605.06065v1.pdf

For Code & Papers: https://github.com/shahariarrabby/One-Shot

Thank You

One Shot Learning

Human Level Concept Learning through Probabilistic Program Induction