


# CS 638 / 838 LAB 3

---

Presented by Yuting Liu  
Feb 7, 2017

# OUTLINE



1. Data Set
  2. Code Skeleton
  3. Requirements
  4. Suggestions
- 



next

2

# 01

## D A T A S E T

- Includes 912 images (from [http://www.vision.caltech.edu/Image\\_Datasets/Caltech101/](http://www.vision.caltech.edu/Image_Datasets/Caltech101/))
- Down-sampled to  $128 * 128$
- Classified into 6 categories
- Image files (in three subdirs) are named in format:  
`label_image_XXXX` (4 digits)



Flower



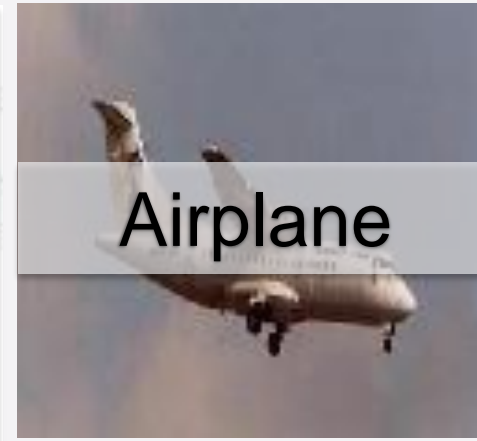
Star Fish



Watch



Grand Piano



Airplane

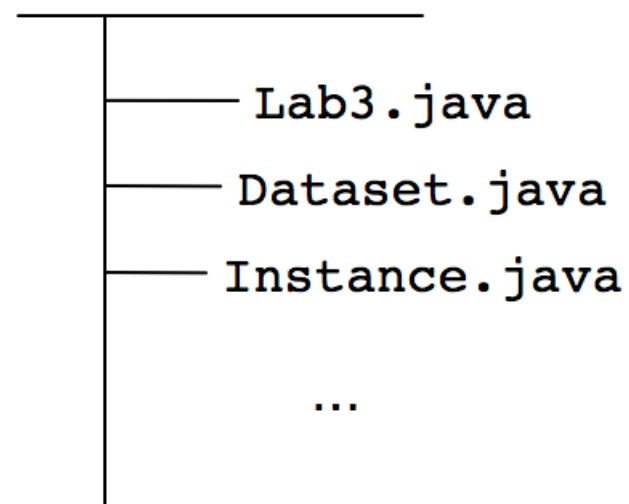


Butterfly

# 02

## CODE SKELETON

- [Code](#) and [data](#) (zipped [here](#)) on line
- Code hierarchy



- Lab3.java (main function)
  - Loads in the image as Instance object
  - Stores all image instances in the dataset
- Dataset.java
  - Stores all image instances as a list in the dataset
- Instance.java
  - Provides the methods to separate RGB channels in the image
  - Provides the method to change image into grey-scale image

# 02

## CODE SKELETON

Lab3.java

- Void main()
- Converts images to (1D) 'fixed length feature vectors'

Instance.java

- BufferedImage image
- String label
- int width, height
- 2D matrix with separate RGB channels
- Gray-scale images; you choose if you wish to use RGB or grey scale
- FOUR 2D matrices in total: R + G + B + Grey
- Each matrix element an int in [0, 255]

Dataset.java

- void add(Instance image)
- List<Instance> getImages()

# 03

## REQUIREMENTS

- See Day 1 ‘course overview’ slides for required network structure
- Only edit Lab3.java (we will load the other provided files during testing – do NOT edit them! We will fix any bugs, announce, and update files)
- Lab3.java should read in UP TO FOUR strings: the name of the *train*, *tune*, and *test* directories plus *imageSize* (*details later*)
- For each of the six categories, we already put the 5<sup>th</sup> example in TUNE and 6<sup>th</sup> in TEST (rest in TRAIN)

# 03

## REQUIREMENTS

- Use DROPOUT (50% of weights)
- Use ReLU for hidden layers and the six output nodes
- How many HUs per layer?
  - Up to you
  - See how fast your code runs, discuss in Piazza
- Use early stopping to reduce overfitting
- Max number of epochs?
  - Whatever is feasible
  - Discuss in Piazza as you run experiments

# 03

## REQUIREMENTS

- Probably test-set accuracy will be poor 😞
- Learning curve (later) will give us sense if more data would help
- Optional Experiment 1: alter the provided TRAIN images to create more examples
  - Shift a little, change colors?, etc
  - Post links to scientific papers on ideas in the literature
- Optional Experiment 2: try more layers



# 03

## WHAT TO OUTPUT

- Your code should output (in plain ASCII) a CONFUSION MATRIX (see below)
- Each TEST SET example increments the count in ONE CELL
- Also report overall testset accuracy

PREDICTED Cat

CORRECT Category

	Flower	Star Fish	Watch	Piano	Airplane	Butterfly
Flower						
Star Fish						
Watch						
Piano						
Airplane						
Butterfly						

# PERCEPTRON TESTSET RESULTS (WITH DROPOUT)

## CORRECT Category

PREDICTED Cat		airplanes	butterfly	flower	grand_piano	starfish	watch	
	airplanes	36	2	1	1	1	1	row SUM = 42
	butterfly	0	6	3	0	3	2	row SUM = 14
	flower	0	2	28	2	4	3	row SUM = 39
	grand_piano	0	0	0	13	0	0	row SUM = 13
	starfish	1	5	2	1	4	2	row SUM = 15
	watch	4	3	3	2	5	38	row SUM = 55
	column SUM	41	18	37	19	17	46	

total examples = 178, errors = 53 (29.8%)

Used FOUR input units per pixel (red + blue + green + grey)

Trained SIX perceptrons, **100%** TRAIN accuracy after about 50 epochs  
(used early stopping)

Trained EACH as a BINARY CLASSIFIER (eg, *watch* vs. *not\_watch*)

But when PREDICTING used perceptron with largest weighted sum

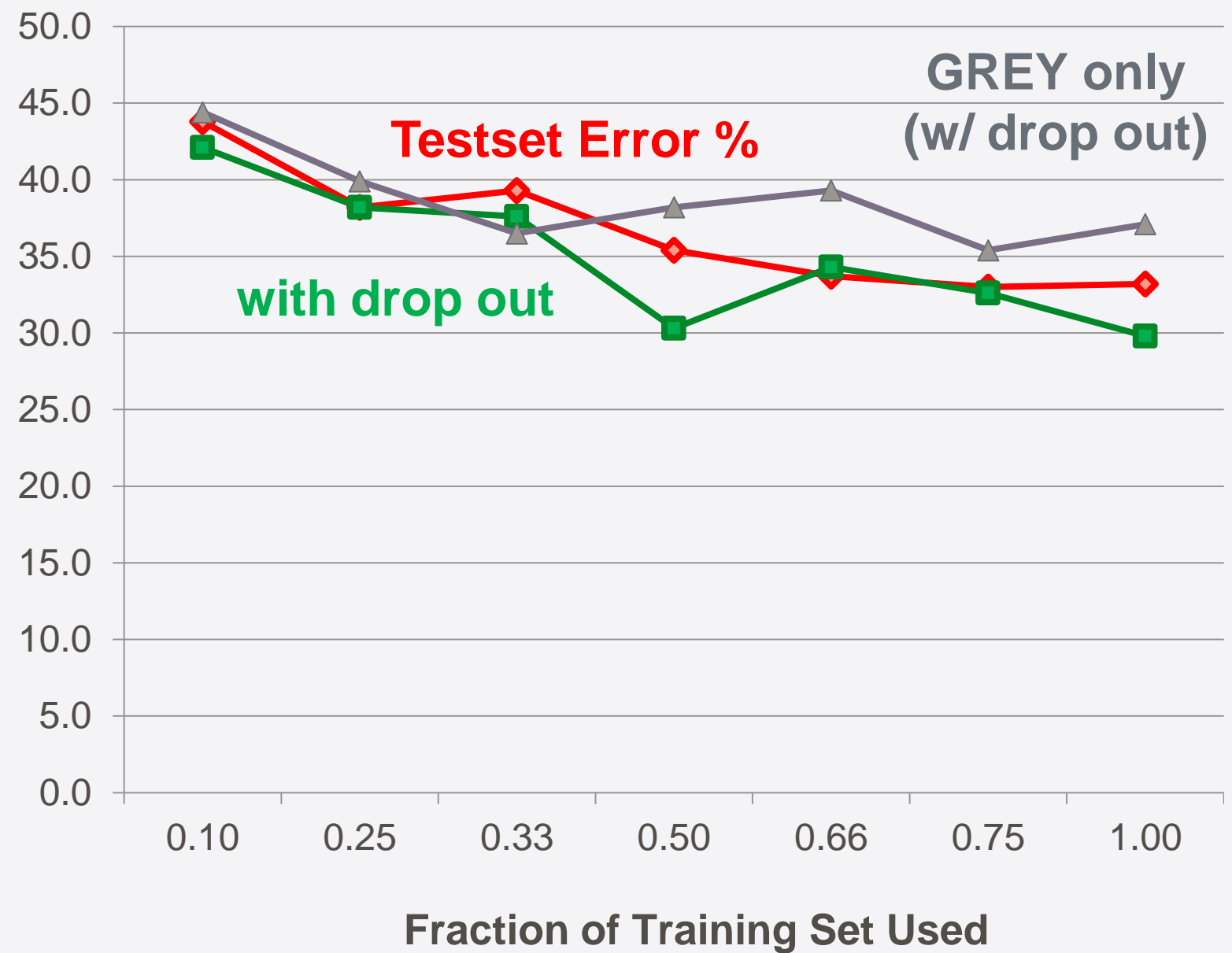
# 03

## EXPERIMENT

- Create and discuss a **learning curve**  
**Plot TEST and maybe TUNE ERROR RATES as function of # of training examples**
- Train on 25%, 50%, 75%, and 100% of the train examples (if your code is fast enough, create more points on the learning curve)
- Use SAME TUNE AND TEST for all points on learning curve (if feasible, multiple times sample the trainset and plot mean)
- The code you submit for our testing should use 100% of the training examples

03

# PERCEPTRON LEARNING CURVE



# 03

## I M A G E   S I Z E

- Raw images are 128x128
- We provide code for re-sizing
- Some **perceptron** test-set error rates

8x8	41%	(46% grey only)
16x16	33%	(39%)
32x32	31%	(36%)
64x64	31%	(35%)
128x128	30%	(33%)

# 03

## ONE-LAYER HIDDEN UNITS

- Used 250 HUs, 1000 epochs, dropout wgts
- Not doing as well as perceptrons; why?

8x8: 48% error rate on testset

16x16: 58% (46% GREY only)

32x32: (47% GREY only)

- Training slow; still debugging
  - will post more stats later
  - will start on convolution-net soon

# 04

## SUGGESTIONS

- You might want to try using your Lab 1 and/or Lab 2 code first!
  - get everything to work, then add layers
  - nice 'baseline' system to evaluate value of DEEP
- First get training to work and report errors
- Next create confusion matrix on train set
- After that create confusion matrices on tune and test sets (use test=train to debug)
- Add 'drop out' next
- Then incorporate early-stopping code
- Finally allow using a fraction of TRAIN to generate the 'learning curve' (permute, then copy first  $N$  items)

# 04

## D R O P O U T   T I P S

1. Need to remember which weights are 'dropped out' during forward prop, so they are also ignored during back prop  

```
Boolean[] dropOutMask = (dropoutRate > 0.0 ? new Boolean[inputVectorSize] : null);
```
2. Remember on TUNE and TEST examples, to multiply all weights by  $(1 - dropoutRate)$  so average weighted sum is the same as during training
3. I never dropped out the weight rep'ing the BIAS (so didn't multiply by  $1 - dropoutRate$ )
4. Ideally would do *multiple measurements* per plotted point since random numbers used



Enjoy it! 😊

