

Assignment 4: Deep Convolutional Neural Networks

cs543/ece549 Spring 2016

Due date: Wednesday, May 4,
11:59:59PM

Prepared with the help of Chih-Hui Ho

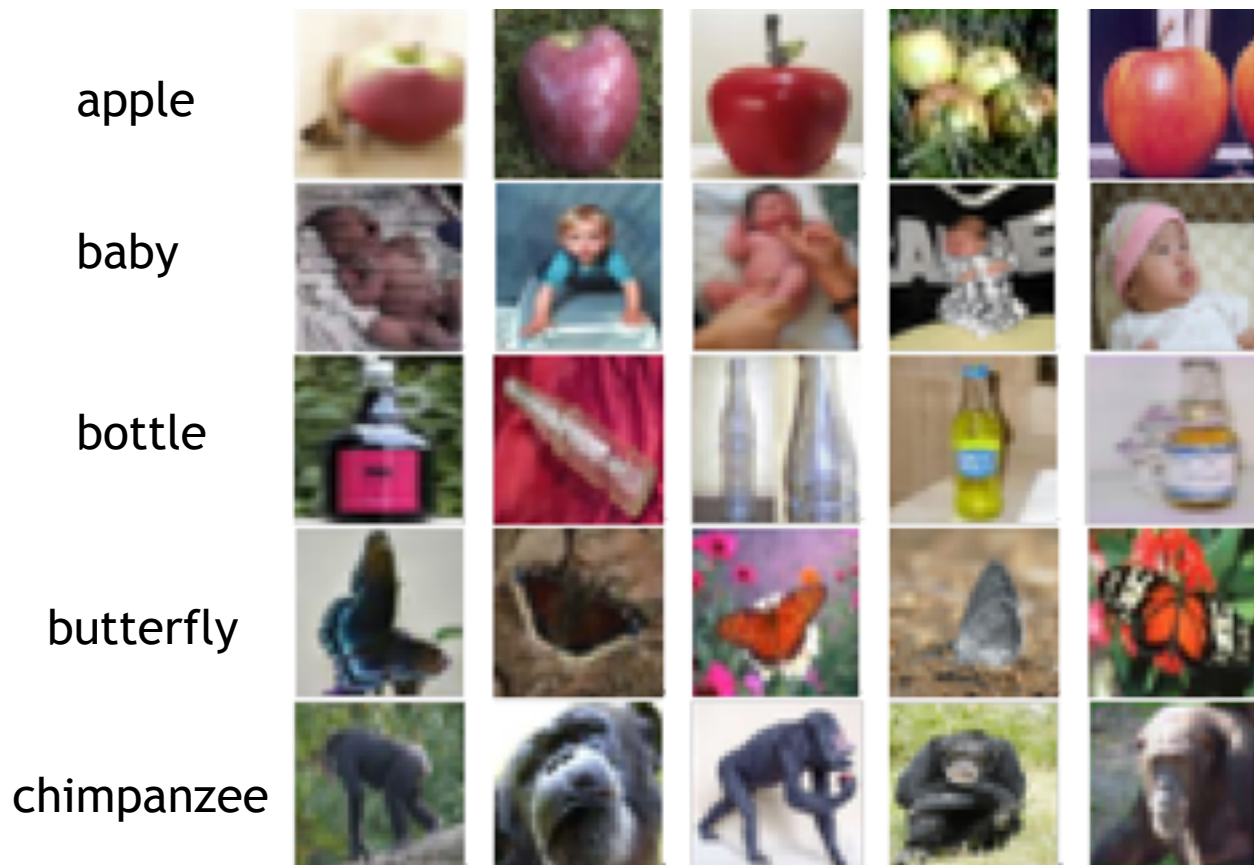
Platform

- [Kaggle in class](#)
- Create an account
- Click on [invitation](#)
- Then you will be added to the competition and will see the detailed instructions page

Goal

- Understand basic convolutional neural networks and build your own deep learning architecture
- Train the network and use it to predict labels for new test images
- Understand the function of each layer and the hyperparameters in the code

Dataset -- Cifar



Dataset -- Cifar

- Dataset: [Cifar100](#)
- Total 100 classes
 - Note that we use “fine” labels in this MP
- Image size 32x32x3
- Original dataset split
 - Five training batches and one validation batch
 - Each batch contains 10K images
- Our split
 - 45K training, 5K validation, 10K testing (labels withheld)
 - You will need to upload your prediction csv file to Kaggle to find out your accuracy

Toolbox

- Open source toolbox: [matconvnet](#)
- **You do not have to download matconvnet by yourself**, we have packed everything in the zip file.

Steps to run the code

- Download the [zip file](#)

File Name	Available Formats
computer_vision_MP	.zip (25.34 mb)

Steps to run the code

- unzip the package
- cd computer_vision_MP
- run setup.m every time you start Matlab
- If GPU is not supported
run setup or setup('useGPU',false)

```
>> setup  
>> setup('useGPU',false);
```
- If GPU is supported
run setup or setup('useGPU',true)

```
>> setup('useGPU',true);
```


Steps to run the code--setup

- if GPU is supported, change the useGpu parameter to "true" in cnn_cifar.m

```
% use the GPU to train  
opts.train.useGpu = false ;
```

- Matconvnet GPU support [reference](#)

MATLAB version	Release	CUDA Devkit
8.2	2013b	5.5
8.3	2014a	5.5
8.4	2014b	6.0
8.6	2015b	Latest(>7.0)

Example Dataset: MNIST

- Dataset: [mnist data set](#)
- This dataset is only for demo use, not for the competition or MP grading
- Contains digits 0~9, total of 10 classes
- 60K training images and 10K validation images
- Image size 28x28x1

Example Dataset: MNIST

0 0 0 0 0

1 1 1 1 1

2 2 2 2 2

3 3 3 3 3

4 4 4 4 4

5 5 5 5 5


6 6 6 6 6

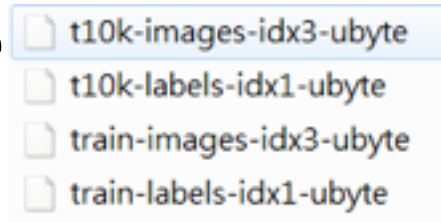
7 7 7 7 7

8 8 8 8 8

9 9 9 9 9

Steps to run the code--Mnist

- Demo code for Mnist
- Run Mnist first. The training is quick
- `cd computer_vision_MP/code`
- Run `cnn_mnist.m`  `cnn_mnist`
- Program will start downloading dataset, store it in `code\mnist_data\mnist`
- Dataset will be organized into `imdb.mat`, stored in `code\mnist_data\mnist-baseline`



Steps to run the code--Mnist

- imdb.mat contains two struct, images and meta
- Try to load imdb.mat and understand it



- images
- data
 - **Labels**
 - Digits from 0~9
 - **Set**
 - 1 for training , 2 for validation and 3 for test image



- meta
- **Sets**
 - 1 for training , 2 for validation and 3 for test image
 - **Clas**

Steps to run the code--Mnist

- The data will be downloaded only the first time
- After imdb is generated, mnist_groundtruth csv file will also be generated
- mnist_groundtruth saves all the validation image labels

Steps to run the code--Mnist

- Start training and you will see this:

```
layerl      1l      2l      3l      4l      5l      6l      7l      8l
type1       cnvl    mpool1  cnvl    mpool1  cnvl    relu1  cnvl    sftml1
support1    5x5l    2x2l    5x5l    2x2l    4x4l    1x1l    1x1l    1x1l
stridel      1l      2l      1l      2l      1l      1l      1l      1l
pad1         0l      0l      0l      0l      0l      0l      0l      0l
out dim1     20l     20l     50l     50l     500l    500l    10l     10l
filt dim1    1l      n/a1    20l     n/a1    50l     n/a1    500l    n/a1
rec. field1  5l       6l      14l     16l     28l     28l     28l     28l
c/g net KB1  2/0l     0/0l    98/0l   0/0l    1564/0l 0/0l    20/0l   0/0l
total network CPU/GPU memory: 1.6/0 MB
learning rate changed (0.000000 --> 0.001000): resetting momentum
training: epoch 01: processing batch 1 of 600 ... 0.19 s (539.3 images/s) err 91.0 err5 51.0
training: epoch 01: processing batch 2 of 600 ... 0.15 s (680.5 images/s) err 90.0 err5 49.5
training: epoch 01: processing batch 3 of 600 ... 0.15 s (646.4 images/s) err 90.0 err5 45.3
```

Steps to run the code--Mnist

- Format for displaying network structure in MP report:

Layer	type	Input	Filter	stride	pad	Output
Layer 1	Conv	28x28x1	5x5x1x20	1	0	24x24x20
Layer 2	Pool (max)	24x24x20	2x2	2	0	12x12x20
Layer 3	conv	12x12x20	5x5x20x50	1	0	8x8x50
Layer 4	Pool (max)	8x8x50	2x2	2	0	4x4x50
Layer 5	Conv3	4x4x50	4x4x50x500	1	0	1x1x500
Layer 6	Relu1	1x1x500	Max(0,x)	1	0	1x1x500
Layer 7	Conv4	1x1x500	1x1x500x10	1	0	1x1x10
Layer 8	Softmaxloss	1x1x10		1	0	1x10

Steps to run the code--Mnist

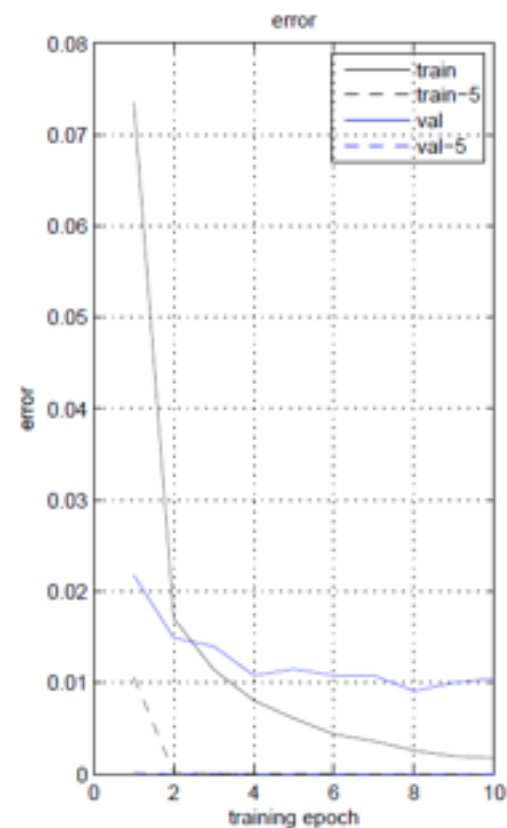
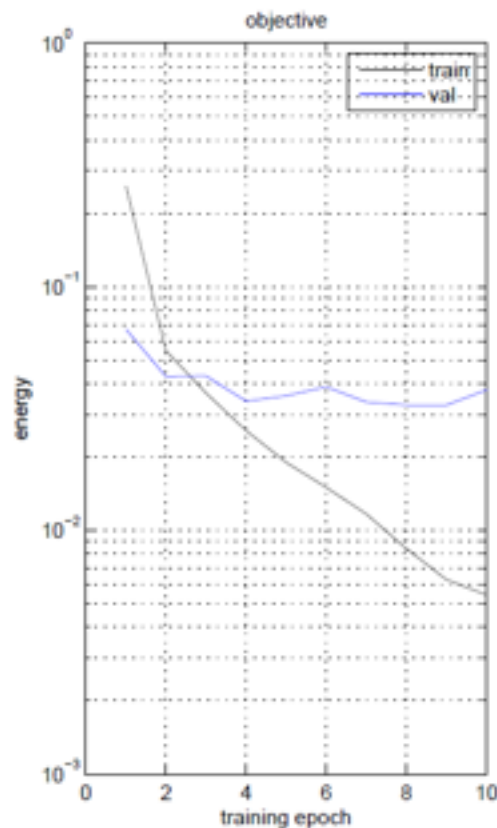
- Training details

Training iteration	Current batch /total batch	Batch time	Speed	Top 1 error	Top5 error
training: epoch 01: processing batch	1 of 600 ...	0.17 s (593.4 images/s)		err 85.0	err5 40.0
training: epoch 01: processing batch	2 of 600 ...	0.17 s (601.0 images/s)		err 88.5	err5 40.5
training: epoch 01: processing batch	3 of 600 ...	0.19 s (531.1 images/s)		err 88.7	err5 39.3

- How do we get 600 batches? We set mini-batch size 100 and there are 60K images
- Top 1 error: Checks if the target label is the top prediction (the one with the highest probability).
- Top 5 error: Checks if the target label is one of your top 5 predictions (the five with the highest probabilities).

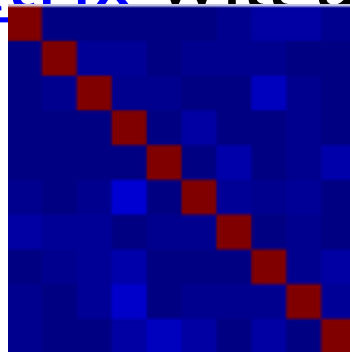
Steps to run the code--Mnist

- During training, a plot will be shown:



Steps to run the code--Mnist

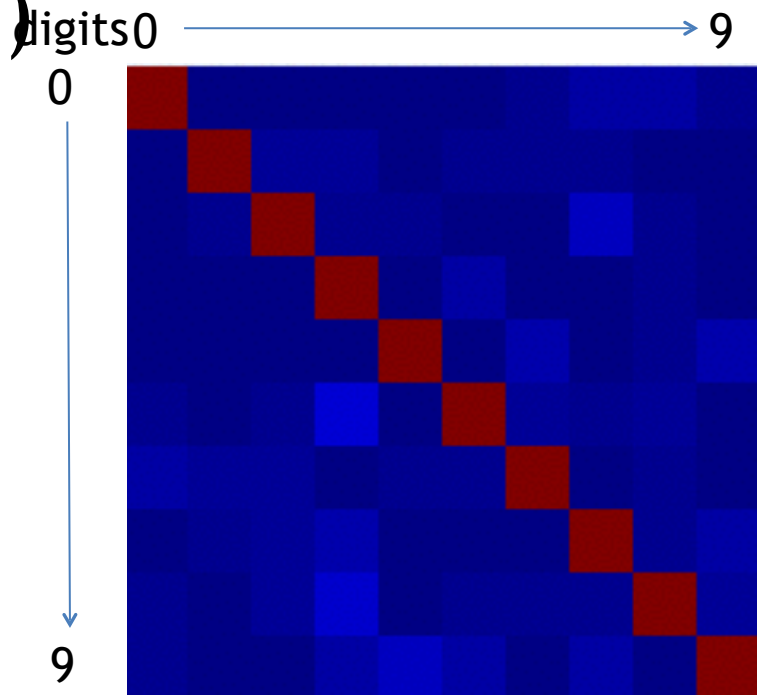
- After 10 epochs, the training is done
- mnist_prediction csv file, which saves all the prediction labels, will also be generated
- The error between ground truth and prediction will be the same as Err in last epoch
- A [confusion matrix](#) will also be generated



mnist_groundTruth
mnist_prediction

Confusion Matrix

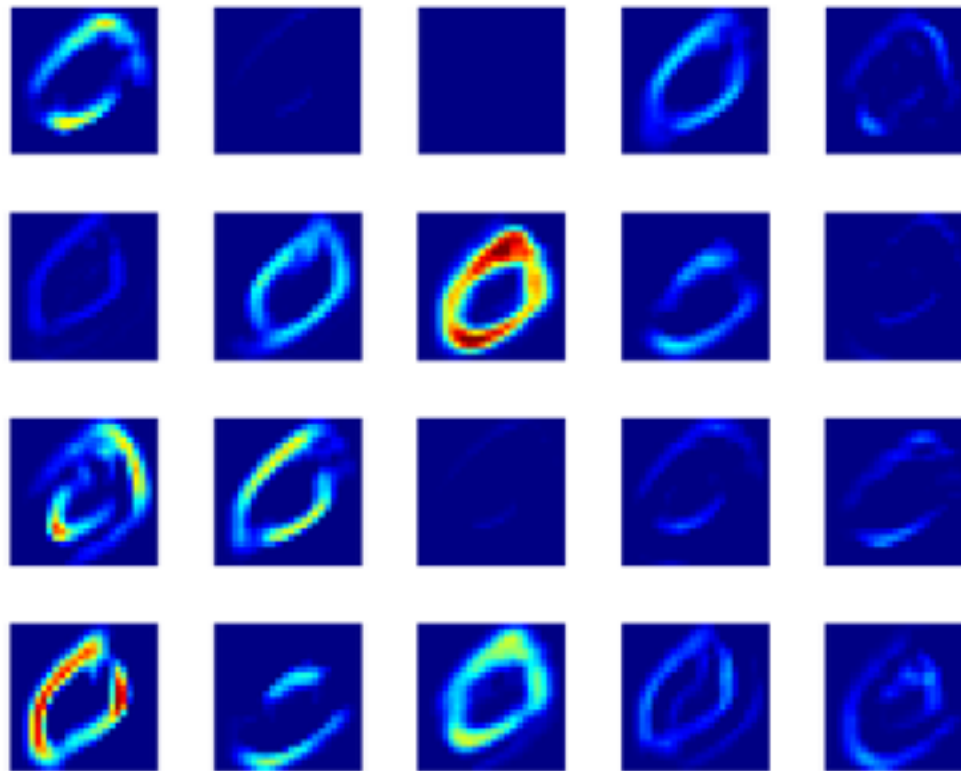
- `ConfusionMatrix(i,j)`
à probability of recognizing class i as class j
(proportion of examples from class i
misclassified as class j)
- Lower error
→ diagonal matrix



Net Dissection



- See `net_dissection_mnist.m`

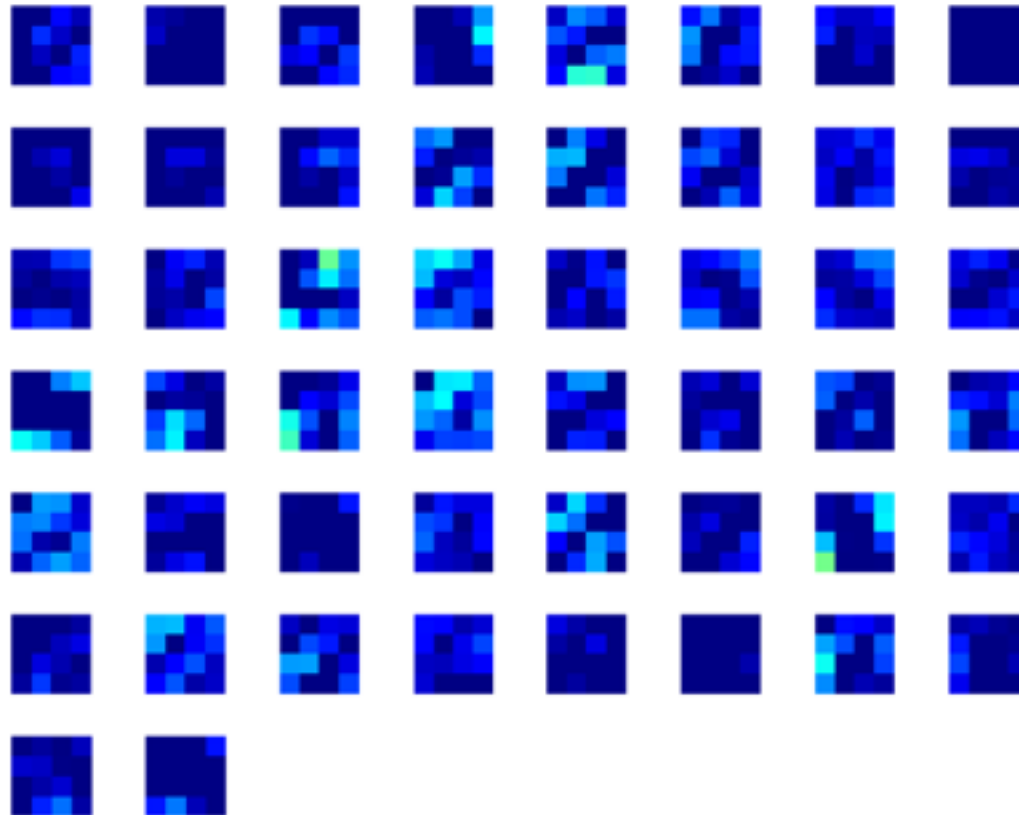


First layer output

Net Dissection



- See `net_dissection_mnist.m`



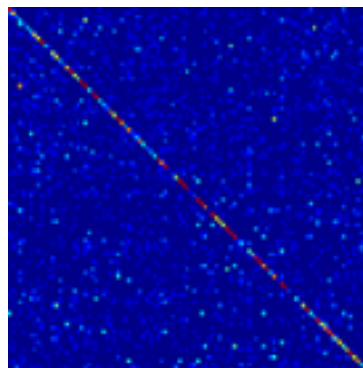
Last layer output

Steps to run the code--Cifar

- Let's start Cifar 100
- Steps are exactly the same as Mnist
- `run_cnn_cifar('fine')`
- **Except we have provide modified imdb**
- Modified imdb has 45K training images, 5K validation images, and 10K test images
- And you have to **build your net architecture**
→ **Modify `cnn_cifar.m`**
- How ?
- We provide an example model

Go through model 1

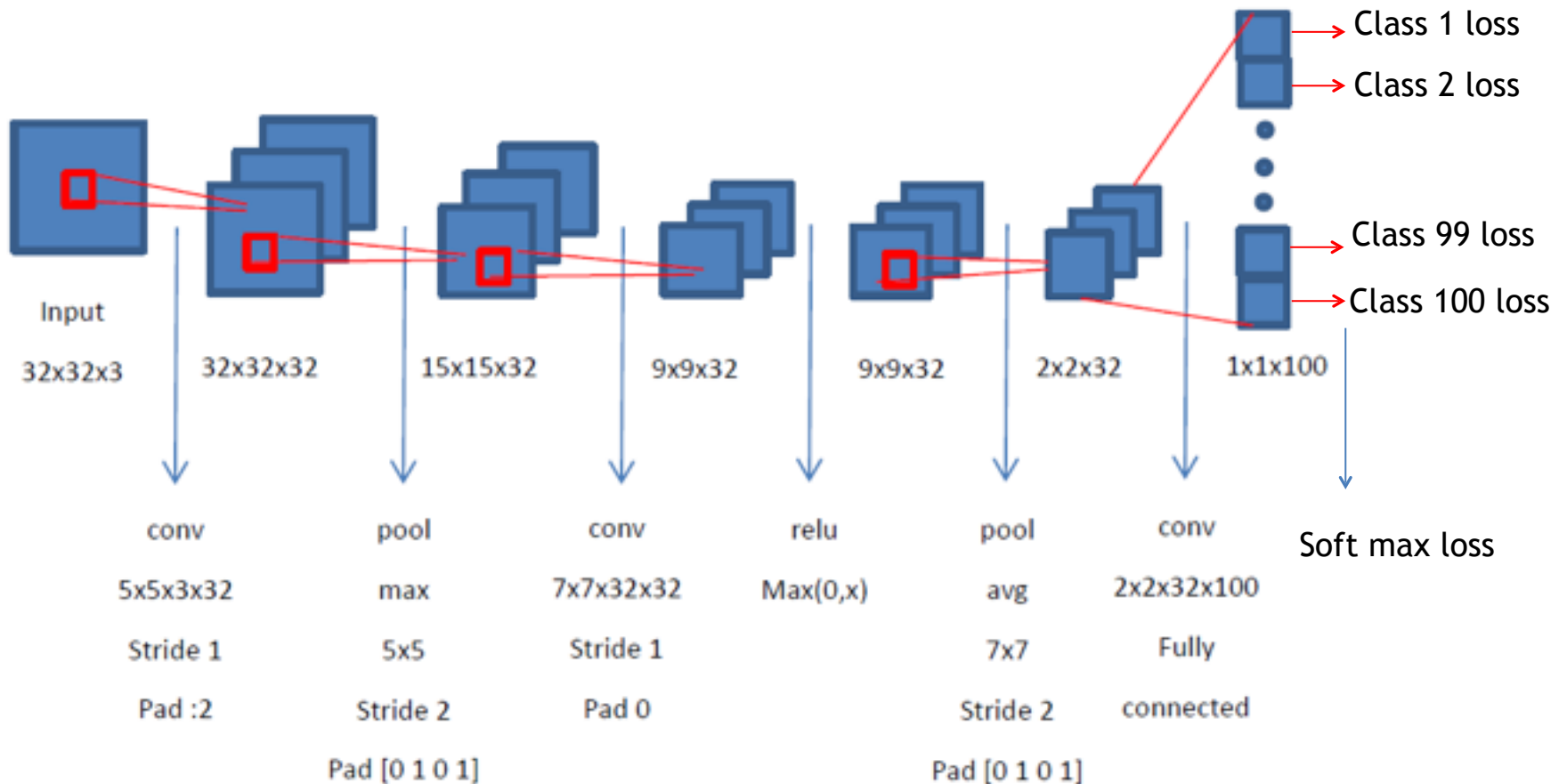
- Accuracy 25 %
- Confusion matrix
- Layers:



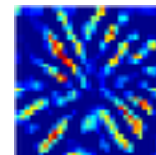
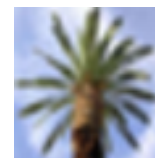
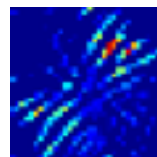
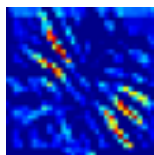
```
>> cnn_cifar('fine')
```

layer1	11	21	31	41	51	61	71
type1	cnv1	mpool1	cnv1	relu1	apool1	cnv1	sftml1
support1	5x51	5x51	7x71	1x11	7x71	2x21	1x11
stride1	11	21	11	11	21	11	11
pad1	2 0,1x0,11		01	0 0,1x0,11		01	01
out dim1	321	321	321	321	321	1001	1001
filt dim1	31	n/a1	321	n/a1	n/a1	321	n/a1

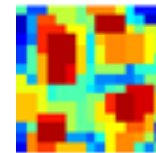
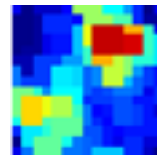
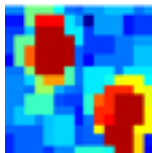
Block diagram



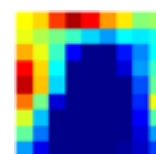
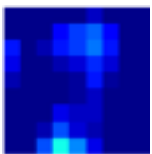
Input	output	filter
32x32x3	32x32x32	Conv 5x5x3x32 Stride 1 Pad 2
32x32x32	15x15x32	Pool Max 5x5 Stride 2 Pad [0 1 0 1]
15x15x32	9x9x32	Conv 7x7x32x32 Stride 1 Pad 0
9x9x32	9x9x32	Relu Max(0,x)
9x9x32	2x2x32	Pool Avg 7x7 Stride 2 Pad [0 1 0 1]
2x2x32	1x1x100	conv 2x2x32x100 Fully connect
1x1x100	1x100	Softmaxloss



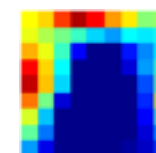
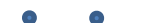
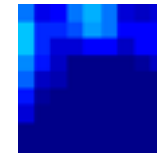
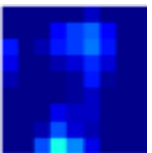
Size 32x32



Size 15x15



Size 9x9

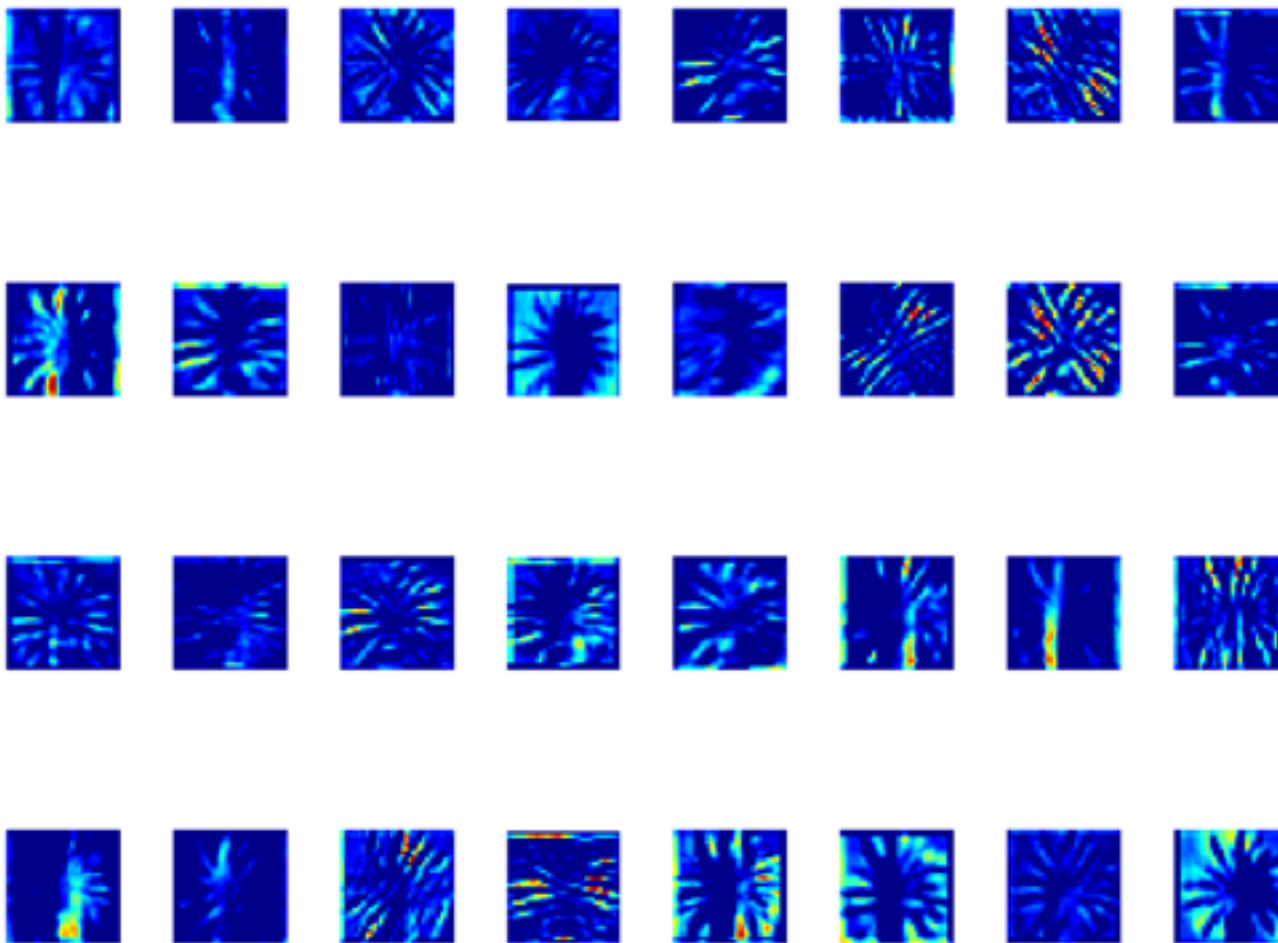


Size 9x9



Size 2x2

First layer output

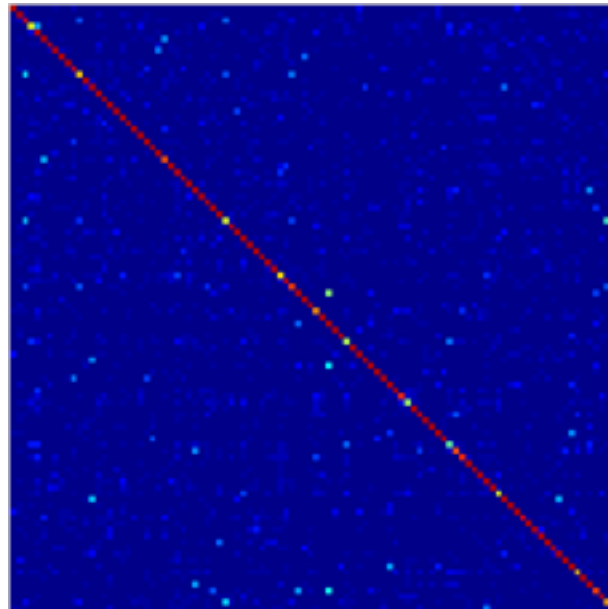


Last conv. layer output



How well should you be able to do?

Our best model



Accuracy 61 %

Computational requirements

- Running times and memory requirements on an Intel Core i5-3210M CPU 2.5GHz, CPU-only mode:
 - MNIST reference model (provided): 10 min to train, 620MB memory
 - CIFAR reference model (provided): 70 min to train, 1200MB
 - Our best CIFAR model: 22 hours to train, 2700MB

Submission

- You need to submit your prediction results to Kaggle and upload code and report to Compass as usual
- Refer to the [Kaggle page](#) for detailed instructions