CV HW #3: Deep Learning

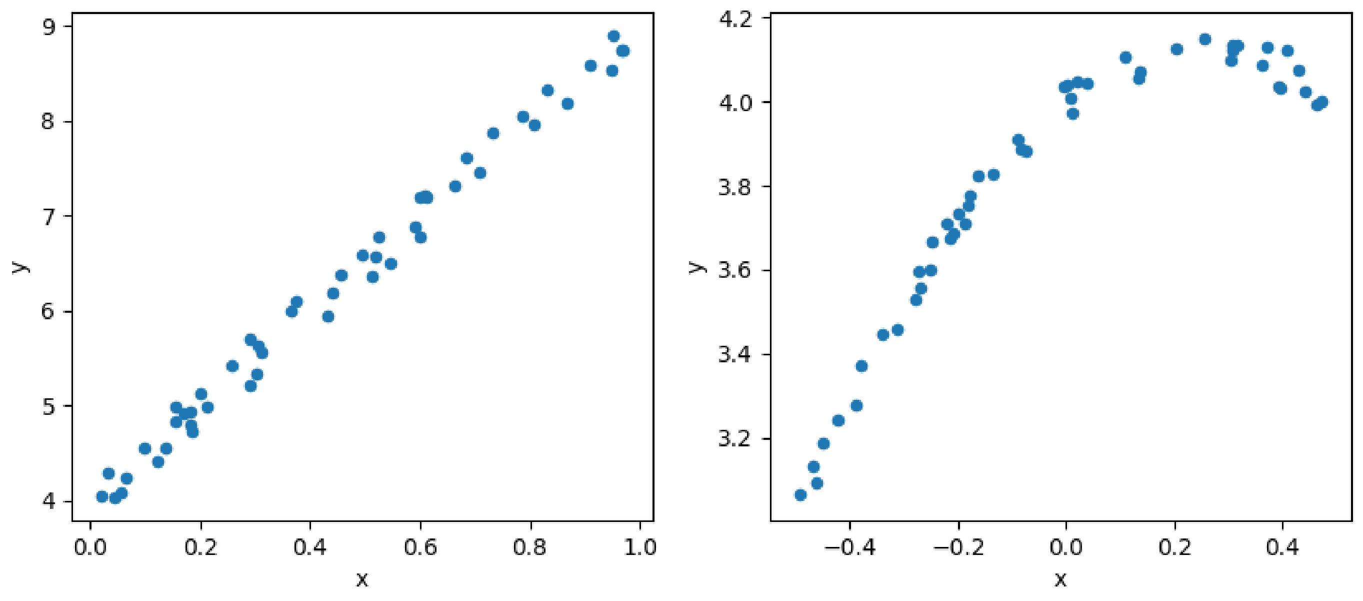
Deadline: 12/11 11:59pm

Problem A: Line Fitting

Download two csv files `pA1.csv` and `pA2.csv` using:

```
wget -nc 140.114.76.113:8000/pA1.csv
wget -nc 140.114.76.113:8000/pA2.csv
```

You are asked to find the curve that fits the data using Stochastic Gradient Descent with a deep learning framework (PyTorch, Keras, etc). A1, A2 should be written in one Colab Notebook.

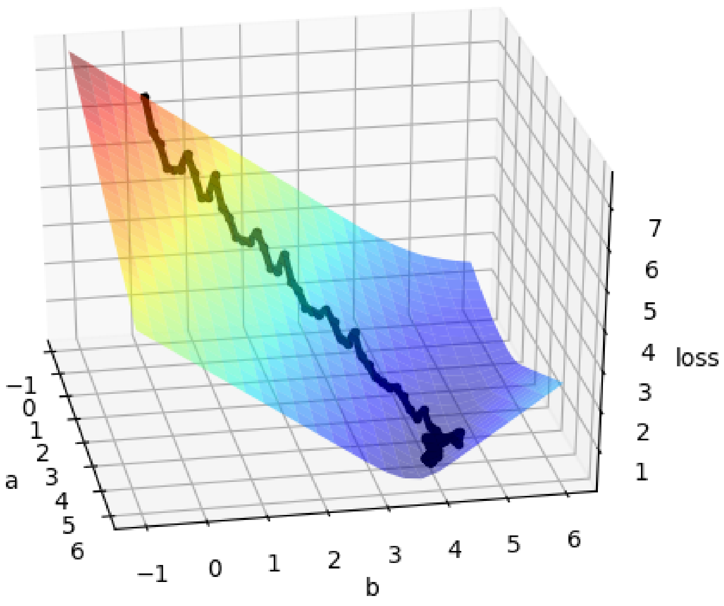


Left is `pA1.csv` and right is `pA2.csv` .

A1

Assume that the curve is $y = f(x; a, b) = ax + b$. You are asked to find the a, b that makes the curve best fit the data `pA1.csv` . Requirements:

1. A PyTorch code that outputs a and b. You get the points only if predicted a, b and ground truth $\hat{a}, \hat{b}$ satisfies $|\hat{a} - a| \leq 0.2$ and $|\hat{b} - b| \leq 0.2$. (8 points)
2. A 3D figure that visualizes the loss function (of whole dataset) against parameters space. The figure may differ with different setting. (7 points)



There's a sample code here (https://colab.research.google.com/drive/1GZ_oX3PPwoDOamx0_XGbQHyt9ag1x5HM). Feel free to modify it. Hyperparameters(loss function, optimizer, batch size, epoch, ...) are not restricted to the one used in sample code.

A2

You are asked to fit the data `pA2.csv` using following model:

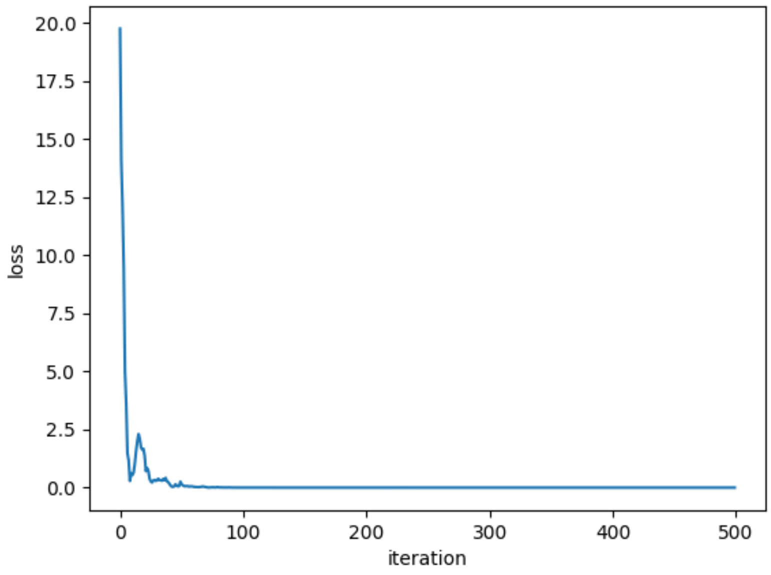
$$y = f(x; \mathbf{w}) = w_0 x^2 + w_1 x + w_2$$

where $\mathbf{w}$ are parameters.

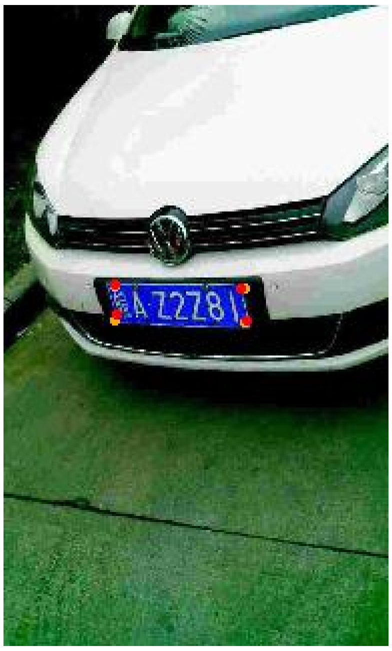
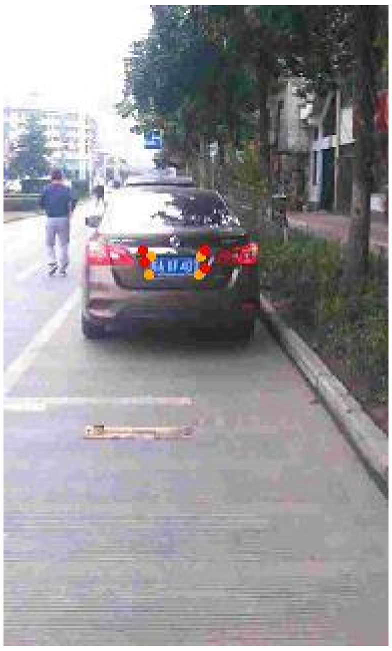
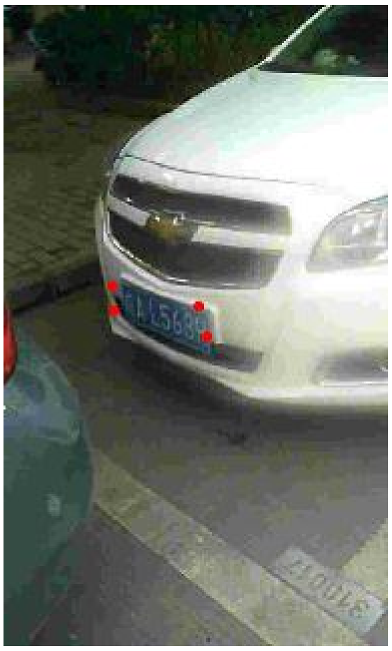
Your code outputs w_0, w_1, w_2. Similar to problem A1, you get the points only if predicted $\mathbf{w}$ and ground truth $\hat{\mathbf{w}}$ satisfies $|w_0 - \hat{w}_0| < 0.2$, $|w_1 - \hat{w}_1| < 0.2$ and $|w_2 - \hat{w}_2| < 0.2$. Requirements:

1. Use `nn.Linear` (PyTorch) / `Dense` (Keras or Tensorflow) to accomplish the task. (8 points)

2. A plot of loss against iteration or epoch. (7 points)



Problem B: License Plate Localization

Train	Valid	Test
		

Ground-truth are drawn in orange. Prediction are drawn in red.

Overview

For each image, there is one license plate. You are asked to localize the 4 corners of the license plate. That is, predict the (x, y) of each corner, 8 values in total. To reduce difficulties, you can fill in the blank of this reference code (https://colab.research.google.com/drive/1D4TYD73crnnyfvWa9-N-lpnsJlvRbhlC) to achieve the baseline.

Data

To download the data (317MB):

```
wget -nc 140.114.76.113:8000/ccpd6000.zip
```

SHA256 checksum:
`977d7124a53e565c3f2b371a871ee04ebbe572f07deb0b38c5548ddaae0cb2c9`

Data is organized as:

```
ccpd6000/
    train_images/
    test_images/
    train.csv
    sample.csv
```

There are 3000 images with annotation for training, 3000 images without label for testing. All images are taken from CCPD (https://github.com/detectRecog/CCPD).

Each row in `train.csv` has following fields:

1. `name` specifies the name of the image, full path is `ccpd6000/train_images/<name>`
2. `BR_x` , `BR_y` is the position of bottom-right corner
3. `BL_x` , `BL_y` is the position of bottom-left corner
4. `TL_x` , `TL_y` is the position of top-left corner
5. `TR_x` , `TR_y` is the position of top-right corner

The origin is at the top-left of the image.

`sample.csv` serves as a sample submission. Your submission should have the same format as `sample.csv` . Note that `name` is sorted in alphabetical order.

Evaluation

The metric is the root mean-square error between the predicted locations and the ground-truth locations of the 3000 testing images:

$$RMSE = \sqrt{\frac{1}{4N}\sum_{i=1}^{N}\sum_{j=1}^{4}\|\mathbf{p}_i^j - \hat{\mathbf{p}}_i^j\|^2}$$

where:

N is the number of images,
j is the index of the corner,
$\mathbf{p}_i^j$ is the predicted location (x, y) of the j-th corner of image i.

$\hat{\mathbf{p}}_i^j$ is the ground-truth location (x, y) of the j-th corner of image i.

To evaluate your prediction `test.csv` , use `curl` to POST the file to the server:

```
curl -F "file=@test.csv" -X POST 140.114.76.113:5000/cs6550 -i
```

If nothing goes wrong, you will get a dictionary containing the $RMSE$ metric. If you get status code 500 (Internal Server Error), please check your submission first. If you don't think it's your problem, please contact TA (amoshuangyc@gmail.com (mailto:amoshuangyc@gmail.com)) or leave a message in iLMS Forums.

Scoring

Baseline (40 points)

1. Training & Validation (10 points)
2. Visualization of 25 training samples, 25 validation samples every epoch during training. (10 points)
3. Overlay training losses and validation losses in the same figure against step or epoch. (10 points)
4. Testing and $RMSE \leq 35.0$ (10 points)

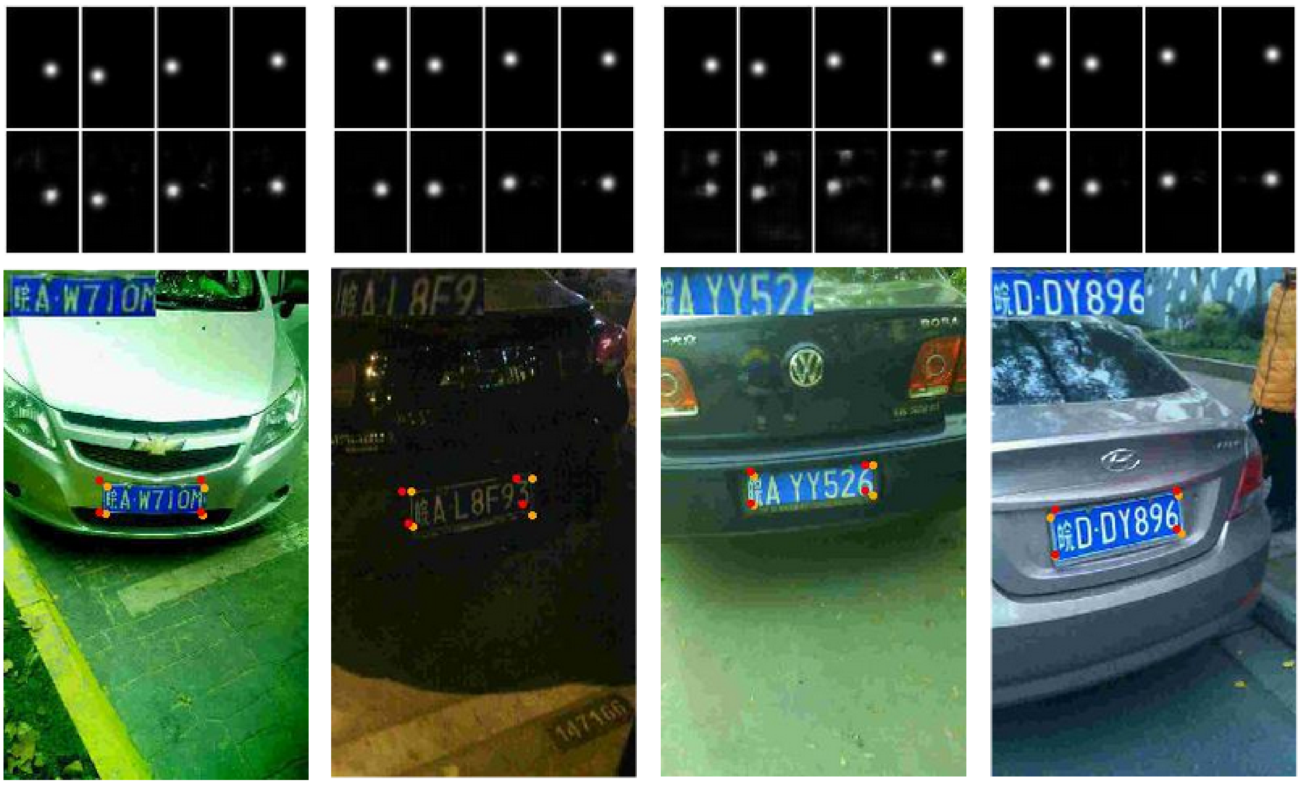
Improvement (20 points)

$RMSE \leq 20.0$

Possible ways:

1. LR(learning rate) decay or lower LR.
2. Train longer (typically until the validation loss is converged).
3. Use deeper model, like ResNet18, to extract features.
4. Different optimizer, loss, etc.
5. Data augmentation.
6. Auxiliary task, like segmentation.

7. Regress heatmaps instead of values.



Bonus (10 points)

$RMSE \leq 15.0$

You don’t need to copy the code many times to get improvement points and bonus points. You get the points automatically if your $RMSE$ reach the criteria.

Report (10 points)

Describe the problems/difficulties you have faced in this homework. The report `report.pdf` should be 2 pages at most.

Misc.

The structure of the turned-in file `hw3_107062566.zip` should be:

```
hw3/
    pA.ipynb
    pB.ipynb
    report.pdf
    problem.pdf
```

The zip file should be less than 5MB. The `*.ipynb` is the Colab Notebook downloaded as `ipynb`.

The notebooks will be uploaded to Colab and executed by pressing “Runtime/Run All”. If your code is not executable, you will get no point. Additionally, following conditions also make you 0 point:

1. Use external data other than the given data.
2. Code is not written by yourself.
3. Plagiarize.
4. Lookup the ground truth in the original dataset.
5. Label the images manually.
6. DoS attack the server.

When using Colab, remember to change “Runtime Type” to “GPU” to accelerate training. Typically the model parameters are initialized randomly, therefore the result may not be the same every time. It’s your responsibility to make your code reproducible (by fixing seed or by other methods)