**Week 5 Work Summary by Eugene Lin and Qinghao Meng:**

Part 1: Problem Overview:

In the last few weeks, we have been trying to apply the L2 variant fastmap algorithm to the face images, and then doing face detection with CNN. We applied celebA dataset from the Multimedia Library of the Chinese University of Hong Kong. The dataset contains about 200,000 face images in the complex background and each image contains only a single face. The dimension of each image is 218 \* 178 \* 3(RGB). In this problem, we randomly picked 2,000 images from the original dataset.

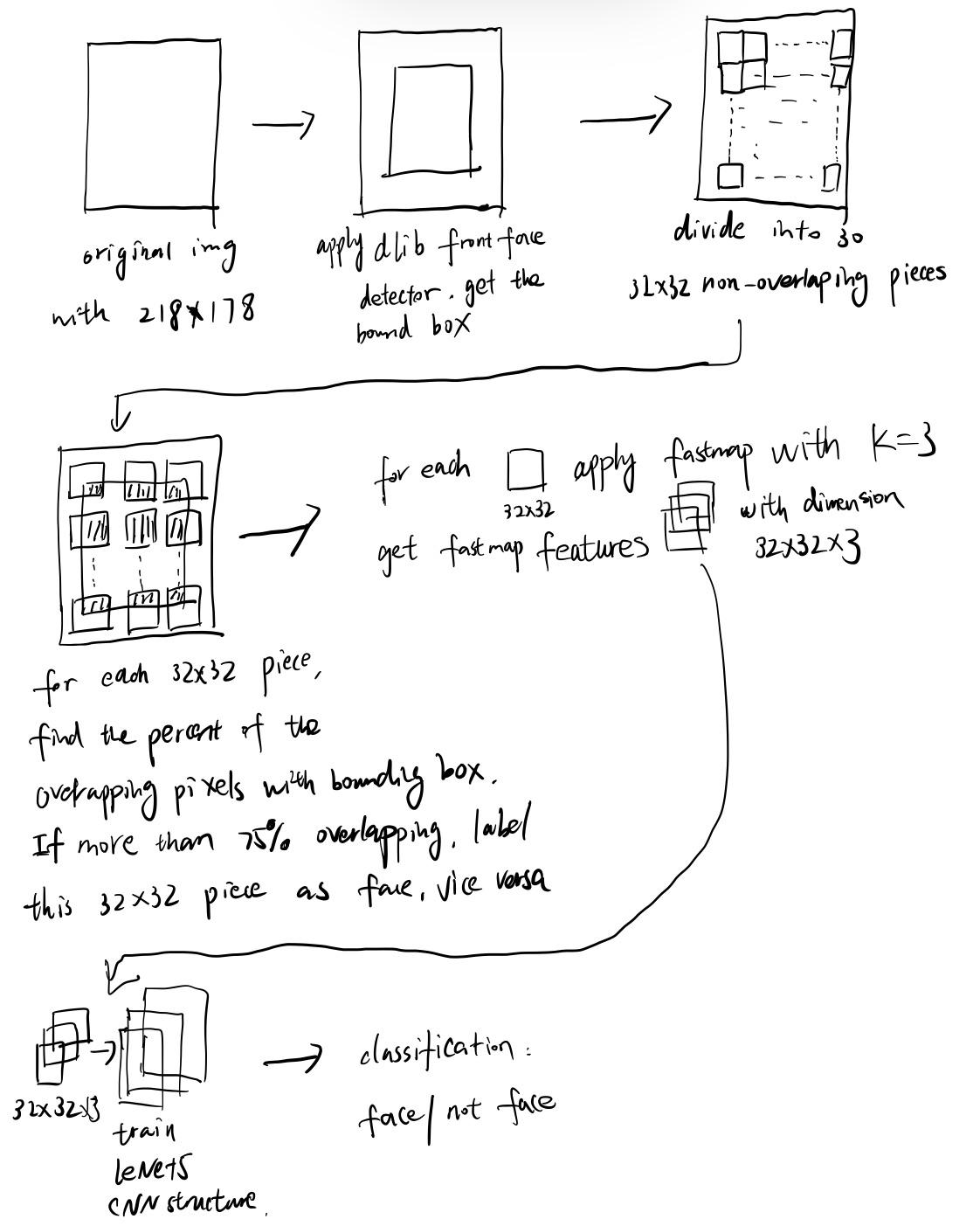
Part 2: Flow chart of the Algorithm:

The flowchart of the Algorithm is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | RGB Image | Fastmap 1 | F1& RGB image | Fastmap 2 | F2& RGB image |
| # of channel | 3 | 3 | 6 | 3 | 6 |
| Best Test Accuracy | 0.8204 | 0.9011 | **0.9491** | 0.9445 | 0.9490 |

Cifar 10:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | RGB Image | Fastmap 1 | F1& RGB image | Fastmap 2 | F2& RGB image |
| # of channel | 3 | 3 | 6 | 3 | 6 |
| Best Test Accuracy | **0.7671** | 0.4101 | 0.7572 | 0.6941 | 0.7521 |



We started with our original rgb image with the dimension 218 \* 178 \*3, then we convert it to the grayscale image. After that we divided each image into 30 32\*32 non-overlapping pieces. Since we would like to know whether each piece is a face component or not, the first thing is to label them. We labeled them based on the bounding box returned by **dlib front face detector**. The rule is if a piece overlaps 75% of its pixels with the bounding box, we label it as the face component, vice versa.

After labeling, we applied L2 variant fastmap algorithm on each 32 \* 32 piece with K =3 and fed these preprocessed 32 \*32 \* 3 fastmap features to CNN lenet5 structure and trained from the scratch. We output each piece as face/not face(binary classification)

Part 3: Two different approaches in defining the graph of the image:

To apply the fastmap algorithm, we will need to define the graph of the image. Currently, we used two different ways:

Method 1: Define each pixel as a node, and there is an edge between each of its neighboring pixels(left,right,up,down) and itself. Each pixel node has uint8 pixel value, the weight of the edge between two nodes a and b is calculated by l2 norm(pixel(a) - pixel(b)).



Method 2: Each picture is directly feeded into the Fast Map algorithm with each pixel having x coordinate, y coordinate, and grayscale value from 0-255. We modified the original Shortest Path algorithm with the l2 distance between two pixels. The distance between two nodes a and b is calculated by the l2 norm(V(xa,ya,pixela) - V(xb,yb,pixelb)). This would be the same as if each pixel is a node and assuming every two nodes are connected.

Therefore, for graph 1, only neighboring pixels have an edge, while in graph 2 every two pixels have an edge implicitly.

Part 4: Experiment results:

In the experiment, we built our CNN using pytorch and compared the classification accuracy with graph1, graph2 and original pixel space.

Input size:

Total number of images (original image with size 218 \* 178): **1,969**

Total number of 32 \* 32 pieces after slicing: **59,070**

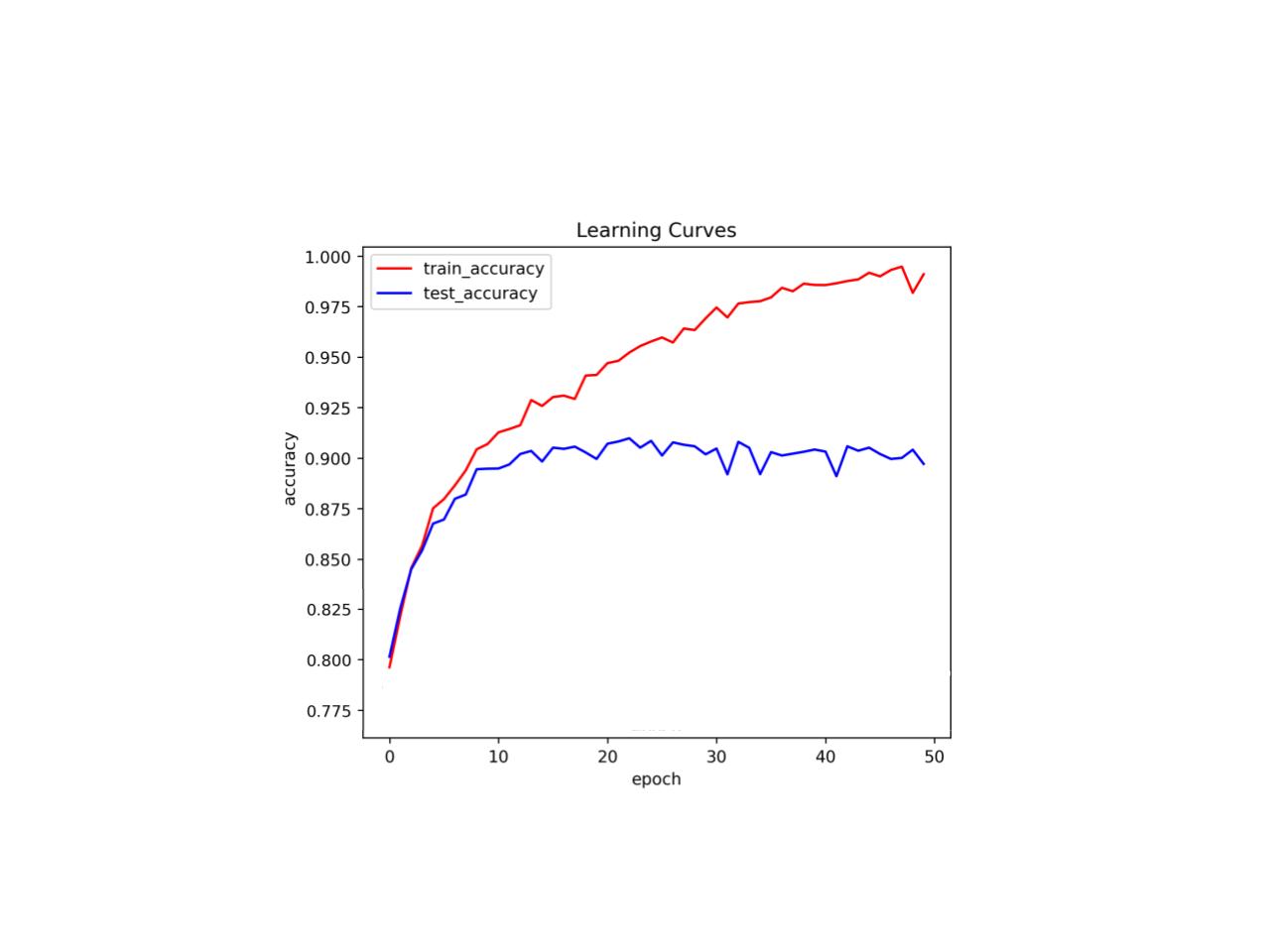
Split train and test data: **47,256**, **11,814**

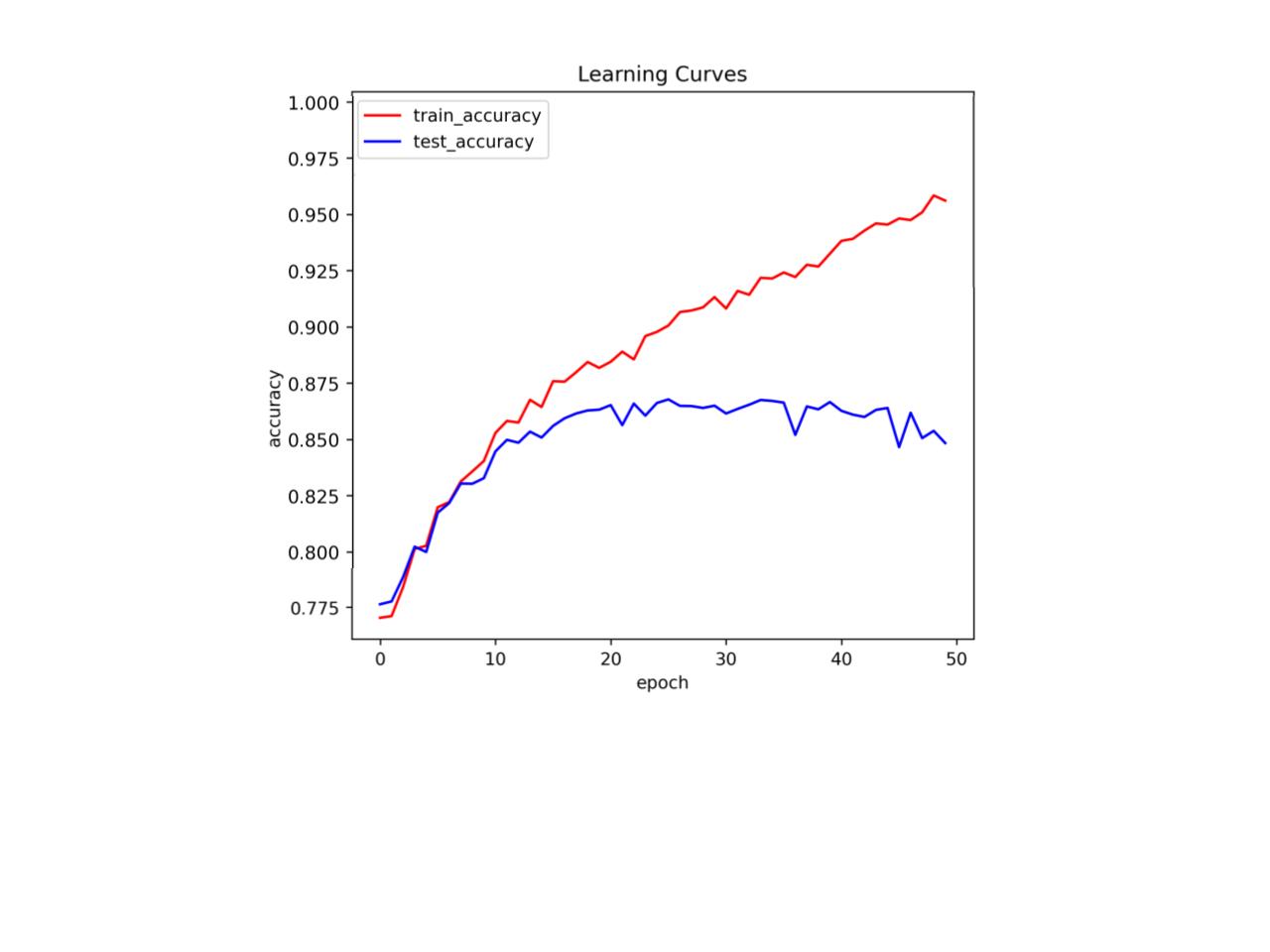
Lenet 5:

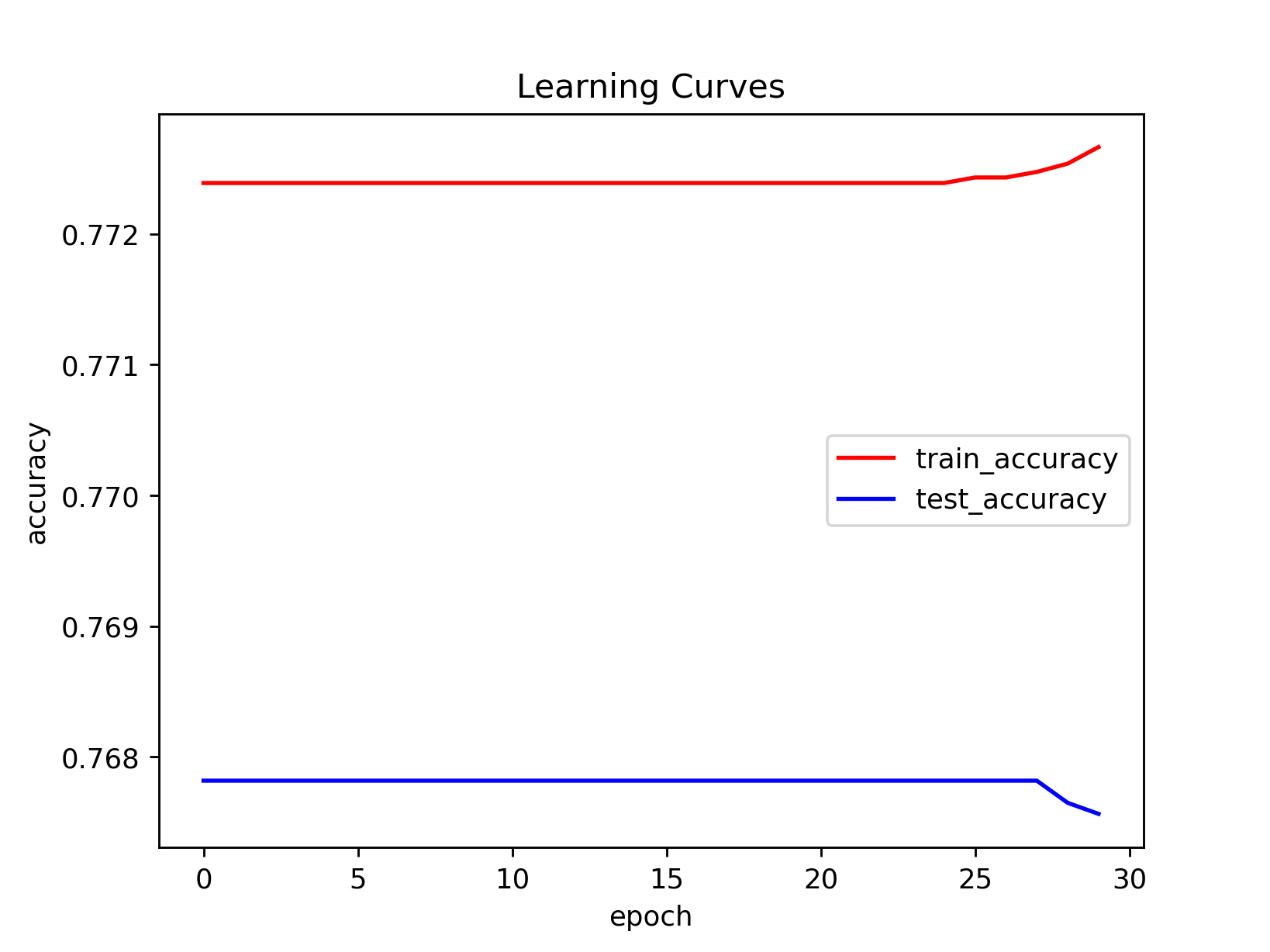
Total number of parameters: 394,198

Learning curves:

Graph 1: Only neighboring pixels have edges Method 2: Each two pixels will have a defined edge





Without FastMap(Original pixel space): 

Learning Stats:

|  |  |  |  |
| --- | --- | --- | --- |
| Stats | Number of epochs | Best test accuracy | Training time |
| Graph 1 | 50 | 0.8672 | 469.586s  (AWS p2 instance) |
| Graph 2 | 50 | 0.9098 | 472.483s  (AWS p2 instance) |
| Pixel Space | 30 | 0.7678 | 501.01s  (AWS p2 instance) |

From the current stats, we can see that graph 2 has a higher test accuracy than graph 1. However, the pixel space did not work.

Part 5: Problems encountered and some future ideas:

1) The time cost when running fastmap algorithm is large:

For running 59,070 images with the dimension 32\*32, it takes 9473.49s(2.63 hours) for method 1 and 18730.3075s(5.20 hours) for method graph2.

To reduce the fastmap run time, including aspect of algorithm implementation, we could also try to slice the image into smaller pieces such as 12\*12. In this way less calculations would be needed.

2) We also tried to pass our fast map features directly to some of the face detectors such as openCV haar cascaded filter, dlib front face detector,mtcnn and facenet. However, these detectors did not accept our input because of the scale and so on.

3) Learning curve for pixel space data:

When running images with pixel space, the training and test accuracy did not move. We have checked the labeling and inputs, they seem correct. What might be the problem?

4) Currently, our 32\* 32 pieces do not overlap with each other. Next, we could try some delay of the sliding window, to see whether the accuracy will improve. More experiments will be needed to determine the best windows size and delay size.

5) For labeling, instead of slicing the whole image into pieces and labeling each of them, we can also label the whole bounding box(full face) from the detector as a face, mixing some non-face images with the same size of the bounding box. After that, train them.

6) We have only tried preprocessing the images with FastMap hyperparameter k = 3. Other k values could be experimented.

7) In graph 2 method, we could add the weights to the x, y, and the pixel value when calculating distances between two nodes in the graph, which could affect the fastmap result

8) Instead of deep learning methods like CNN, we could also try some traditional machine learning classifiers to this problem and see how the result goes.

Class 0: airplane

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 322.29 | 316.93 | 337.49 | 319.60 | 337.75 |
| Best Test Accuracy | **0.881** | 0.77 | 0.8785 | 0.8555 | 0.8805 |

Class 1: automobile

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 321.80 | 324.43 | 337.91 | 317.85 | 338.16 |
| Test Accuracy | **0.9325** | 0.7705 | 0.921 | 0.8805 | 0.917 |

Class 2: bird

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 317.14 | 316.95 | 338.69 | 317.11 | 338.0885 |
| Test Accuracy | **0.82** | 0.6695 | 0.812 | 0.7765 | 0.8085 |

Class 3: cat

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 319.40 | 319.41 | 338.97 | 320.38 | 339.08 |
| Test Accuracy | 0.7995 | 0.647 | 0.7875 | 0.74 | **0.8015** |

Class 4: dear

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 319.18 | 319.03 | 340.08 | 320.90 | 342.03 |
| Test Accuracy | **0.86** | 0.6755 | 0.828 | 0.816 | 0.832 |

Class 5: dog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 319.13 | 321.84 | 339.57 | 323.90 | 339.01 |
| Test Accuracy | **0.8455** | 0.691 | 0.842 | 0.8065 | 0.8385 |

Class 6: frog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 319.83 | 327.45 | 341.34 | 319.76 | 341.68 |
| Test Accuracy | **0.908** | 0.723 | 0.8955 | 0.8535 | 0.89 |

Class 7: horse

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 319.02 | 322.24 | 342.33 | 319.94 | 340.00 |
| Test Accuracy | **0.895** | 0.6655 | 0.8785 | 0.853 | 0.883 |

Class 8: ship

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 319.36 | 318.90 | 344.65 | 321.00 | 344.35 |
| Test Accuracy | 0.9275 | 0.769 | 0.924 | 0.879 | 0.9205 |

Class 9: truck

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rgb | fastmap1 | f1&rgb | fastmap2 | f2&rgb |
| Training time | 320.62 | 318.48 | 339.90 | 320.18 | 341.58 |
| Test Accuracy | 0.9165 | 0.718 | 0.902 | 0.874 | 0.899 |