

# HW2

Shaan Gill

September 2023

# 1 Deliverables

1.

Link to Github:

<https://github.com/ShaanGil1/CS6220-HW2>

Link to dataset:

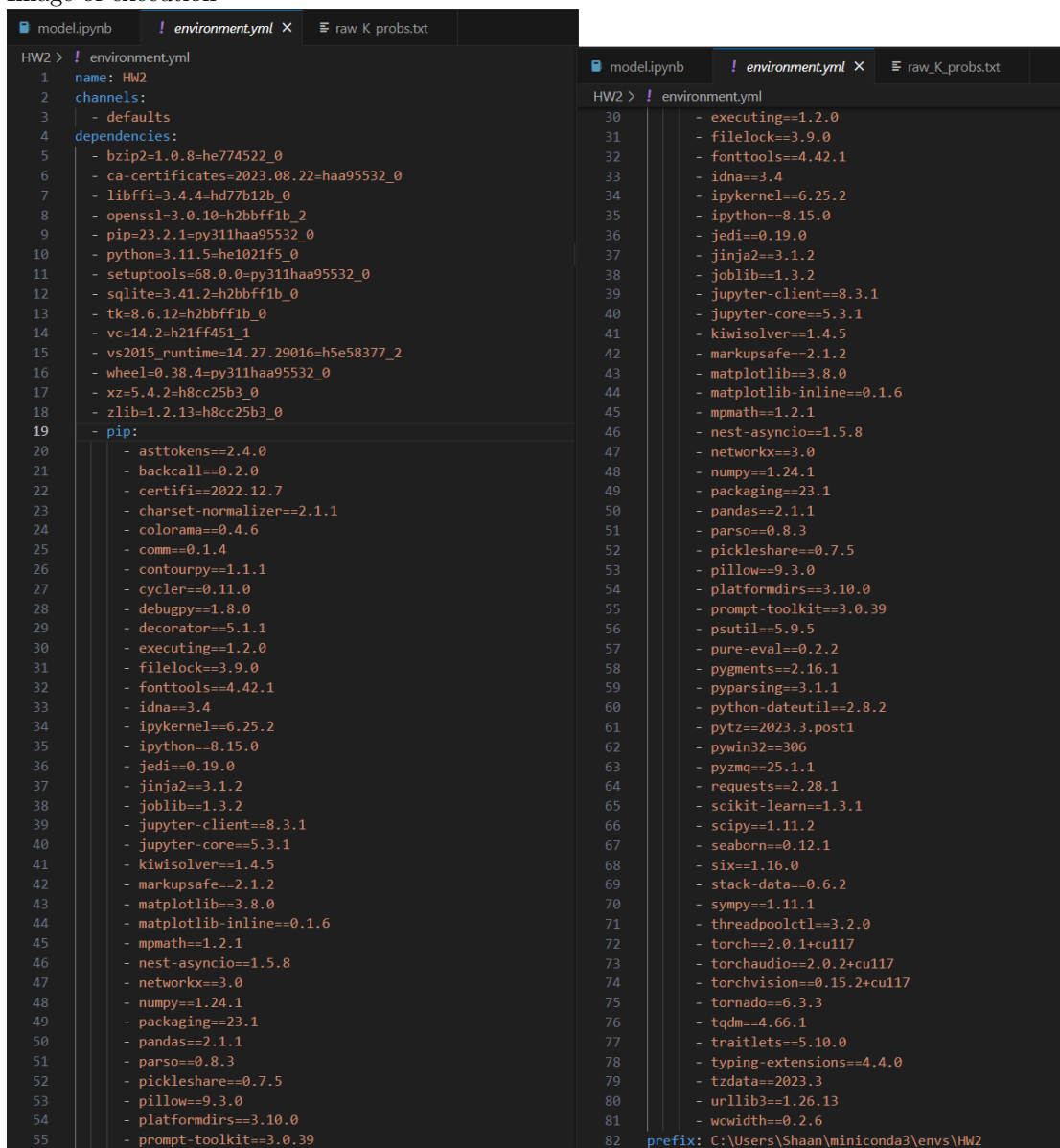
<https://www.cs.toronto.edu/~kriz/cifar.html>

Tutorial on how to download dataset on PyTorch:

[https://pytorch.org/tutorials/beginner/blitz/cifar10\\_tutorial.html](https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html)

2.

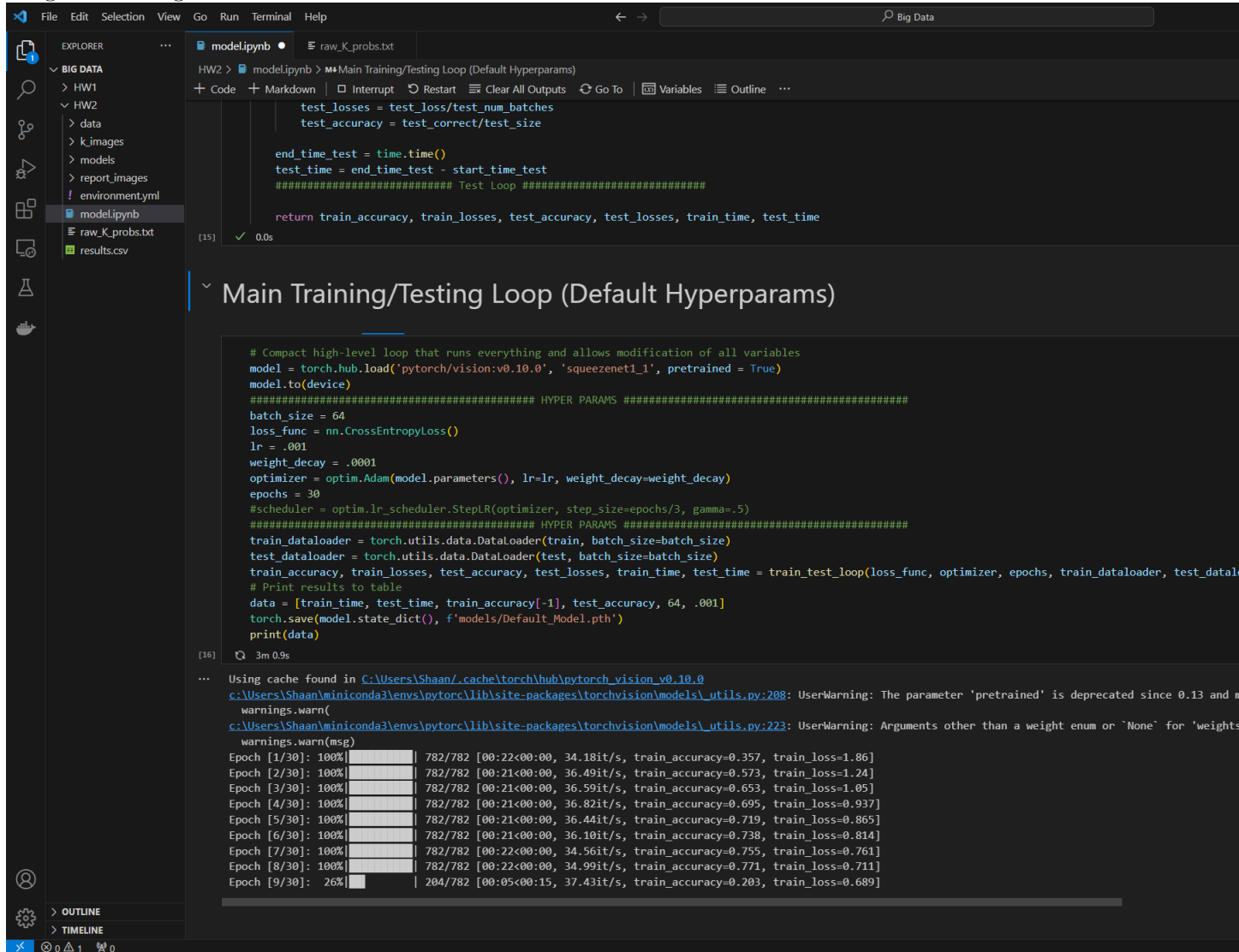
Image of execution



```
model.ipynb | ! environment.yml x | raw_K_probs.txt
HW2 > ! environment.yml
1 name: HW2
2 channels:
3   - defaults
4 dependencies:
5   - bzip2=1.0.8=he774522_0
6   - ca-certificates=2023.08.22=haa95532_0
7   - libffi=3.4.4=hd77b12b_0
8   - openssl=3.0.10=h2bbff1b_2
9   - pip=23.2.1=py311haa95532_0
10  - python=3.11.5=he1021f5_0
11  - setuptools=68.0.0=py311haa95532_0
12  - sqlite=3.41.2=h2bbff1b_0
13  - tk=8.6.12=h2bbff1b_0
14  - vc=14.2=h21ff451_1
15  - vs2015_runtime=14.27.29016=h5e58377_2
16  - wheel=0.38.4=py311haa95532_0
17  - xz=5.4.2=h8cc25b3_0
18  - zlib=1.2.13=h8cc25b3_0
19  - pip:
20    - asttokens==2.4.0
21    - backcall==0.2.0
22    - certifi==2022.12.7
23    - charset-normalizer==2.1.1
24    - colorama==0.4.6
25    - comm==0.1.4
26    - contourpy==1.1.1
27    - cycler==0.11.0
28    - debugpy==1.8.0
29    - decorator==5.1.1
30    - executing==1.2.0
31    - filelock==3.9.0
32    - fonttools==4.42.1
33    - idna==3.4
34    - ipykernel==6.25.2
35    - ipython==8.15.0
36    - jedi==0.19.0
37    - Jinja2==3.1.2
38    - joblib==1.3.2
39    - jupyter-client==8.3.1
40    - jupyter-core==5.3.1
41    - kiwisolver==1.4.5
42    - MarkupSafe==2.1.2
43    - matplotlib==3.8.0
44    - matplotlib-inline==0.1.6
45    - mpmath==1.2.1
46    - nest-asyncio==1.5.8
47    - networkx==3.0
48    - numpy==1.24.1
49    - packaging==23.1
50    - pandas==2.1.1
51    - parso==0.8.3
52    - pickleshare==0.7.5
53    - pillow==9.3.0
54    - platformdirs==3.10.0
55    - prompt-toolkit==3.0.39
56    - executing==1.2.0
57    - filelock==3.9.0
58    - fonttools==4.42.1
59    - idna==3.4
60    - ipykernel==6.25.2
61    - ipython==8.15.0
62    - jedi==0.19.0
63    - Jinja2==3.1.2
64    - joblib==1.3.2
65    - jupyter-client==8.3.1
66    - jupyter-core==5.3.1
67    - kiwisolver==1.4.5
68    - MarkupSafe==2.1.2
69    - matplotlib==3.8.0
70    - matplotlib-inline==0.1.6
71    - mpmath==1.2.1
72    - nest-asyncio==1.5.8
73    - networkx==3.0
74    - numpy==1.24.1
75    - packaging==23.1
76    - pandas==2.1.1
77    - parso==0.8.3
78    - pickleshare==0.7.5
79    - pillow==9.3.0
80    - platformdirs==3.10.0
81    - prompt-toolkit==3.0.39
82    - executing==1.2.0
83    - filelock==3.9.0
84    - fonttools==4.42.1
85    - idna==3.4
86    - ipykernel==6.25.2
87    - ipython==8.15.0
88    - jedi==0.19.0
89    - Jinja2==3.1.2
90    - joblib==1.3.2
91    - jupyter-client==8.3.1
92    - jupyter-core==5.3.1
93    - kiwisolver==1.4.5
94    - MarkupSafe==2.1.2
95    - matplotlib==3.8.0
96    - matplotlib-inline==0.1.6
97    - mpmath==1.2.1
98    - nest-asyncio==1.5.8
99    - networkx==3.0
100   - numpy==1.24.1
101   - packaging==23.1
102   - pandas==2.1.1
103   - parso==0.8.3
104   - pickleshare==0.7.5
105   - pillow==9.3.0
106   - platformdirs==3.10.0
107   - prompt-toolkit==3.0.39
108   - executing==1.2.0
109   - filelock==3.9.0
110   - fonttools==4.42.1
111   - idna==3.4
112   - ipykernel==6.25.2
113   - ipython==8.15.0
114   - jedi==0.19.0
115   - Jinja2==3.1.2
116   - joblib==1.3.2
117   - jupyter-client==8.3.1
118   - jupyter-core==5.3.1
119   - kiwisolver==1.4.5
120   - MarkupSafe==2.1.2
121   - matplotlib==3.8.0
122   - matplotlib-inline==0.1.6
123   - mpmath==1.2.1
124   - nest-asyncio==1.5.8
125   - networkx==3.0
126   - numpy==1.24.1
127   - packaging==23.1
128   - pandas==2.1.1
129   - parso==0.8.3
130   - pickleshare==0.7.5
131   - pillow==9.3.0
132   - platformdirs==3.10.0
133   - prompt-toolkit==3.0.39
134   - executing==1.2.0
135   - filelock==3.9.0
136   - fonttools==4.42.1
137   - idna==3.4
138   - ipykernel==6.25.2
139   - ipython==8.15.0
140   - jedi==0.19.0
141   - Jinja2==3.1.2
142   - joblib==1.3.2
143   - jupyter-client==8.3.1
144   - jupyter-core==5.3.1
145   - kiwisolver==1.4.5
146   - MarkupSafe==2.1.2
147   - matplotlib==3.8.0
148   - matplotlib-inline==0.1.6
149   - mpmath==1.2.1
150   - nest-asyncio==1.5.8
151   - networkx==3.0
152   - numpy==1.24.1
153   - packaging==23.1
154   - pandas==2.1.1
155   - parso==0.8.3
156   - pickleshare==0.7.5
157   - pillow==9.3.0
158   - platformdirs==3.10.0
159   - prompt-toolkit==3.0.39
160   - executing==1.2.0
161   - filelock==3.9.0
162   - fonttools==4.42.1
163   - idna==3.4
164   - ipykernel==6.25.2
165   - ipython==8.15.0
166   - jedi==0.19.0
167   - Jinja2==3.1.2
168   - joblib==1.3.2
169   - jupyter-client==8.3.1
170   - jupyter-core==5.3.1
171   - kiwisolver==1.4.5
172   - MarkupSafe==2.1.2
173   - matplotlib==3.8.0
174   - matplotlib-inline==0.1.6
175   - mpmath==1.2.1
176   - nest-asyncio==1.5.8
177   - networkx==3.0
178   - numpy==1.24.1
179   - packaging==23.1
180   - pandas==2.1.1
181   - parso==0.8.3
182   - pickleshare==0.7.5
183   - pillow==9.3.0
184   - platformdirs==3.10.0
185   - prompt-toolkit==3.0.39
186   - executing==1.2.0
187   - filelock==3.9.0
188   - fonttools==4.42.1
189   - idna==3.4
190   - ipykernel==6.25.2
191   - ipython==8.15.0
192   - jedi==0.19.0
193   - Jinja2==3.1.2
194   - joblib==1.3.2
195   - jupyter-client==8.3.1
196   - jupyter-core==5.3.1
197   - kiwisolver==1.4.5
198   - MarkupSafe==2.1.2
199   - matplotlib==3.8.0
200   - matplotlib-inline==0.1.6
201   - mpmath==1.2.1
202   - nest-asyncio==1.5.8
203   - networkx==3.0
204   - numpy==1.24.1
205   - packaging==23.1
206   - pandas==2.1.1
207   - parso==0.8.3
208   - pickleshare==0.7.5
209   - pillow==9.3.0
210   - platformdirs==3.10.0
211   - prompt-toolkit==3.0.39
212   - executing==1.2.0
213   - filelock==3.9.0
214   - fonttools==4.42.1
215   - idna==3.4
216   - ipykernel==6.25.2
217   - ipython==8.15.0
218   - jedi==0.19.0
219   - Jinja2==3.1.2
220   - joblib==1.3.2
221   - jupyter-client==8.3.1
222   - jupyter-core==5.3.1
223   - kiwisolver==1.4.5
224   - MarkupSafe==2.1.2
225   - matplotlib==3.8.0
226   - matplotlib-inline==0.1.6
227   - mpmath==1.2.1
228   - nest-asyncio==1.5.8
229   - networkx==3.0
230   - numpy==1.24.1
231   - packaging==23.1
232   - pandas==2.1.1
233   - parso==0.8.3
234   - pickleshare==0.7.5
235   - pillow==9.3.0
236   - platformdirs==3.10.0
237   - prompt-toolkit==3.0.39
238   - executing==1.2.0
239   - filelock==3.9.0
240   - fonttools==4.42.1
241   - idna==3.4
242   - ipykernel==6.25.2
243   - ipython==8.15.0
244   - jedi==0.19.0
245   - Jinja2==3.1.2
246   - joblib==1.3.2
247   - jupyter-client==8.3.1
248   - jupyter-core==5.3.1
249   - kiwisolver==1.4.5
250   - MarkupSafe==2.1.2
251   - matplotlib==3.8.0
252   - matplotlib-inline==0.1.6
253   - mpmath==1.2.1
254   - nest-asyncio==1.5.8
255   - networkx==3.0
256   - numpy==1.24.1
257   - packaging==23.1
258   - pandas==2.1.1
259   - parso==0.8.3
260   - pickleshare==0.7.5
261   - pillow==9.3.0
262   - platformdirs==3.10.0
263   - prompt-toolkit==3.0.39
264   - executing==1.2.0
265   - filelock==3.9.0
266   - fonttools==4.42.1
267   - idna==3.4
268   - ipykernel==6.25.2
269   - ipython==8.15.0
270   - jedi==0.19.0
271   - Jinja2==3.1.2
272   - joblib==1.3.2
273   - jupyter-client==8.3.1
274   - jupyter-core==5.3.1
275   - kiwisolver==1.4.5
276   - MarkupSafe==2.1.2
277   - matplotlib==3.8.0
278   - matplotlib-inline==0.1.6
279   - mpmath==1.2.1
280   - nest-asyncio==1.5.8
281   - networkx==3.0
282   - numpy==1.24.1
283   - packaging==23.1
284   - pandas==2.1.1
285   - parso==0.8.3
286   - pickleshare==0.7.5
287   - pillow==9.3.0
288   - platformdirs==3.10.0
289   - prompt-toolkit==3.0.39
290   - executing==1.2.0
291   - filelock==3.9.0
292   - fonttools==4.42.1
293   - idna==3.4
294   - ipykernel==6.25.2
295   - ipython==8.15.0
296   - jedi==0.19.0
297   - Jinja2==3.1.2
298   - joblib==1.3.2
299   - jupyter-client==8.3.1
300   - jupyter-core==5.3.1
301   - kiwisolver==1.4.5
302   - MarkupSafe==2.1.2
303   - matplotlib==3.8.0
304   - matplotlib-inline==0.1.6
305   - mpmath==1.2.1
306   - nest-asyncio==1.5.8
307   - networkx==3.0
308   - numpy==1.24.1
309   - packaging==23.1
310   - pandas==2.1.1
311   - parso==0.8.3
312   - pickleshare==0.7.5
313   - pillow==9.3.0
314   - platformdirs==3.10.0
315   - prompt-toolkit==3.0.39
316   - executing==1.2.0
317   - filelock==3.9.0
318   - fonttools==4.42.1
319   - idna==3.4
320   - ipykernel==6.25.2
321   - ipython==8.15.0
322   - jedi==0.19.0
323   - Jinja2==3.1.2
324   - joblib==1.3.2
325   - jupyter-client==8.3.1
326   - jupyter-core==5.3.1
327   - kiwisolver==1.4.5
328   - MarkupSafe==2.1.2
329   - matplotlib==3.8.0
330   - matplotlib-inline==0.1.6
331   - mpmath==1.2.1
332   - nest-asyncio==1.5.8
333   - networkx==3.0
334   - numpy==1.24.1
335   - packaging==23.1
336   - pandas==2.1.1
337   - parso==0.8.3
338   - pickleshare==0.7.5
339   - pillow==9.3.0
340   - platformdirs==3.10.0
341   - prompt-toolkit==3.0.39
342   - executing==1.2.0
343   - filelock==3.9.0
344   - fonttools==4.42.1
345   - idna==3.4
346   - ipykernel==6.25.2
347   - ipython==8.15.0
348   - jedi==0.19.0
349   - Jinja2==3.1.2
350   - joblib==1.3.2
351   - jupyter-client==8.3.1
352   - jupyter-core==5.3.1
353   - kiwisolver==1.4.5
354   - MarkupSafe==2.1.2
355   - matplotlib==3.8.0
356   - matplotlib-inline==0.1.6
357   - mpmath==1.2.1
358   - nest-asyncio==1.5.8
359   - networkx==3.0
360   - numpy==1.24.1
361   - packaging==23.1
362   - pandas==2.1.1
363   - parso==0.8.3
364   - pickleshare==0.7.5
365   - pillow==9.3.0
366   - platformdirs==3.10.0
367   - prompt-toolkit==3.0.39
368   - executing==1.2.0
369   - filelock==3.9.0
370   - fonttools==4.42.1
371   - idna==3.4
372   - ipykernel==6.25.2
373   - ipython==8.15.0
374   - jedi==0.19.0
375   - Jinja2==3.1.2
376   - joblib==1.3.2
377   - jupyter-client==8.3.1
378   - jupyter-core==5.3.1
379   - kiwisolver==1.4.5
380   - MarkupSafe==2.1.2
381   - matplotlib==3.8.0
382   - matplotlib-inline==0.1.6
383   - mpmath==1.2.1
384   - nest-asyncio==1.5.8
385   - networkx==3.0
386   - numpy==1.24.1
387   - packaging==23.1
388   - pandas==2.1.1
389   - parso==0.8.3
390   - pickleshare==0.7.5
391   - pillow==9.3.0
392   - platformdirs==3.10.0
393   - prompt-toolkit==3.0.39
394   - executing==1.2.0
395   - filelock==3.9.0
396   - fonttools==4.42.1
397   - idna==3.4
398   - ipykernel==6.25.2
399   - ipython==8.15.0
400   - jedi==0.19.0
401   - Jinja2==3.1.2
402   - joblib==1.3.2
403   - jupyter-client==8.3.1
404   - jupyter-core==5.3.1
405   - kiwisolver==1.4.5
406   - MarkupSafe==2.1.2
407   - matplotlib==3.8.0
408   - matplotlib-inline==0.1.6
409   - mpmath==1.2.1
410   - nest-asyncio==1.5.8
411   - networkx==3.0
412   - numpy==1.24.1
413   - packaging==23.1
414   - pandas==2.1.1
415   - parso==0.8.3
416   - pickleshare==0.7.5
417   - pillow==9.3.0
418   - platformdirs==3.10.0
419   - prompt-toolkit==3.0.39
420   - executing==1.2.0
421   - filelock==3.9.0
422   - fonttools==4.42.1
423   - idna==3.4
424   - ipykernel==6.25.2
425   - ipython==8.15.0
426   - jedi==0.19.0
427   - Jinja2==3.1.2
428   - joblib==1.3.2
429   - jupyter-client==8.3.1
430   - jupyter-core==5.3.1
431   - kiwisolver==1.4.5
432   - MarkupSafe==2.1.2
433   - matplotlib==3.8.0
434   - matplotlib-inline==0.1.6
435   - mpmath==1.2.1
436   - nest-asyncio==1.5.8
437   - networkx==3.0
438   - numpy==1.24.1
439   - packaging==23.1
440   - pandas==2.1.1
441   - parso==0.8.3
442   - pickleshare==0.7.5
443   - pillow==9.3.0
444   - platformdirs==3.10.0
445   - prompt-toolkit==3.0.39
446   - executing==1.2.0
447   - filelock==3.9.0
448   - fonttools==4.42.1
449   - idna==3.4
450   - ipykernel==6.25.2
451   - ipython==8.15.0
452   - jedi==0.19.0
453   - Jinja2==3.1.2
454   - joblib==1.3.2
455   - jupyter-client==8.3.1
456   - jupyter-core==5.3.1
457   - kiwisolver==1.4.5
458   - MarkupSafe==2.1.2
459   - matplotlib==3.8.0
460   - matplotlib-inline==0.1.6
461   - mpmath==1.2.1
462   - nest-asyncio==1.5.8
463   - networkx==3.0
464   - numpy==1.24.1
465   - packaging==23.1
466   - pandas==2.1.1
467   - parso==0.8.3
468   - pickleshare==0.7.5
469   - pillow==9.3.0
470   - platformdirs==3.10.0
471   - prompt-toolkit==3.0.39
472   - executing==1.2.0
473   - filelock==3.9.0
474   - fonttools==4.42.1
475   - idna==3.4
476   - ipykernel==6.25.2
477   - ipython==8.15.0
478   - jedi==0.19.0
479   - Jinja2==3.1.2
480   - joblib==1.3.2
481   - jupyter-client==8.3.1
482   - jupyter-core==5.3.1
483   - kiwisolver==1.4.5
484   - MarkupSafe==2.1.2
485   - matplotlib==3.8.0
486   - matplotlib-inline==0.1.6
487   - mpmath==1.2.1
488   - nest-asyncio==1.5.8
489   - networkx==3.0
490   - numpy==1.24.1
491   - packaging==23.1
492   - pandas==2.1.1
493   - parso==0.8.3
494   - pickleshare==0.7.5
495   - pillow==9.3.0
496   - platformdirs==3.10.0
497   - prompt-toolkit==3.0.39
498   - executing==1.2.0
499   - filelock==3.9.0
500   - fonttools==4.42.1
501   - idna==3.4
502   - ipykernel==6.25.2
503   - ipython==8.15.0
504   - jedi==0.19.0
505   - Jinja2==3.1.2
506   - joblib==1.3.2
507   - jupyter-client==8.3.1
508   - jupyter-core==5.3.1
509   - kiwisolver==1.4.5
510   - MarkupSafe==2.1.2
511   - matplotlib==3.8.0
512   - matplotlib-inline==0.1.6
513   - mpmath==1.2.1
514   - nest-asyncio==1.5.8
515   - networkx==3.0
516   - numpy==1.24.1
517   - packaging==23.1
518   - pandas==2.1.1
519   - parso==0.8.3
520   - pickleshare==0.7.5
521   - pillow==9.3.0
522   - platformdirs==3.10.0
523   - prompt-toolkit==3.0.39
524   - executing==1.2.0
525   - filelock==3.9.0
526   - fonttools==4.42.1
527   - idna==3.4
528   - ipykernel==6.25.2
529   - ipython==8.15.0
530   - jedi==0.19.0
531   - Jinja2==3.1.2
532   - joblib==1.3.2
533   - jupyter-client==8.3.1
534   - jupyter-core==5.3.1
535   - kiwisolver==1.4.5
536   - MarkupSafe==2.1.2
537   - matplotlib==3.8.0
538   - matplotlib-inline==0.1.6
539   - mpmath==1.2.1
540   - nest-asyncio==1.5.8
541   - networkx==3.0
542   - numpy==1.24.1
543   - packaging==23.1
544   - pandas==2.1.1
545   - parso==0.8.3
546   - pickleshare==0.7.5
547   - pillow==9.3.0
548   - platformdirs==3.10.0
549   - prompt-toolkit==3.0.39
550   - executing==1.2.0
551   - filelock==3.9.0
552   - fonttools==4.42.1
553   - idna==3.4
554   - ipykernel==6.25.2
555   - ipython==8.15.0
556   - jedi==0.19.0
557   - Jinja2==3.1.2
558   - joblib==1.3.2
559   - jupyter-client==8.3.1
560   - jupyter-core==5.3.1
561   - kiwisolver==1.4.5
562   - MarkupSafe==2.1.2
563   - matplotlib==3.8.0
564   - matplotlib-inline==0.1.6
565   - mpmath==1.2.1
566   - nest-asyncio==1.5.8
567   - networkx==3.0
568   - numpy==1.24.1
569   - packaging==23.1
570   - pandas==2.1.1
571   - parso==0.8.3
572   - pickleshare==0.7.5
573   - pillow==9.3.0
574   - platformdirs==3.10.0
575   - prompt-toolkit==3.0.39
576   - executing==1.2.0
577   - filelock==3.9.0
578   - fonttools==4.42.1
579   - idna==3.4
580   - ipykernel==6.25.2
581   - ipython==8.15.0
582   - jedi==0.19.0
583   - Jinja2==3.1.2
584   - joblib==1.3.2
585   - jupyter-client==8.3.1
586   - jupyter-core==5.3.1
587   - kiwisolver==1.4.5
588   - MarkupSafe==2.1.2
589   - matplotlib==3.8.0
590   - matplotlib-inline==0.1.6
591   - mpmath==1.2.1
592   - nest-asyncio==1.5.8
593   - networkx==3.0
594   - numpy==1.24.1
595   - packaging==23.1
596   - pandas==2.1.1
597   - parso==0.8.3
598   - pickleshare==0.7.5
599   - pillow==9.3.0
600   - platformdirs==3.10.0
601   - prompt-toolkit==3.0.39
602   - executing==1.2.0
603   - filelock==3.9.0
604   - fonttools==4.42.1
605   - idna==3.4
606   - ipykernel==6.25.2
607   - ipython==8.15.0
608   - jedi==0.19.0
609   - Jinja2==3.1.2
610   - joblib==1.3.2
611   - jupyter-client==8.3.1
612   - jupyter-core==5.3.1
613   - kiwisolver==1.4.5
614   - MarkupSafe==2.1.2
615   - matplotlib==3.8.0
616   - matplotlib-inline==0.1.6
617   - mpmath==1.2.1
618   - nest-asyncio==1.5.8
619   - networkx==3.0
620   - numpy==1.24.1
621   - packaging==23.1
622   - pandas==2.1.1
623   - parso==0.8.3
624   - pickleshare==0.7.5
625   - pillow==9.3.0
626   - platformdirs==3.10.0
627   - prompt-toolkit==3.0.39
628   - executing==1.2.0
629   - filelock==3.9.0
630   - fonttools==4.42.1
631   - idna==3.4
632   - ipykernel==6.25.2
633   - ipython==8.15.0
634   - jedi==0.19.0
635   - Jinja2==3.1.2
636   - joblib==1.3.2
637   - jupyter-client==8.3.1
638   - jupyter-core==5.3.1
639   - kiwisolver==1.4.5
640   - MarkupSafe==2.1.2
641   - matplotlib==3.8.0
642   - matplotlib-inline==0.1.6
643   - mpmath==1.2.1
644   - nest-asyncio==1.5.8
645   - networkx==3.0
646   - numpy==1.24.1
647   - packaging==23.1
648   - pandas==2.1.1
649   - parso==0.8.3
650   - pickleshare==0.7.5
651   - pillow==9.3.0
652   - platformdirs==3.10.0
653   - prompt-toolkit==3.0.39
654   - executing==1.2.0
655   - filelock==3.9.0
656   - fonttools==4.42.1
657   - idna==3.4
658   - ipykernel==6.25.2
659   - ipython==8.15.0
660   - jedi==0.19.0
661   - Jinja2==3.1.2
662   - joblib==1.3.2
663   - jupyter-client==8.3.1
664   - jupyter-core==5.3.1
665   - kiwisolver==1.4.5
666   - MarkupSafe==2.1.2
667   - matplotlib==3.8.0
668   - matplotlib-inline==0.1.6
669   - mpmath==1.2.1
670   - nest-asyncio==1.5.8
671   - networkx==3.0
672   - numpy==1.24.1
673   - packaging==23.1
674   - pandas==2.1.1
675   - parso==0.8.3
676   - pickleshare==0.7.5
677   - pillow==9.3.0
678   - platformdirs==3.10.0
679   - prompt-toolkit==3.0.39
680   - executing==1.2.0
681   - filelock==3.9.0
682   - fonttools==4.42.1
683   - idna==3.4
684   - ipykernel==6.25.2
685   - ipython==8.15.0
686   - jedi==0.19.0
687   - Jinja2==3.1.2
688   - joblib==1.3.2
689   - jupyter-client==8.3.1
690   - jupyter-core==5.3.1
691   - kiwisolver==1.4.5
692   - MarkupSafe==2.1.2
693   - matplotlib==3.8.0
694   - matplotlib-inline==0.1.6
695   - mpmath==1.2.1
696   - nest-asyncio==1.5.8
697   - networkx==3.0
698   - numpy==1.24.1
699   - packaging==23.1
700   - pandas==2.1.1
701   - parso==0.8.3
702   - pickleshare==0.7.5
703   - pillow==9.3.0
704   - platformdirs==3.10.0
705   - prompt-toolkit==3.0.39
706   - executing==1.2.0
707   - filelock==3.9.0
708   - fonttools==4.42.1
709   - idna==3.4
710   - ipykernel==6.25.2
711   - ipython==8.15.0
712   - jedi==0.19.0
713   - Jinja2==3.1.2
714   - joblib==1.3.2
715   - jupyter-client==8.3.1
716   - jupyter-core==5.3.1
717   - kiwisolver==1.4.5
718   - MarkupSafe==2.1.2
719   - matplotlib==3.8.0
720   - matplotlib-inline==0.1.6
721   - mpmath==1.2.1
722   - nest-asyncio==1.5.8
723   - networkx==3.0
724   - numpy==1.24.1
725   - packaging==23.1
726   - pandas==2.1.1
727   - parso==0.8.3
728   - pickleshare==0.7.5
729   - pillow==9.3.0
730   - platformdirs==3.10.0
731   - prompt-toolkit==3.0.39
732   - executing==1.2.0
733   - filelock==3.9.0
734   - fonttools==4.42.1
735   - idna==3.4
736   - ipykernel==6.25.2
737   - ipython==8.15.0
738   - jedi==0.19.0
739   - Jinja2==3.1.2
740   - joblib==1.3.2
741   - jupyter-client==8.3.1
742   - jupyter-core==5.3.1
743   - kiwisolver==1.4.5
744   - MarkupSafe==2.1.2
745   - matplotlib==3.8.0
746   - matplotlib-inline==0.1.6
747   - mpmath==1.2.1
748   - nest-asyncio==1.5.8
749   - networkx==3.0
750   - numpy==1.24.1
751   - packaging==23.1
752   - pandas==2.1.1
753   - parso==0.8.3
754   - pickleshare==0.7.5
755   - pillow==9.3.0
756   - platformdirs==3.10.0
757   - prompt-toolkit==3.0.39
758   - executing==1.2.0
759   - filelock==3.9.0
760   - fonttools==4.42.1
761   - idna==3.4
762   - ipykernel==6.25.2
763   - ipython==8.15.0
764   - jedi==0.19.0
765   - Jinja2==3.1.2
766   - joblib==1.3.2
767   - jupyter-client==8.3.1
768   - jupyter-core==5.3.1
769   - kiwisolver==1.4.5
770   - MarkupSafe==2.1.2
771   - matplotlib==3.8.0
772   - matplotlib-inline==0.1.6
773   - mpmath==1.2.1
774   - nest-asyncio==1.5.8
775   - networkx==3.0
776   - numpy==1.24.1
777   - packaging==23.1
778   - pandas==2.1.1
779   - parso==0.8.3
780   - pickleshare==0.7.5
781   - pillow==9.3.0
782   - platformdirs==3.10.0
783   - prompt-toolkit==3.0.39
784   - executing==1.2.0
785   - filelock==3.9.0
786   - fonttools==4.42.1
787   - idna==3.4
788   - ipykernel==6.25.2
789   - ipython==8.15.0
790   - jedi==0.19.0
791   - Jinja2==3.1.2
792   - joblib==1.3.2
793   - jupyter-client==8.3.1
794   - jupyter-core==5.3.1
795   - kiwisolver==1.4.5
796   - MarkupSafe==2.1.2
797   - matplotlib==3.8.0
798   - matplotlib-inline==0.1.6
799   - mpmath==1.2.1
800   - nest-asyncio==1.5.8
801   - networkx==3.0
802   - numpy==1.24.1
803   - packaging==23.1
804   - pandas==2.1.1
805   - parso==0.8.3
806   - pickleshare==0.7.5
807   - pillow==9.3.0
808   - platformdirs==3.10.0
809   - prompt-toolkit==3.0.39
810   - executing==1.2.0
811   - filelock==3.9.0
812   - fonttools==4.42.1
813   - idna==3.4
814   - ipykernel==6.25.2
815   - ipython==8.15.0
816   - jedi==0.19.0
817   - Jinja2==3.1.2
818   - joblib==1.3.2
819   - jupyter-client==8.3.1
820   - jupyter-core==5.3.1
821   - kiwisolver==1.4.5
822   - MarkupSafe==2.1.2
823   - matplotlib==3.8.0
824   - matplotlib-inline==0.1.6
825   - mpmath==1.2.1
826   - nest-asyncio==1.5.8
827   - networkx==3.0

```

## Image of Running the default model



```
HW2 > modelLipynb > Main Training/Testing Loop (Default Hyperparams)
+ Code + Markdown + Interrupt + Restart + Clear All Outputs + Go To + Variables + Outline ...

test_losses = test_loss/test_num_batches
test_accuracy = test_correct/test_size

end_time_test = time.time()
test_time = end_time_test - start_time_test
##### Test Loop #####

return train_accuracy, train_losses, test_accuracy, test_losses, train_time, test_time

[15] ✓ 0.0s

Main Training/Testing Loop (Default Hyperparams)

# Compact high-level loop that runs everything and allows modification of all variables
model = torch.hub.load('pytorch/vision:v0.10.0', 'squeezenet1_1', pretrained = True)
model.to(device)
##### HYPER PARAMS #####
batch_size = 64
loss_func = nn.CrossEntropyLoss()
lr = .001
weight_decay = .0001
optimizer = optim.Adam(model.parameters(), lr=lr, weight_decay=weight_decay)
epochs = 30
#scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=epochs/3, gamma=.5)
##### HYPER PARAMS #####
train_dataloader = torch.utils.data.DataLoader(train, batch_size=batch_size)
test_dataloader = torch.utils.data.DataLoader(test, batch_size=batch_size)
train_accuracy, train_losses, test_accuracy, test_losses, train_time, test_time = train_test_loop(loss_func, optimizer, epochs, train_dataloader, test_dataloader)
# Print results to table
data = [train_time, test_time, train_accuracy[-1], test_accuracy, 64, .001]
torch.save(model.state_dict(), f'models/Default_Model.pth')
print(data)

[16] 3m 0.9s

... Using cache found in C:\Users\Shaan\.cache\torch\hub\pytorch_vision_v0.10.0
c:\Users\Shaan\miniconda3\envs\pytorch\lib\site-packages\torchvision\models\_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and will be removed in a future version of PyTorch. Please use 'weights' instead.
warnings.warn(
c:\Users\Shaan\miniconda3\envs\pytorch\lib\site-packages\torchvision\models\_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated.
warnings.warn(msg)

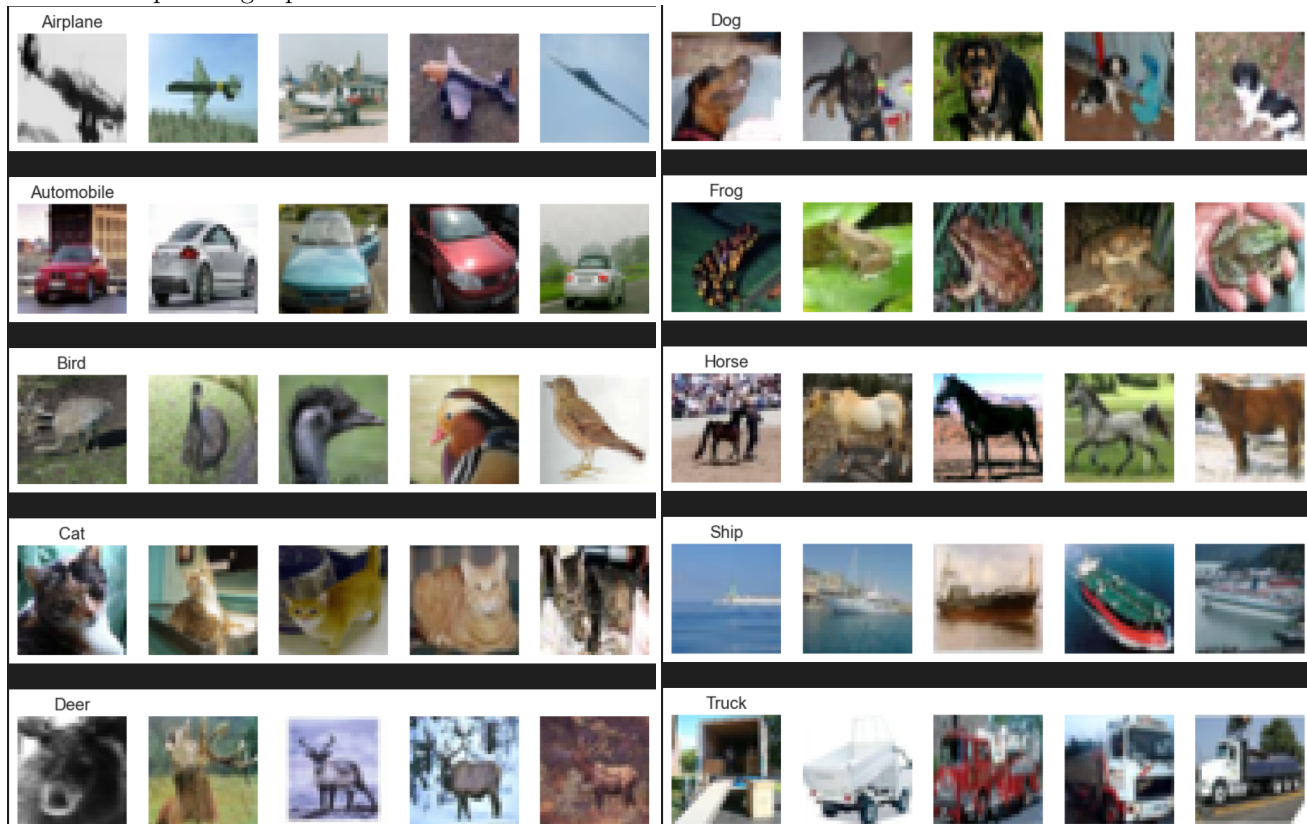
Epoch [1/30]: 100% 782/782 [00:22<00:00, 34.18it/s, train_accuracy=0.357, train_loss=1.86]
Epoch [2/30]: 100% 782/782 [00:21<00:00, 36.49it/s, train_accuracy=0.573, train_loss=1.24]
Epoch [3/30]: 100% 782/782 [00:21<00:00, 36.59it/s, train_accuracy=0.653, train_loss=1.05]
Epoch [4/30]: 100% 782/782 [00:21<00:00, 36.82it/s, train_accuracy=0.695, train_loss=0.937]
Epoch [5/30]: 100% 782/782 [00:21<00:00, 36.44it/s, train_accuracy=0.719, train_loss=0.865]
Epoch [6/30]: 100% 782/782 [00:21<00:00, 36.10it/s, train_accuracy=0.738, train_loss=0.814]
Epoch [7/30]: 100% 782/782 [00:22<00:00, 34.56it/s, train_accuracy=0.755, train_loss=0.761]
Epoch [8/30]: 100% 782/782 [00:22<00:00, 34.99it/s, train_accuracy=0.771, train_loss=0.711]
Epoch [9/30]: 26% 204/782 [00:05<00:15, 37.43it/s, train_accuracy=0.203, train_loss=0.689]
```

## 2 Input Analysis

### a. Input Dataset:

- Size: 60,000 Images
- Resolution: 32 x 32 x 3
- Size of each image in Kilobytes: 3.072 KB
- Size of the whole dataset in Megabytes: 184.32 MB

b. 5 Sample Images per class for all K classes:



c. Test/Training Split was .2 (The dataset had already defined a train/test set and that was the value given and used)

- Training Set: 50,000 Images
- Testing Set: 10,000 Images

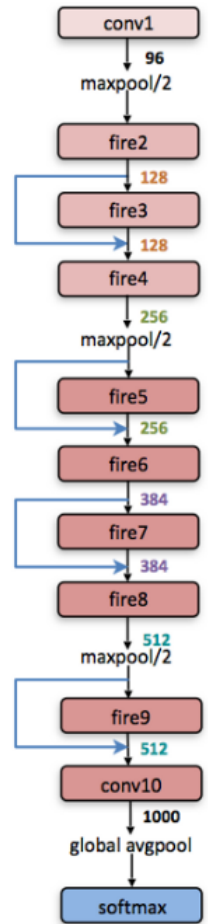
d. Default settings for NN:

Input Parameters	
Input Image Size	Varied
Image Size Used for Training/Testing	32 x 32 x 3
Number of Classes	10
Number of Training Images	50,000
Number of Testing Images	10,000
Loss Function	Cross Entropy Loss
Kernel Sizes/Number of Filters	Refer to Image Below
Weight Initialization	Pretrained Weights
Learning Rate	0.0001
Batch Size	64
Optimizer	Adam
Weight Decay	0.0001
Epochs	30

e. Default # iterations used for convergence was 30 epochs (the point where the model's accuracy stopped having significant changes)

layer name/type	output size	filter size / stride (if not a fire layer)	depth	$s_{1 \times 1}$ (#1x1 squeeze)	$e_{1 \times 1}$ (#1x1 expand)	$e_{3 \times 3}$ (#3x3 expand)
input image	224x224x3					
conv1	111x111x96	7x7/2 {x96}	1			
maxpool1	55x55x96	3x3/2	0			
fire2	55x55x128		2	16	64	64
fire3	55x55x128		2	16	64	64
fire4	55x55x256		2	32	128	128
maxpool4	27x27x256	3x3/2	0			
fire5	27x27x256		2	32	128	128
fire6	27x27x384		2	48	192	192
fire7	27x27x384		2	48	192	192
fire8	27x27x512		2	64	256	256
maxpool8	13x12x512	3x3/2	0			
fire9	13x13x512		2	64	256	256
conv10	13x13x1000	1x1/1 {x1000}	1			
avgpool10	1x1x1000	13x13/1	0			

activations  
(input/output data between layers)
parameters



A Fire layer is a layer that compresses (squeezes) the input and then expands it two times using convolution layers followed by ReLU functions for these 3 steps. View the chart for more details on the exact values used by this model.

Image credit:

[https://pytorch.org/hub/pytorch\\_vision\\_squeezenet/](https://pytorch.org/hub/pytorch_vision_squeezenet/) (Left Image)

<https://www.kaggle.com/datasets/pytorch/squeezenet11> (Right Image)

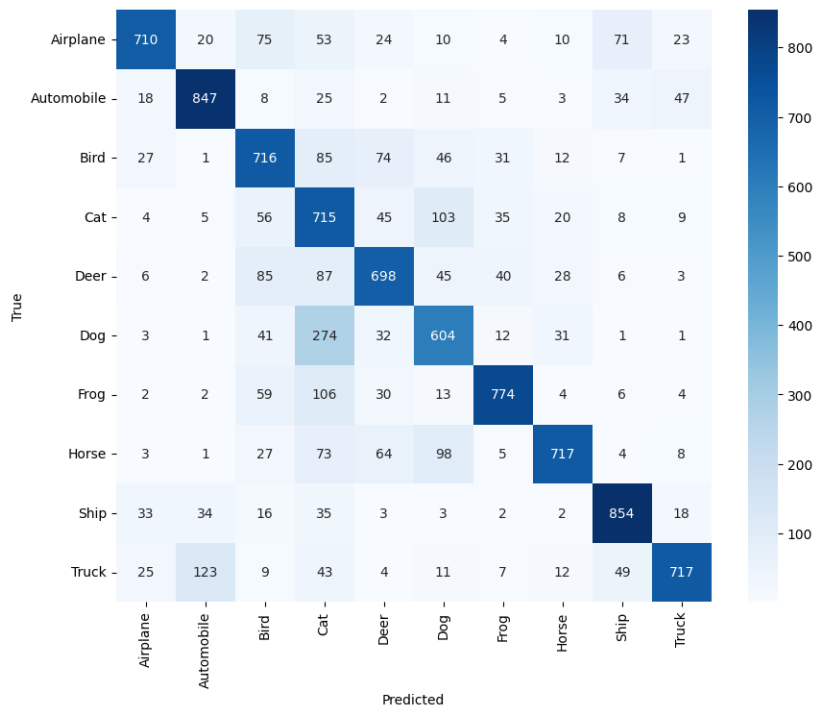
## 3 Output Analysis

### 3.1 Default Model Results

Here are the required metrics to be reported:

- Storage Size: 2.86MB
- Test Accuracy: 73.57
- Training Time: 682.71 Seconds
- Test Time: 1.53 Seconds

- Confusion Matrix: See Image Below



Just some quick observations from the confusion matrix we can see the model does a fairly good job on most categories performing strongest on ships and automobiles while doing the worst on dogs. Let's see how the other models compare to the base model.

### 3.2 Other models

I decided to train 10 more models varying the parameters of batch size and learning rate while keeping all other parameters the same as the default model. I tested batch size in the search space of [16, 32, 128, 256, 512] and learning rate in the search space of [.0002, .00005]. The reason the learning rate is such a low value is because I tried high ones such as .001, however, the model stagnated at 10% accuracy showing that .001 was too large of a learning rate and I needed to use something smaller. There are 5 batch configurations x 2 learning rate configurations = 10 models + base model = 11 total models. I wanted to get more batch sizes I could also graph and better show the effects of changing batch size while keeping other parameters the same. The overall results of all the models are detailed below in the table and further shown in the confusion matrices printed below along with some graphs showing the effect of increasing batch size with fixed LR (2 graphs since we fixed LR to 2 different values).

Table 1: Model Performance Metrics

Model*	Train Time(s)	Test Time(s)	Train Accuracy (%)	Test Accuracy (%)	Batch Size	Learning Rate
Default	682.71	1.53	87.11%	73.57%	64	.0001
Model 1	2313.26	3.62	93.59%	75.52%	16	.0002
Model 2	1240.33	2.17	93.24%	75.26%	32	.0002
Model 3	420.47	1.24	87.02%	72.86%	128	.0002
Model 4	286.51	1.19	84.20%	73.87%	256	.0002
Model 5	231.00	1.03	79.30%	74.32%	512	.0002
Model 6	2299.58	3.64	88.89%	75.15%	16	.00005
Model 7	1235.05	2.14	85.09%	73.85%	32	.00005
Model 8	417.44	1.25	77.30%	72.37%	128	.00005
Model 9	286.53	1.10	73.79%	71.40%	256	.00005
Model 10	232.06	1.04	69.48%	68.60%	512	.00005

\*ALL Models have the same size of 2.86MB since they all have the SAME number of parameters, not listed on the table since it would not fit and it's the same value so they don't add much to the analysis

### 3.3 Bunch of Confusion Matrices

Model 1:

True	Airplane	758	10	82	21	14	7	5	13	74	16
	Automobile	30	782	18	15	3	4	13	6	58	71
	Bird	30	1	778	45	55	30	29	20	9	3
	Cat	16	6	91	601	47	125	41	43	18	12
	Deer	13	0	83	59	710	38	30	53	10	4
	Dog	10	0	69	191	37	625	20	40	5	3
	Frog	8	2	56	78	24	20	798	7	3	4
	Horse	12	2	33	39	48	39	2	815	4	6
	Ship	40	10	16	20	2	8	2	6	883	13
	Truck	50	53	11	20	4	3	4	19	40	796
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

Model 2:

True	Airplane	826	10	41	20	10	3	15	14	38	23
	Automobile	21	777	5	17	2	5	16	4	48	105
	Bird	51	0	664	64	69	51	67	21	7	6
	Cat	22	4	41	588	46	167	79	31	10	12
	Deer	15	1	54	55	717	44	62	43	5	4
	Dog	16	2	29	172	33	681	24	38	0	5
	Frog	8	4	27	56	23	16	852	6	2	6
	Horse	21	2	16	46	47	75	7	778	2	6
	Ship	88	14	11	25	5	3	12	4	811	27
	Truck	33	50	3	21	3	5	4	18	26	837
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

Model 3:

True	Airplane	880	42	16	16	3	3	3	6	22	9
	Automobile	23	927	0	2	0	1	1	4	12	30
	Bird	123	20	560	74	65	78	45	25	5	5
	Cat	44	36	25	614	44	141	25	40	11	20
	Deer	58	16	33	73	640	54	26	89	5	6
	Dog	20	11	18	180	21	683	6	48	4	9
	Frog	19	37	26	87	34	39	730	7	2	19
	Horse	43	14	10	47	34	54	3	770	2	23
	Ship	105	73	4	11	1	6	2	5	751	42
	Truck	65	211	1	9	0	4	2	6	11	691
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									



Model 4:

True	Airplane	791	21	64	20	11	13	8	18	30	24
	Automobile	15	869	4	14	0	4	5	5	14	70
	Bird	43	4	670	77	43	63	59	27	4	10
	Cat	11	8	47	574	25	206	50	57	8	14
	Deer	11	4	86	78	582	59	65	106	4	5
	Dog	7	4	30	158	24	693	11	69	0	4
	Frog	5	7	32	76	13	30	821	9	2	5
	Horse	7	2	20	58	16	65	7	815	2	8
	Ship	75	63	19	25	6	6	10	8	750	38
	Truck	35	101	7	15	2	6	4	14	14	802
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

Model 5:

True	Airplane	792	17	54	20	17	5	5	17	53	20
	Automobile	19	867	10	6	0	5	6	3	25	59
	Bird	44	2	694	44	79	39	54	31	9	4
	Cat	28	7	72	554	68	137	48	57	21	8
	Deer	17	1	80	51	682	25	43	89	10	2
	Dog	9	3	72	175	40	605	16	71	5	4
	Frog	9	5	57	68	36	17	786	10	8	4
	Horse	13	3	25	52	36	43	4	804	2	18
	Ship	53	37	13	15	8	3	2	2	849	18
	Truck	38	110	5	12	4	5	3	21	46	756
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

Model 6:

True	Airplane	828	22	35	18	20	3	3	20	40	11
	Automobile	28	887	6	7	1	1	4	4	23	39
	Bird	49	3	729	44	60	40	37	28	6	4
	Cat	16	12	87	574	43	146	50	54	9	9
	Deer	15	3	85	47	666	40	51	83	5	5
	Dog	8	2	52	152	30	660	14	78	2	2
	Frog	11	8	50	79	21	18	800	6	2	5
	Horse	8	5	21	42	39	46	3	825	4	7
	Ship	71	41	17	16	4	5	3	6	821	16
	Truck	42	140	7	8	7	8	0	24	34	730
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

Model 7:

True	Airplane	744	13	68	32	29	5	2	19	73	15
	Automobile	28	824	15	12	2	7	8	8	48	48
	Bird	32	0	688	44	107	66	34	19	8	2
	Cat	7	3	87	544	83	176	41	41	8	10
	Deer	6	2	48	37	772	44	35	52	2	2
	Dog	6	1	51	133	47	708	12	40	0	2
	Frog	5	2	65	63	49	32	771	7	5	1
	Horse	6	1	19	42	79	79	3	764	5	2
	Ship	33	30	18	18	21	5	2	1	854	18
	Truck	39	98	16	24	7	10	9	31	58	708
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

Model 8:

True	Airplane	793	34	31	19	12	4	2	17	52	36
	Automobile	16	838	5	8	0	5	6	2	18	102
	Bird	69	8	582	75	86	61	62	41	6	10
	Cat	22	8	52	552	44	165	49	62	17	29
	Deer	34	3	64	66	615	44	44	115	8	7
	Dog	9	5	38	186	28	626	9	86	3	10
	Frog	9	8	38	84	26	26	777	14	7	11
	Horse	18	5	19	56	30	49	3	793	1	26
	Ship	74	55	5	21	8	4	1	4	774	54
	Truck	28	96	4	7	1	8	1	18	17	820
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

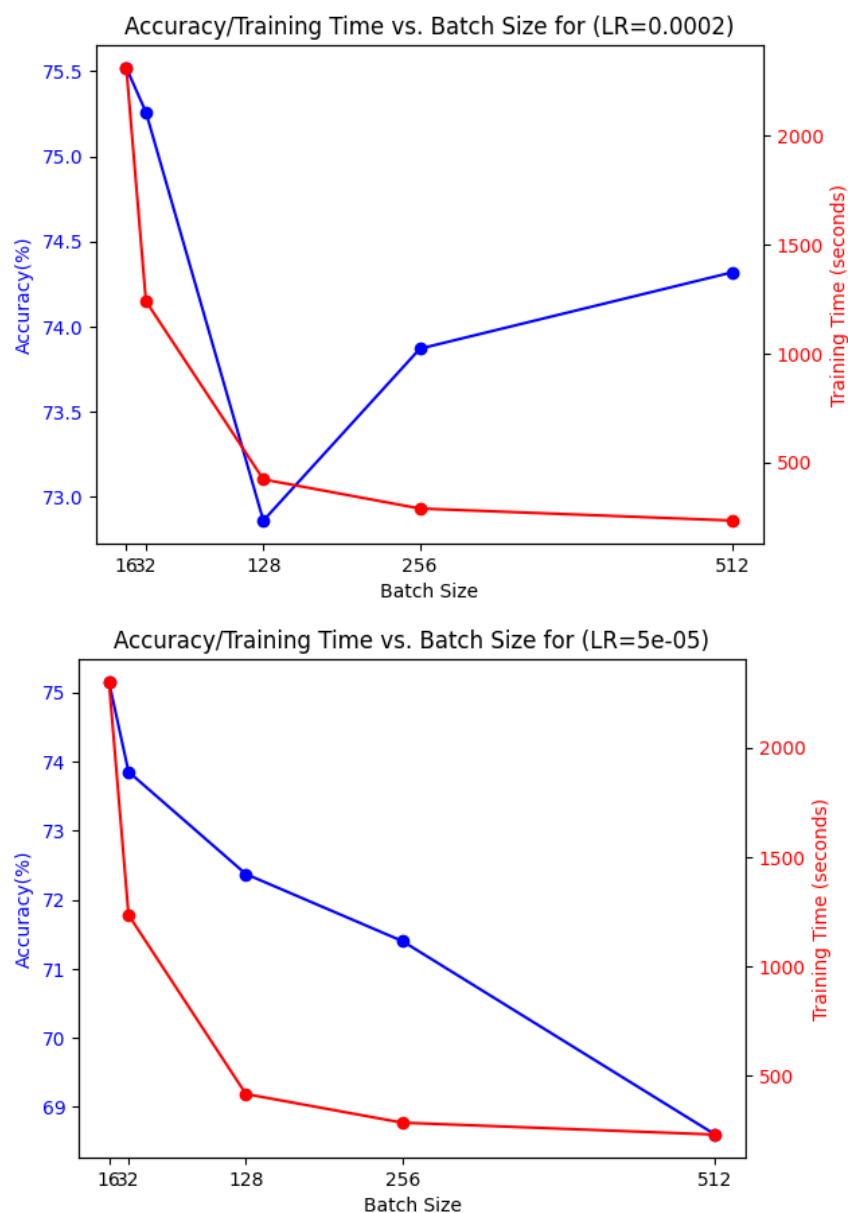
Model 9:

True	Airplane	736	27	36	16	27	2	9	20	90	37
	Automobile	25	790	3	7	0	4	10	6	34	121
	Bird	59	7	576	74	92	48	82	43	8	11
	Cat	20	7	69	522	59	139	75	63	24	22
	Deer	19	3	55	64	642	26	75	98	12	6
	Dog	9	3	50	216	38	563	26	79	6	10
	Frog	3	8	31	73	35	18	802	12	8	10
	Horse	12	5	27	61	58	45	7	758	2	25
	Ship	46	43	4	20	9	3	1	6	831	37
	Truck	16	76	4	15	4	3	7	19	41	815
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

Model 10:

True	Airplane	725	31	80	17	15	7	5	24	67	29
	Automobile	32	769	6	6	1	8	4	10	30	134
	Bird	72	5	599	63	76	76	63	32	5	9
	Cat	16	14	105	483	52	185	60	52	19	14
	Deer	34	1	115	75	567	55	54	87	6	6
	Dog	6	5	66	175	40	616	15	68	4	5
	Frog	9	7	73	85	43	30	729	12	5	7
	Horse	16	5	46	66	66	101	10	662	2	26
	Ship	83	40	19	27	10	1	2	3	780	35
	Truck	34	106	4	20	6	9	2	18	41	760
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
		Predicted									

### 3.4 Effects of Batch Size on 2 different Learning Rates



The first 2 tick marks are supposed to read "16" and "32", not "1632" They are just too close in value to properly scale

### 3.5 Analysis/Observations

- Based on the above confusion matrices we can see the class the the models had the most trouble with was Cat, although notably, the default model had the most issues with the class Dog. Also, Model 3 was the only one to have the most issues with class Bird. (By most issues I mean the class that has the lowest diagonal value/highest number of misclassifications)
- The class that was most correctly classified varies from model to model, the top classes for most classes were usually either ship or automobile. Although a lot of them seemed to do well on Airplanes, Trucks, Ships, and Automobiles. Notably Model 2 was the only model to have Frog as it's best classified category.

- The confusion matrices although provide a good measure of where the model is messing up and where it is doing well, overall it seems to do better on the non-animal categories (Airplane, Automobile, Ship, and Truck) and worse on the animals (Bird, Cat, Deer, Dog, Frog, Horse) and this can be attributed to the fact that animals may look more similar and also share a more similar color scheme making it harder for the model to differentiate these classes. We can see this trend on all the confusion matrices as the shading of the middle diagonal on the animal classes is on average lighter than the non-animal classes
- The most commonly misclassified classes were mistaking a dog for a cat and visa versa, followed by automobile vs. truck. Both of these logically make some sense they both can look somewhat similar, especially to an image net.
- Now looking at Table 1 we can see that batch size heavily influences runtime, the higher the batch size the lower the runtime, this logically makes since the backpropagation is the most computationally expensive step when training a CNN. Increasing batch size reduces the amount of times you have to do backpropagation, therefore lowering the runtime. This trend is also clearly shown in the graph which shows a nice exponential decay curve of runtime vs. batch size. Since the batch size stops at 512 we cannot know if it continues to fall or if there is a point where runtime will increase again.
- In terms of LR we can see on average the LR of 0.0002 outperformed 0.00005, which suggests that the LR of 0.00005 is likely too small and it is taking too long to converge in just 30 epochs. Since we kept this parameter constant it may have simply not had enough time to properly converge. We can really see that in the case of Model 10, the combination of lower LR and high batch size made the model perform significantly worse than the other models. Increasing the batch size means we perform backpropagation less. Higher batch size = fewer weight updates, and lower LR = less movement per weight update, the combination of both of these shows in the accuracy of model 10 as it likely did not have enough movement in the weights to converge and get a better accuracy (like the other models displayed).
- Lower batch sizes on average performed better than higher value batch sizes and it is likely due to the reasons mentioned above
- When comparing to the default model its metrics seem to fall where expected. The runtime is between the batch size of 32 and 128 (which makes sense since 64 is in between those numbers). Its accuracy is between LR=.0002/Batch=128 and LR=.0002/Batch=32, which makes sense since its parameters are between that as well. Its accuracy is also between LR = .00005/Batch=32 and LR = .00005/Batch=128, which once again makes sense since its parameters are between those values.
- The relationship between accuracy and batch size is defined well on the LR = .00005 graph. This graph shows a clear downtrend where increasing batch size decreases accuracy. However, for the LR = .0002 graph, there doesn't seem to be a clear relationship as accuracy shoots down and then back up again which makes it hard to draw any concrete conclusions.
- Also notably the LR = .0002 models seem to overfit data more than the .00005 models since the difference between test/train accuracy was higher on average in comparison to LR = .00005 models.

## 4 Outlier Test Scenario

### 4.1 Images Used

This section asked us to use the initial model and see the model results on 10 out of domain images and see what the model classified them as. I chose to take 10 images of former President Obama, here are the images I used:



These images are scaled down to be 32 x 32 x 3 so that we keep data consistent. Also for analysis, it is important to note that images are in order for analysis, the indices for each image are the numerical value displayed above them.

## 4.2 Models Tested

I decided to train 5 different models (+ the default one). Here is the procedure I followed:

For the default model:

- 1 Created Test Set/Resized Images and Prepared Input
- 2 Loaded Model Weights from the previously trained default model
- 3 Test model on the created test set
- 4 Printed/Record Results

For the new different models:

- 1 Created Test Set/Resized Images and Prepared Input
- 2 Loaded Model from pre-trained weights
- 3 Convert the model to 10 class classifier (change last layer)
- 4 Fine tune model to dataset using training set (same used training set for default model 50,000 images)
- 5 Test model on the created test set
- 6 Printed/Record Results

The 6 used models 1 (default) + 5 (new models), along with model size (since too much to put in the table):

- squeezenet (default) - 2.86MB
- resnet18 - 42.65MB
- resnet50 - 89.75MB

- vgg11 - 8.53MB
- vgg19 - 491.36MB
- mobilenet - 532.57MB

The following Hyperparameters were chosen to finetune the models:

- Batch Size: 64
- Loss Function: Cross-Entropy Loss
- Learning Rate: 0.001
- Weight Decay: 0.0001
- Optimizer: Adam
- Epochs: 10

### 4.3 Results

The results are shown in the table below:

Model	Image 0	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7	Image 8	Image 9
Default*	Dog	Dog	Cat	Dog	Cat	Bird	Dog	Bird	Bird	Bird
resnet18	Dog	Cat	Dog	Cat	Dog	Bird	Dog	Cat	Dog	Cat
resnet50	Dog	Dog	Dog	Truck	Cat	Bird	Dog	Dog	Truck	Deer
mobilenet	Dog	Dog	Dog	Dog	Dog	Bird	Dog	Bird	Cat	Dog
vgg11	Bird	Dog	Dog	Dog	Dog	Bird	Dog	Dog	Dog	Dog
vgg19	Dog	Dog	Dog	Dog	Dog	Bird	Dog	Cat	Cat	Dog

Table 2: Image Classification Results

\*Default model is squeezenet

For full raw probability distribution refer to the following file:

raw\_K\_probs.txt

### 4.4 Analysis/Observations

- As shown by the table all the results were not the same, there was some variation in the results but the overwhelming majority of images were classified as "Dog" from the models as shown in the table above
- The reason they are different is that they are different models which will result in different probability distributions and ultimately different results
- For most models if you give the same input it will generate the same output because weights at test time are frozen meaning with a fixed input it will produce a fixed output. This can be changed by introducing some randomness at test time or modifying some hyperparameters at test time (such as temperature as mentioned in Ed Discussion post #96)
- Interestingly both resnet models had the highest variation of moving away from the consensus value (Dog in most cases). resnet18 had "cat" as output more than other models and resnet50 was the only one to classify anything as a "Deer" or "Truck". Interestingly "Truck" is the only non-animal classification. SqueezeNet seemed to guess "Bird" a lot more than the other models
- Image 5 was classified as "Bird" by ALL models (I don't really see but it is very interesting). Images 7 & 8 had the highest variation (Models classified as many different categories)



## 4.5 References

Used some helper methods from the prior assignment (train loop/image printouts):

<https://github.com/ShaanGil1/CS6220-HW1>

Used images from:

[https://pytorch.org/hub/pytorch\\_vision\\_squeezenet](https://pytorch.org/hub/pytorch_vision_squeezenet)

<https://www.kaggle.com/datasets/pytorch/squeezenet11>