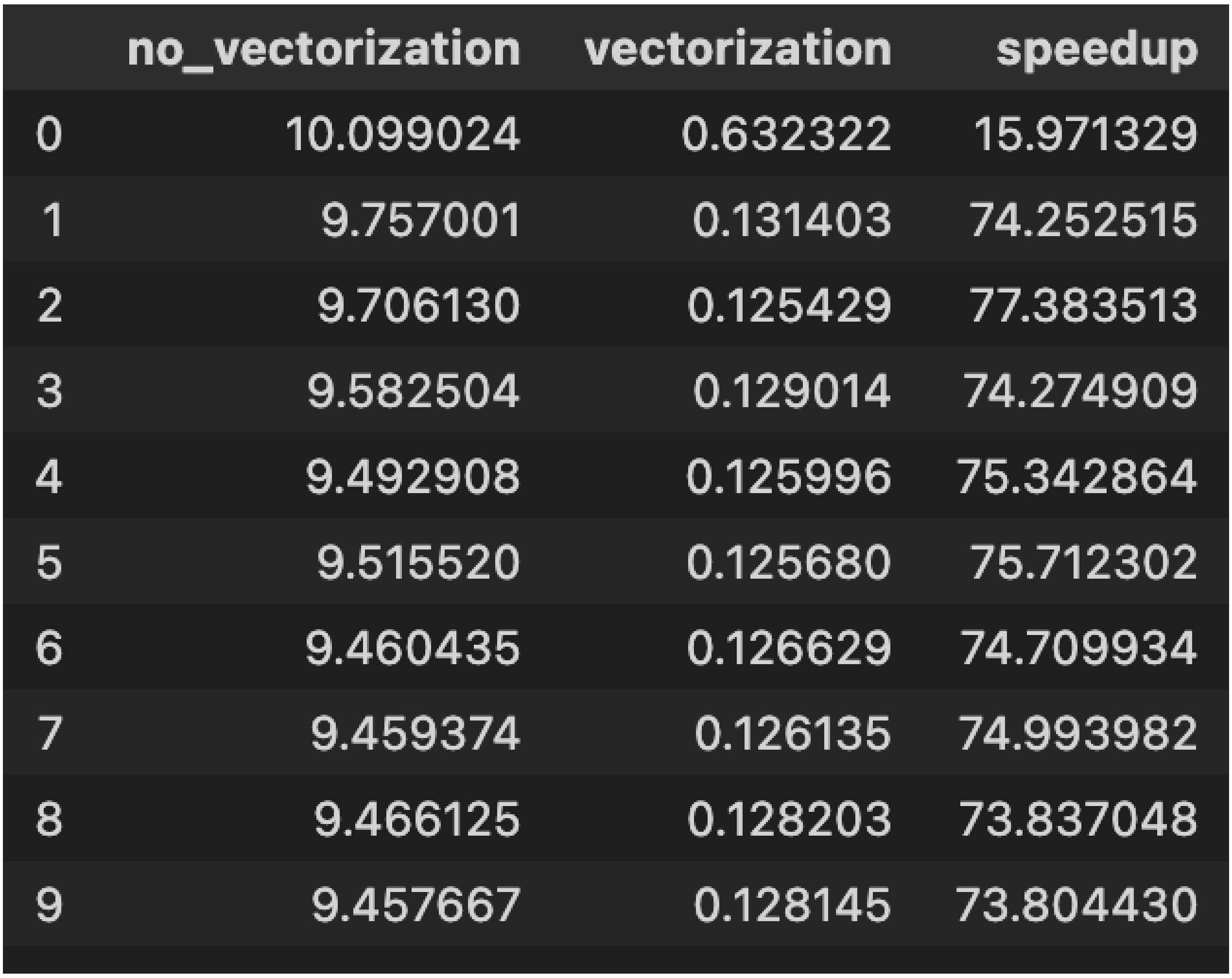
Lecture 3 Exercise

Note I use Apple Silicon M2

Exercise 1

I disigned to run process 10 times then use trimmean to compare and, for speedup, I used speedup each time to compute geometric mean
then the result is following.



	no_vectorization	vectorization	speedup
0	10.099024	0.632322	15.971329
1	9.757001	0.131403	74.252515
2	9.706130	0.125429	77.383513
3	9.582504	0.129014	74.274909
4	9.492908	0.125996	75.342864
5	9.515520	0.125680	75.712302
6	9.460435	0.126629	74.709934
7	9.459374	0.126135	74.993982
8	9.466125	0.128203	73.837048
9	9.457667	0.128145	73.804430

code

the result,

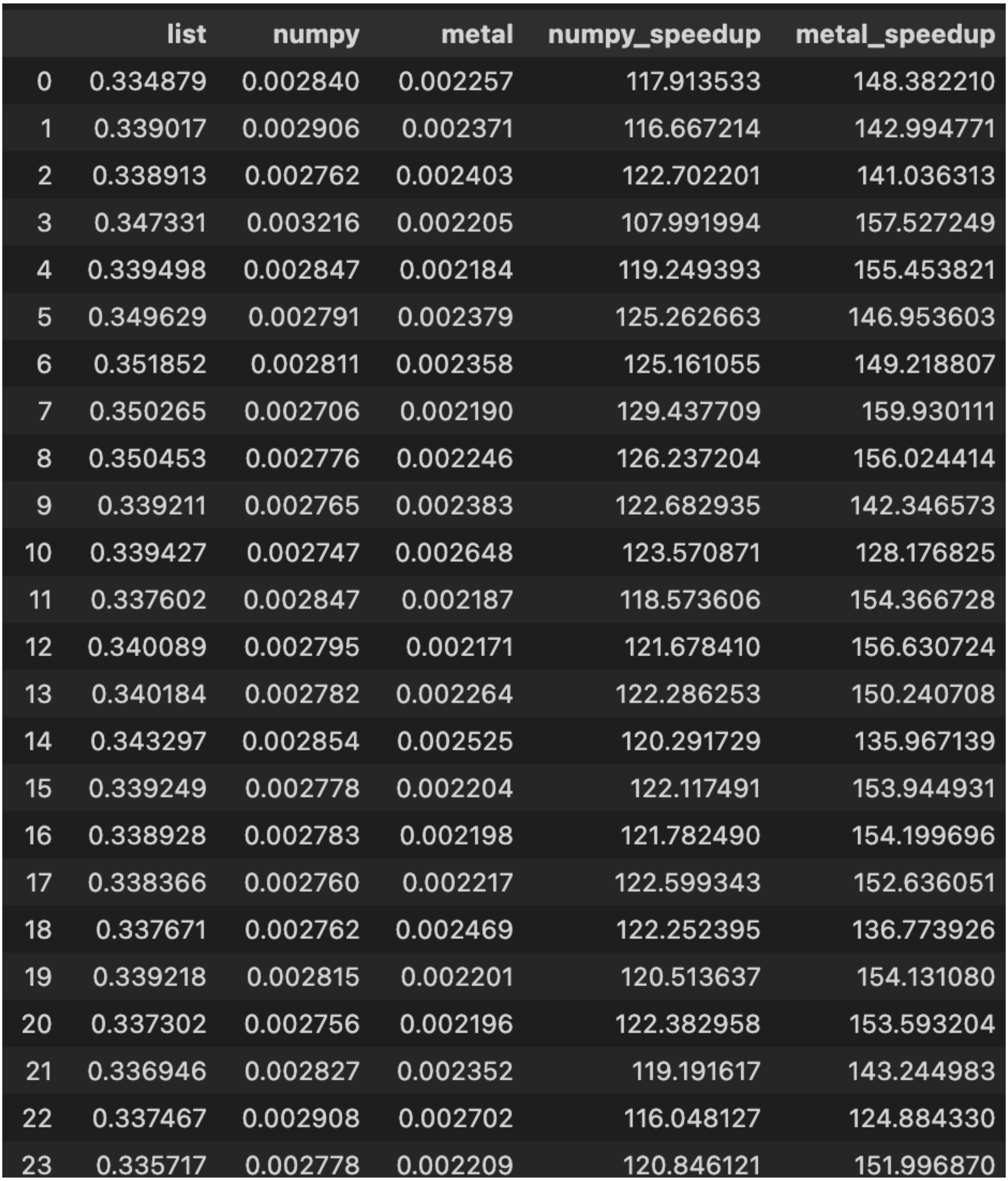
1. trimmean of `no vectorization` : 9.5550 sec.
2. trimmean of `vectorization` : 0.1277 sec.
3. speedup mean : 64.1876 times

Exercise 2

I designed to compare performance between **normal list, numpy and torch with metal**. I run each approach 51 times then use trimmean to compare and, for speedup, I used speedup each time to

compute geometric mean
then the result is following.

example first 24 times



	list	numpy	metal	numpy_speedup	metal_speedup
0	0.334879	0.002840	0.002257	117.913533	148.382210
1	0.339017	0.002906	0.002371	116.667214	142.994771
2	0.338913	0.002762	0.002403	122.702201	141.036313
3	0.347331	0.003216	0.002205	107.991994	157.527249
4	0.339498	0.002847	0.002184	119.249393	155.453821
5	0.349629	0.002791	0.002379	125.262663	146.953603
6	0.351852	0.002811	0.002358	125.161055	149.218807
7	0.350265	0.002706	0.002190	129.437709	159.930111
8	0.350453	0.002776	0.002246	126.237204	156.024414
9	0.339211	0.002765	0.002383	122.682935	142.346573
10	0.339427	0.002747	0.002648	123.570871	128.176825
11	0.337602	0.002847	0.002187	118.573606	154.366728
12	0.340089	0.002795	0.002171	121.678410	156.630724
13	0.340184	0.002782	0.002264	122.286253	150.240708
14	0.343297	0.002854	0.002525	120.291729	135.967139
15	0.339249	0.002778	0.002204	122.117491	153.944931
16	0.338928	0.002783	0.002198	121.782490	154.199696
17	0.338366	0.002760	0.002217	122.599343	152.636051
18	0.337671	0.002762	0.002469	122.252395	136.773926
19	0.339218	0.002815	0.002201	120.513637	154.131080
20	0.337302	0.002756	0.002196	122.382958	153.593204
21	0.336946	0.002827	0.002352	119.191617	143.244983
22	0.337467	0.002908	0.002702	116.048127	124.884330
23	0.335717	0.002778	0.002209	120.846121	151.996870

code

the result,

1. trimmean of **normal list** : 0.3402 sec.
2. trimmean of **numpy** : 0.0028 sec.
3. trimmean of **metal torch** : 0.0023 sec.

4. **numpy** speedup : 121.7043 times
5. **metal** speedup : 148.4287 times

Exercise 3

In my opinion, I think the situation that vectorization will be bad is the program that compute intensely but work on small data **bs.** we need time to move data to from CPU to GPU memery and the data is too small fill the warp fully so the computation will lose time for overhead more than computing.

Another situation is software may update lately to support new hardware bs. there is no general language that compatible with many hardware e.g. CUDA and OpenCL that support different hardware and not able to cross compile to another.