

Assignment 2: Vision Transformer Debugging Report

October 2, 2025

Introduction

This report documents the step-by-step debugging of a Vision Transformer (ViT) in PyCharm. Each snapshot shows the code location, the current tensor values, and shapes, with a brief description and an explanation connecting the tensor to the ViT pipeline.

Snapshots

1 Environment Setup (PyCharm + WSL)

1.1 Platform

- **Host OS:** Windows 10/11 with WSL2 (Ubuntu 22.04).
- **PyCharm:** Professional/Community, Python interpreter configured to use the WSL environment.
- **Python/PyTorch:** Python 3.10+, PyTorch, torchvision, numpy, pillow.
- **GPU (if available):** NVIDIA CUDA.

1.2 Interpreter Configuration

1. In PyCharm: `Settings > Project Interpreter > Add > WSL`.
2. Install deps: `pip install torch torchvision torchaudio pillow numpy`.
3. Confirm device: `torch.cuda.is_available()`.

1.3 Debugger Settings

- Place breakpoints at each pipeline step (`# BREAKPOINT Sxx`).
- Use the Debugger Variables panel to inspect shapes/values.
- Do not use print statements.

2 Unique Input Image & Preprocessing

2.1 Preprocessing Details

- Convert to RGB, resize to 224×224 , normalize with ImageNet mean/std.
- Final tensor shape $(1, 3, 224, 224)$.



Figure 1: Original Input Image

S01 — Raw input tensor

```
212
213     def load_image_tensor(path: str) -> torch.Tensor: 1usage
214         img = Image.open(path).convert("RGB")
215         t = preprocess(img).unsqueeze(0) # (1,3,224,224)
216         return t
217
218     def main(): 1usage
219         # Load + preprocess
220         img_tensor = load_image_tensor(IMAGE_PATH)  img_tensor: tensor([[[[-0.2342, -0.2171, -0.2171, ..., -0.2684, -0.2856, -0.2856], [-0.2171, -0.2171, -0.1828, ..., -1.4843, 0.8448, 0.8448, ..., -0.8110, -0.8284, -0.8633], [-0.2171, -0.2171, -0.1999, ..., -0.2342, -0.2844, -0.8110, -0.8284, -0.8458], [-0.2171, -0.2171, -0.1828, ..., -1.4843, 0.8448, 0.8448, ..., -0.8110, -0.8284, -0.8633]]])
221         # BREAKPOINT S01 - Raw input image tensor (after preprocessing)
222
223         model = ViT().to(DEVICE)
224         img_tensor = img_tensor.to(DEVICE)
225
226         # Forward
227         outputs = model(img_tensor)
228
229     if __name__ == "__main__":
230         main()
231
```

Image

Description: Preprocessed input image as a tensor of shape $1 \times 3 \times 224 \times 224$.

Explanation: Image loaded with PIL, normalized, and batched. Channel-first layout (C, H, W) with batch size 1 is expected by PyTorch modules.

S02 — Patches (4D view)

Image

Description: Image split into non-overlapping patches before flattening; shown as $(B, \# \text{patches}, C, 16, 16)$ which corresponds to $1 \times 196 \times 3 \times 16 \times 16$.

Explanation: ‘unfold’ extracts 16×16 windows with stride 16. There are $14 \times 14 = 196$ patches.

S03 — Flattened patches

The screenshot shows the PyCharm IDE interface with the following details:

- Code Editor:** The main window displays the `vit_transformer.py` file. The code includes two classes: `PatchEmbed` and `EncoderBlock`. The `PatchEmbed` class handles patch extraction and projection. The `EncoderBlock` class is a VIT encoder block with exposed Q, K, V and attention math.
- Debug Bar:** A horizontal bar at the bottom provides quick access to common debugging functions like Step Into, Step Out, and Run.
- Toolbars:** Standard PyCharm toolbars for file operations, search, and navigation are visible along the top.
- Bottom Status Bar:** Shows the current file (`SheetAssignment2> vit_transformer.py`), line number (58), and Python version (Python 3.10).

Image

Description: Flattened patch vectors of shape $1 \times 196 \times 768$.

Explanation: Each $16 \times 16 \times 3$ patch becomes a vector of length 768, ready for the linear projection.

S04 — Patch embeddings

```
140     class ViT(nn.Module):  # usage
141         def __init__(self,
142             # (Optional) init
143             nn.init.trunc_normal_(self.pos_embed, std=0.02)
144             nn.init.trunc_normal_(self.cls_token, std=0.02)
145
146         def forward(self, x):
147             # ViT(1 comes from the preprocessed input tensor
148             # patches_4d, patches_flat, x_embed = self.patch_embed(x) # S02, S03, S04 handled inside x_embed: tensor([[0.3839, -0.3620, 0.3796, ..., -0.0338,
149
150             # Class token + pos embed
151             B = x.size(0)  # 1
152             cls_tok = self.cls_token.expand(B, -1, -1) # (B, 1, C) cls_tok: tensor([[0.1252e-03, 2.5835e-02, 7.0004e-03, -2.1143e-02, 3.2950e-02, 3.3457e-02, -4.0910e-02, -8.2322e-03, -1.5642e-02, 6.9770e-02, -9.8348e-03, 4.3902e-01, View as Array
153
154             # BREAKPOINT S06 - Class token BEFORE concatenation
155             x_tokens = torch.cat([cls_tok, x_embed], dim=1) # (B, 1+196, 768)
156
157             # BREAKPOINT S07 - Embeddings AFTER adding the class token
158
159             x_pos = x_tokens + self.pos_embed # (B, 197, 768)
160
161             # BREAKPOINT S07 - Embeddings AFTER adding positional encoding
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
417
418
419
419
420
421
422
423
423
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
```

S05 — Class token (before concat)

Image

Description: Learned [CLS] token expanded to batch: $1 \times 1 \times 768$.

Explanation: A trainable vector summarizes the entire sequence after attention; it will be used for classification.

S06 — Tokens after adding [CLS]

Image

Description: Sequence with class token prepended: $1 \times 197 \times 768$.

Explanation: Concatenate [CLS] (length 1) with the 196 patch tokens to form the transformer input sequence.

S07 — + Positional encoding

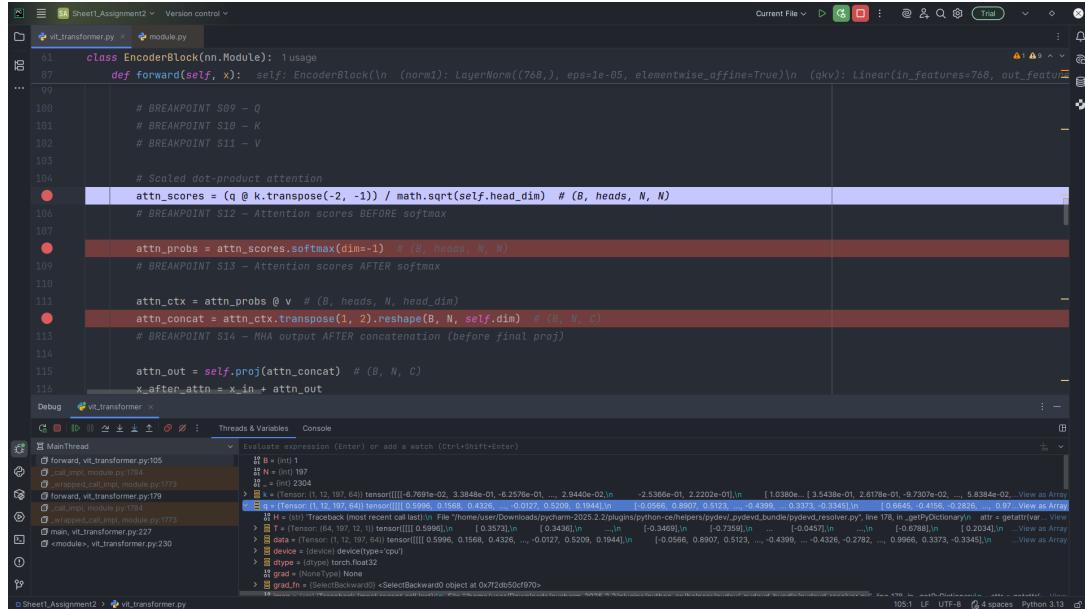
```
61     class EncoderBlock(nn.Module): 1 usage
62         def __init__(self, dim=EMBED_DIM, num_heads=NUM_HEADS, mlp_ratio=MLP_RATIO):
63             # Post-MLP norm (explicitly included to match the assignment's "post-MLP normalization" snapshot)
64             self.norm3 = nn.LayerNorm(dim)
65
66
67         def forward(self, x):  self: EncoderBlock(\n            norm1: LayerNorm(768,) , eps=1e-05, elementwise_affine=True)\n            (\n                qkv: Linear(in_features=768, out_features=768, \n                    bias=True)\n            )\n            (\n                x: Tensor(1, 197, 768) tensor([[[-7.909e-03, 4.252e-03, 5.148e-04, ..., 6.980e-02, 6.053e-02, 2.1865e-02], ...]\n                ]\n            )\n        )\n        # BREAKPOINT S08 - Encoder block input tensor\n        x_in = x\n        # ---- Multi-Head Self Attention ---- (PreNorm)\n        x_norm1 = self.norm1(x_in)\n        qkv = self.qkv(x_norm1) # (B, N, 3*C)\n        B, N, _ = qkv.shape\n\n        qkv = qkv.reshape(B, N, 3, self.num_heads, self.head_dim).permute(2, 0, 3, 1, 4)\n        q, k, v = qkv[0], qkv[1], qkv[2] # each: (B, heads, N, head_dim)\n\n        # BREAKPOINT S09 - q\n        # BREAKPOINT S10 - k
```

Image

Description: Token embeddings after adding positional encodings: $1 \times 197 \times 768$.

Explanation: Sinusoidal/learned positional vectors inject order information so attention can exploit spatial positions.

S08 — Block input (pre-norm)



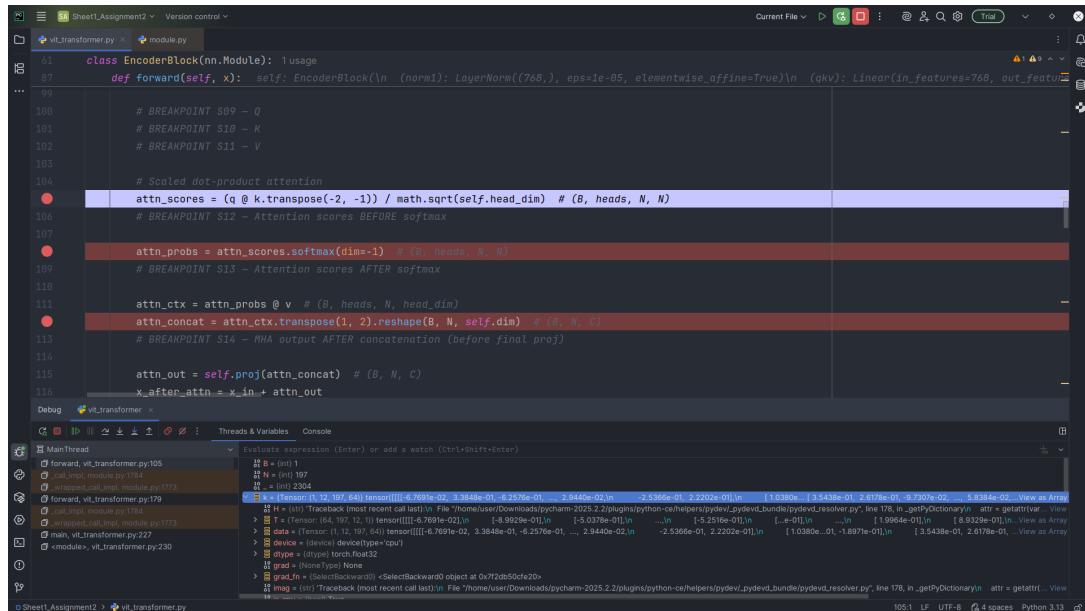
```
61     class EncoderBlock(nn.Module): 1 usage
87         def forward(self, x):  self: EncoderBlock\n          (norm1): LayerNorm((768,), eps=1e-05, elementwise_affine=True)\n          (qkv): Linear(in_features=768, out_features=768 * 3)\n...
99
100        # BREAKPOINT S09 - Q
101        # BREAKPOINT S10 - K
102        # BREAKPOINT S11 - V
103
104        # Scaled dot-product attention
105        attn_scores = (q @ k.transpose(-2, -1)) / math.sqrt(self.head_dim) # (B, heads, N, N)
106        # BREAKPOINT S12 - Attention scores BEFORE softmax
107
108        attn_probs = attn_scores.softmax(dim=-1) # (B, heads, N, N)
109        # BREAKPOINT S13 - Attention scores AFTER softmax
110
111        attn_ctx = attn_probs @ v # (B, heads, N, head_dim)
112        attn_concat = attn_ctx.transpose(1, 2).reshape(B, N, self.dim) # (B, N, C)
113        # BREAKPOINT S14 - MHA output AFTER concatenation (before final proj)
114
115        attn_out = self.proj(attn_concat) # (B, N, C)
116        x_after_attn = x_in + attn_out
```

Image

Description: Input to the first encoder block: $1 \times 197 \times 768$.

Explanation: Each block applies LayerNorm (Pre-LN), Multi-Head Self-Attention (MHSAs), residual, then MLP + residual.

S09 — Q (per head)



```
61     class EncoderBlock(nn.Module): 1 usage
87         def forward(self, x):  self: EncoderBlock\n          (norm1): LayerNorm((768,), eps=1e-05, elementwise_affine=True)\n          (qkv): Linear(in_features=768, out_features=768 * 3)\n...
99
100        # BREAKPOINT S09 - Q
101        # BREAKPOINT S10 - K
102        # BREAKPOINT S11 - V
103
104        # Scaled dot-product attention
105        attn_scores = (q @ k.transpose(-2, -1)) / math.sqrt(self.head_dim) # (B, heads, N, N)
106        # BREAKPOINT S12 - Attention scores BEFORE softmax
107
108        attn_probs = attn_scores.softmax(dim=-1) # (B, heads, N, N)
109        # BREAKPOINT S13 - Attention scores AFTER softmax
110
111        attn_ctx = attn_probs @ v # (B, heads, N, head_dim)
112        attn_concat = attn_ctx.transpose(1, 2).reshape(B, N, self.dim) # (B, N, C)
113        # BREAKPOINT S14 - MHA output AFTER concatenation (before final proj)
114
115        attn_out = self.proj(attn_concat) # (B, N, C)
116        x_after_attn = x_in + attn_out
```

Image

Description: Query tensor reshaped to heads: $(B, h, N, d_h) = 1 \times 12 \times 197 \times 64$.

Explanation: Linear projections split d_{model} into $h = 12$ heads with head dimension $d_h = 64$.

S10 — K (per head)

Image

Description: Key tensor $1 \times 12 \times 197 \times 64$.

Explanation: Keys are used with queries to compute attention compatibility scores.

S11 — V (per head)

Image

Description: Value tensor $1 \times 12 \times 197 \times 64$.

Explanation: Values are the information that will be aggregated based on the attention probabilities.

S12 — Attention scores (pre-softmax)

The screenshot shows the PyCharm IDE interface. The top part displays the code for the `EncoderBlock` class. The code includes several comments indicating breakpoints: `# BREAKPOINT S09 - Q`, `# BREAKPOINT S10 - K`, `# BREAKPOINT S11 - V`, `# Scaled dot-product attention`, `# BREAKPOINT S12 - Attention scores BEFORE softmax`, `# attn_probs = attn_scores.softmax(dim=-1) # (B, heads, N, N) attn_probs: tensor([[[[0.0047, 0.0072, 0.0075, ..., 0.0049, 0.0040, 0.0039], ...]`, `# BREAKPOINT S13 - Attention scores AFTER softmax`, `attn_ctx = attn_probs @ v # (B, heads, N, head_dim)`, `attn_concat = attn_ctx.transpose(1, 2).reshape(B, N, self.dim) # (B, N, C)`, `# BREAKPOINT S14 - MHA output AFTER concatenation (before final proj)`, `attn_out = self.proj(attn_concat) # (B, N, C)`, and `x_after_attn = x_in + attn_out`. The bottom part of the screenshot shows the `Debug` tool window, which lists the current thread and variables. It highlights the variable `attn_probs` with its value: `[Tensor(...)]`. The variable `attn_probs` is annotated with a tooltip: `attr = getAttr(... View as Array`.

Image

Description: Scaled dot-product scores $(B, h, N, N) = 1 \times 12 \times 197 \times 197$.

Explanation: Scores are computed as $qk^\top / \sqrt{d_h}$. These are raw compatibilities between all token pairs per head.

S13 — Attention probabilities

The screenshot shows the PyCharm IDE interface with the following details:

- File:** vit_transformer.py
- Breakpoints:** Several red circular markers are placed along the code, indicating where the debugger will stop.
- Code Snippet:** The code implements an EncoderBlock module. It includes operations like LayerNorm, Linear, transpose, reshape, and various tensor manipulations. Breakpoints are present in the attn_concat, x_after_attn, x_post_attn_norm, ff_in, ff_hidden, and main function sections.
- Toolbars:** Standard PyCharm toolbars for file operations, search, and navigation are visible at the top.
- Sidebar:** A sidebar on the left lists the current file and its imports.
- Bottom Status Bar:** Shows the current file as "vit_transformer.py", the line number as 115, and the status "LF" (Line First).

Image

Description: Softmax over the last dimension of scores: $1 \times 12 \times 197 \times 197$.

Explanation: Applying softmax along keys turns scores into probabilities; each row sums to 1 and indicates how much each token attends to others.

S14 — Context per head

Image

Description: Head-wise context $\text{attn_probs} \cdot V$ with shape $1 \times 12 \times 197 \times 64$.

Explanation: Probabilities weight the values to aggregate information from relevant tokens for each head.

S15 — MHA output (concat + proj)

Image

Description: Concatenated head contexts then projected: $\text{attn_out} \in 1 \times 197 \times 768$.

Explanation: Per-head contexts are concatenated along d_h to $197 \times (12 \cdot 64)$ and passed through a linear layer to return to d_model .

S16 — Residual + norm (post-attention)

```

61     class EncoderBlock(nn.Module): 1 usage
...
87         def forward(self, x): x: tensor([[[-7.9809e-03, 4.2521e-03, 5.1484e-04, ..., 6.9683e-02, ..., 6.0531e-02, 2.1865e-02], ...], [3.9172e-02, 1.1643e-01, 0.3994, 0.9296, ..., -1.1643, -0.3994, -0.9296, ..., -3.9172e-02]])
88             ff_in = x_post_attn_norm ff_in: tensor([[0.0756, 0.0789, -0.0845, ..., 0.2341, -0.3760, 0.7350], ...], [-1.1643, -0.3994, -0.9296, ..., -3.9172, 1.1643, 0.3994, 0.9296, ..., 3.9172e-02])
89             # BREAKPOINT S16 - Feed-forward input
90
91             ff_hidden = self.act(self.fc1(ff_in)) # (B, N, hidden) ff_hidden: tensor([[-0.0403, -0.1405, -0.1563, ..., 0.1003, 1.0271, -0.1084], ...], [0.1158, 0.3952, -0.1576, 0.4059, 0.51, 0.4984, 0.4919, 0.1089, 0.1489, 0.0947])
92             # BREAKPOINT S17 - Feed-forward hidden layer output
93
94             ff_out = self.fc2(ff_hidden) # (B, N, C)
95             # BREAKPOINT S18 - Feed-forward output after second linear
96
97             x_after_mlp = ff_in + ff_out
98             x_post_mlp_norm = self.norm3(x_after_mlp) # (B, N, C)
99             # BREAKPOINT S19 - Residual connection + normalization (post-MLP)
100
101             block_out = x_post_mlp_norm
102             # BREAKPOINT S20 - Encoder block final output
103
104             return block_out
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
915
916
916
917
917
918
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
```

Explanation: The first linear expands the channel dimension; GELU adds nonlinearity before the second linear brings it back.

S18 — FFN output (after fc2)

Image

Description: Feed-forward output $\text{ff_out} \in 1 \times 197 \times 768$.

Explanation: Second linear projects back to d model.

S19 — Residual + norm (post-MLP)

```
01 class EncoderBlock(nn.Module): 1 usage
02     def forward(self, x): x: tensor([[[-7.9898e-03, 4.2521e-03, 5.1484e-04, ..., 6.9603e-02, 6.0531e-02, 2.1865e-02], ... [ 3.9172e-02
03         x_post_mlp_norm = self.norm(x_after_mlp) # (B, N, C) x post-MLP norm: tensor([[[-0.1041, 0.8805, -0.8489, ..., 0.1518, -0.2089, 0.7738], ... [-0.0581, -0.5845, 0.8723,
04             # BREAKPOINT S19 - Residual connection + normalization (post-MLP)
05
06             block_out = x_post_mlp_norm + block_out: tensor([[[-0.1041, 0.8805, -0.8489, ..., 0.1518, -0.2089, 0.7738], ... [-0.0581, -0.5845, 0.8723,
07                 # BREAKPOINT S20 - Encoder block final output
08
09             return block_out
10
11
12
13
14 class ViT(nn.Module): 1 usage
15     def __init__(self,
16         img_size=IMG_SIZE,
17         patch_size=PATCH_SIZE,
18         in_chans=3,
19         embed_dim=EMBED_DIM,
20         num_heads=NUM_HEADS,
21         num_blocks=NUM_BLOCKS,
22         num_classes=NUM_CLASSES):
23
24
25
26
27
28
29
2
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
5
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
427
428
429
429
430
431
432
433
433
434
435
435
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
```

Image

Description: Post-MLP normalized tensor $x_post_mlp_norm \in 1 \times 197 \times 768$.

Explanation: Add `ff_out` to `ff_in` (residual) and apply LayerNorm. This is the block output.

S20 — Encoder block output

The screenshot shows a Jupyter Notebook interface with several tabs open. The main tab displays Python code for a ViT transformer module, with specific lines highlighted as 'BREAKPOINT' points. The code involves token embeddings, patch embeddings, and multiple encoder blocks. Below the code, there's a 'Debug' section and a 'Threads & Variables' panel.

At the bottom, a memory dump is being analyzed by the 'pydevd' extension. It lists various memory locations, including tensors for input images, position embeddings, and model parameters like weight matrices and bias vectors. The analysis includes memory addresses, sizes, and values for each tensor.

Image

Description: Final output of the shown encoder block: $1 \times 197 \times 768$.

Explanation: This tensor is fed to subsequent encoder blocks (or heads to classification if it is the last block).

S21 — Encoder block 2 output

Image

Description: Output after the second encoder block: $1 \times 197 \times 768$.

Explanation: Demonstrates propagation of features through deeper layers.

S22 — Encoder block N (last) output

The screenshot shows a Jupyter Notebook interface with several cells of Python code for a ViT transformer. The code includes imports, class definitions for `ViT`, and various methods like `forward`, `final_seq`, and `cls_rep`. A red dot marks a breakpoint at line 180. A stack trace is visible at the bottom, indicating a call to `getPyDictionary` from `pydevd_resolver.py`.

```
140     class ViT(nn.Module): 1 usage
141         def forward(self, x):
142             self: ViT(in_(patch_embed): PatchEmbed\n                (proj): Linear(in_features=768, out_features=768, bias=True)\n            )\n            (blocks): ModuleList\n                (blocks0: nn.Sequential\n                    (x_block1_out: self.blocks[0](x, pos)\n                        # 508...S20 inside the block\n                        x_block1_out: tensor([[ 0.1604,  0.8005, -0.8469, ...,  0.1518, -0.2089,\n\n# BREAKPOINT S21 - Encoder block 2 output\n\nx_block2_out = self.blocks[1](x.block1_out) x.block2_out: tensor([[ [ 0.1268,  0.0050, -0.8176, ...,  0.0793, -0.1473,  0.9456],\n\n[ 1.1164,\n\nx_curr = x.block2_out x_curr: tensor([[[-0.1660,  0.9085, -2.2163, ...,  1.1964,  0.1596, -0.3823],\n\nfor i in range(2, len(self.blocks)): i: 11\n    x_curr = self.blocks[i](x_curr)\n\n# BREAKPOINT S22 - Encoder block N (last) output\n\nx_last = x_curr x.last: tensor([[-0.1660,  0.9085, -2.2163, ...,  1.1964,  0.1596, -0.3823],\n\n# BREAKPOINT S23 - Final sequence output (including class token)\n\nfinal_seq = x.last final_seq: tensor([[-0.1660,  0.9085, -2.2163, ...,  1.1964,  0.1596, -0.3823],\n\n# BREAKPOINT S24 - Class token extracted (final representation)\n\ncls_rep = final_seq[:, 0, :] # (B, 768)\n\nlogits = self.head(cls_rep) # (B, 1000)
```

Image

Description: Last encoder output $1 \times 197 \times 768$.

Explanation: This sequence holds the final token features used to compute the class representation.

S23 — Final sequence (incl. [CLS])

```
140     class ViT(nn.Module): 1usage
141         def forward(self, x):  self: ViT\n            (patch_embed): PatchEmbed\n                (proj): Linear(in_features=768, out_features=768, bias=True)\n                (blocks): ModuleList\n142\n143             x_curr = x_block2_out  x_curr: tensor([[-0.1660,  0.9085, -2.2163, ...,  1.1964,  0.1596, -0.3823],\n144                 for i in range(2, len(self.blocks)):  i: int\n145                 x_curr = self.blocks[i](x_curr)\n146\n147             # BREAKPOINT S22 - Encoder block N (last) output\n148             x_last = x_curr  x.last: tensor([[[-0.1660,  0.9085, -2.2163, ...,  1.1964,  0.1596, -0.3823],\n149                 [ 1.2968,  0.5651, -1.6427, ...,  0.8632],\n150\n151                 # BREAKPOINT S23 - Final sequence output (including class token)\n152                 final_seq = x_last  final_seq: tensor([[[-0.1660,  0.9085, -2.2163, ...,  1.1964,  0.1596, -0.3823],\n153                     [ 1.2968,  0.5651, -1.6427, ...,  0.8632],\n154\n155                 # BREAKPOINT S24 - Class token extracted (final representation)\n156                 cls_rep = final_seq[:, 0, :] # (B, 768)  cls.rep: tensor([[[-1.6595e-01,  9.0854e-01, -2.2163e+00,  1.2217e+00,\n157                     3.9433e-01],\n158\n159                 logits = self.head(cls_rep) # (B, 1000)\n160\n161             # BREAKPOINT S25 - Classification head logits\n162\n163             probs = logits.softmax(dim=1) # (B, 1000)\n164\n165\n166\n167\n168\n169\n170\n171\n172\n173\n174\n175\n176\n177\n178\n179\n180\n181\n182\n183\n184\n185\n186\n187\n188\n189\n190\n191\n192\n193\n194\n195\n196\n197\n198\n199\n200\n201\n202\n203\n204\n205\n206\n207\n208\n209\n210\n211\n212\n213\n214\n215\n216\n217\n218\n219\n220\n221\n222\n223\n224\n225\n226\n227\n228\n229\n230\n231\n232\n233\n234\n235\n236\n237\n238\n239\n240\n241\n242\n243\n244\n245\n246\n247\n248\n249\n250\n251\n252\n253\n254\n255\n256\n257\n258\n259\n260\n261\n262\n263\n264\n265\n266\n267\n268\n269\n270\n271\n272\n273\n274\n275\n276\n277\n278\n279\n280\n281\n282\n283\n284\n285\n286\n287\n288\n289\n290\n291\n292\n293\n294\n295\n296\n297\n298\n299\n300\n301\n302\n303\n304\n305\n306\n307\n308\n309\n310\n311\n312\n313\n314\n315\n316\n317\n318\n319\n320\n321\n322\n323\n324\n325\n326\n327\n328\n329\n330\n331\n332\n333\n334\n335\n336\n337\n338\n339\n340\n341\n342\n343\n344\n345\n346\n347\n348\n349\n350\n351\n352\n353\n354\n355\n356\n357\n358\n359\n360\n361\n362\n363\n364\n365\n366\n367\n368\n369\n370\n371\n372\n373\n374\n375\n376\n377\n378\n379\n380\n381\n382\n383\n384\n385\n386\n387\n388\n389\n389\n390\n391\n392\n393\n394\n395\n396\n397\n398\n399\n399\n400\n401\n402\n403\n404\n405\n406\n407\n408\n409\n409\n410\n411\n412\n413\n414\n415\n416\n417\n418\n419\n419\n420\n421\n422\n423\n424\n425\n426\n427\n428\n429\n429\n430\n431\n432\n433\n434\n435\n436\n437\n438\n439\n439\n440\n441\n442\n443\n444\n445\n446\n447\n448\n449\n449\n450\n451\n452\n453\n454\n455\n456\n457\n458\n459\n459\n460\n461\n462\n463\n464\n465\n466\n467\n468\n469\n469\n470\n471\n472\n473\n474\n475\n476\n477\n478\n479\n479\n480\n481\n482\n483\n484\n485\n486\n487\n488\n489\n489\n490\n491\n492\n493\n494\n495\n496\n497\n498\n499\n499\n500\n501\n502\n503\n504\n505\n506\n507\n508\n509\n509\n510\n511\n512\n513\n514\n515\n516\n517\n518\n519\n519\n520\n521\n522\n523\n524\n525\n526\n527\n528\n529\n529\n530\n531\n532\n533\n534\n535\n536\n537\n538\n539\n539\n540\n541\n542\n543\n544\n545\n546\n547\n548\n549\n549\n550\n551\n552\n553\n554\n555\n556\n557\n558\n559\n559\n560\n561\n562\n563\n564\n565\n566\n567\n568\n569\n569\n570\n571\n572\n573\n574\n575\n576\n577\n578\n579\n579\n580\n581\n582\n583\n584\n585\n586\n587\n588\n589\n589\n590\n591\n592\n593\n594\n595\n596\n597\n598\n599\n599\n600\n601\n602\n603\n604\n605\n606\n607\n608\n609\n609\n610\n611\n612\n613\n614\n615\n616\n617\n618\n619\n619\n620\n621\n622\n623\n624\n625\n626\n627\n628\n629\n629\n630\n631\n632\n633\n634\n635\n636\n637\n638\n639\n639\n640\n641\n642\n643\n644\n645\n646\n647\n648\n649\n649\n650\n651\n652\n653\n654\n655\n656\n657\n658\n659\n659\n660\n661\n662\n663\n664\n665\n666\n667\n668\n669\n669\n670\n671\n672\n673\n674\n675\n676\n677\n678\n679\n679\n680\n681\n682\n683\n684\n685\n686\n687\n688\n689\n689\n690\n691\n692\n693\n694\n695\n696\n697\n698\n699\n699\n700\n701\n702\n703\n704\n705\n706\n707\n708\n709\n709\n710\n711\n712\n713\n714\n715\n716\n717\n718\n719\n719\n720\n721\n722\n723\n724\n725\n726\n727\n728\n729\n729\n730\n731\n732\n733\n734\n735\n736\n737\n738\n739\n739\n740\n741\n742\n743\n744\n745\n746\n747\n748\n749\n749\n750\n751\n752\n753\n754\n755\n756\n757\n758\n759\n759\n760\n761\n762\n763\n764\n765\n766\n767\n768\n769\n769\n770\n771\n772\n773\n774\n775\n776\n777\n778\n779\n779\n780\n781\n782\n783\n784\n785\n786\n787\n788\n789\n789\n790\n791\n792\n793\n794\n795\n796\n797\n798\n799\n799\n800\n801\n802\n803\n804\n805\n806\n807\n808\n809\n809\n810\n811\n812\n813\n814\n815\n816\n817\n818\n819\n819\n820\n821\n822\n823\n824\n825\n826\n827\n828\n829\n829\n830\n831\n832\n833\n834\n835\n836\n837\n838\n839\n839\n840\n841\n842\n843\n844\n845\n846\n847\n848\n849\n849\n850\n851\n852\n853\n854\n855\n856\n857\n858\n859\n859\n860\n861\n862\n863\n864\n865\n866\n867\n868\n869\n869\n870\n871\n872\n873\n874\n875\n876\n877\n878\n879\n879\n880\n881\n882\n883\n884\n885\n886\n887\n888\n889\n889\n890\n891\n892\n893\n894\n895\n896\n897\n898\n899\n899\n900\n901\n902\n903\n904\n905\n906\n907\n908\n909\n909\n910\n911\n912\n913\n914\n915\n916\n917\n918\n919\n919\n920\n921\n922\n923\n924\n925\n926\n927\n928\n929\n929\n930\n931\n932\n933\n934\n935\n936\n937\n938\n939\n939\n940\n941\n942\n943\n944\n945\n946\n947\n948\n949\n949\n950\n951\n952\n953\n954\n955\n956\n957\n958\n959\n959\n960\n961\n962\n963\n964\n965\n966\n967\n968\n969\n969\n970\n971\n972\n973\n974\n975\n976\n977\n978\n979\n979\n980\n981\n982\n983\n984\n985\n986\n987\n988\n989\n989\n990\n991\n992\n993\n994\n995\n996\n997\n998\n999\n999\n1000\n1001\n1002\n1003\n1004\n1005\n1006\n1007\n1008\n1009\n1009\n1010\n1011\n1012\n1013\n1014\n1015\n1016\n1017\n1018\n1019\n1019\n1020\n1021\n1022\n1023\n1024\n1025\n1026\n1027\n1028\n1029\n1029\n1030\n1031\n1032\n1033\n1034\n1035\n1036\n1037\n1038\n1039\n1039\n1040\n1041\n1042\n1043\n1044\n1045\n1046\n1047\n1048\n1049\n1049\n1050\n1051\n1052\n1053\n1054\n1055\n1056\n1057\n1058\n1059\n1059\n1060\n1061\n1062\n1063\n1064\n1065\n1066\n1067\n1068\n1069\n1069\n1070\n1071\n1072\n1073\n1074\n1075\n1076\n1077\n1078\n1079\n1079\n1080\n1081\n1082\n1083\n1084\n1085\n1086\n1087\n1088\n1089\n1089\n1090\n1091\n1092\n1093\n1094\n1095\n1096\n1097\n1098\n1099\n1099\n1100\n1101\n1102\n1103\n1104\n1105\n1106\n1107\n1108\n1109\n1109\n1110\n1111\n1112\n1113\n1114\n1115\n1116\n1117\n1118\n1119\n1119\n1120\n1121\n1122\n1123\n1124\n1125\n1126\n1127\n1128\n1129\n1129\n1130\n1131\n1132\n1133\n1134\n1135\n1136\n1137\n1138\n1139\n1139\n1140\n1141\n1142\n1143\n1144\n1145\n1146\n1147\n1148\n1149\n1149\n1150\n1151\n1152\n1153\n1154\n1155\n1156\n1157\n1158\n1159\n1159\n1160\n1161\n1162\n1163\n1164\n1165\n1166\n1167\n1168\n1169\n1169\n1170\n1171\n1172\n1173\n1174\n1175\n1176\n1177\n1178\n1179\n1179\n1180\n1181\n1182\n1183\n1184\n1185\n1186\n1187\n1188\n1189\n1189\n1190\n1191\n1192\n1193\n1194\n1195\n1195\n1196\n1197\n1198\n1199\n1199\n1200\n1201\n1202\n1203\n1204\n1205\n1206\n1207\n1208\n1209\n1209\n1210\n1211\n1212\n1213\n1214\n1215\n1216\n1217\n1218\n1219\n1219\n1220\n1221\n1222\n1223\n1224\n1225\n1226\n1227\n1228\n1229\n1229\n1230\n1231\n1232\n1233\n1234\n1235\n1236\n1237\n1238\n1239\n1239\n1240\n1241\n1242\n1243\n1244\n1245\n1246\n1247\n1248\n1249\n1249\n1250\n1251\n1252\n1253\n1254\n1255\n1256\n1257\n1258\n1259\n1259\n1260\n1261\n1262\n1263\n1264\n1265\n1266\n1267\n1268\n1269\n1269\n1270\n1271\n1272\n1273\n1274\n1275\n1276\n1277\n1278\n1279\n1279\n1280\n1281\n1282\n1283\n1284\n1285\n1286\n1287\n1288\n1289\n1289\n1290\n1291\n1292\n1293\n1294\n1295\n1296\n1297\n1298\n1299\n1299\n1300\n1301\n1302\n1303\n1304\n1305\n1306\n1307\n1308\n1309\n1309\n1310\n1311\n1312\n1313\n1314\n1315\n1316\n1317\n1318\n1319\n1319\n1320\n1321\n1322\n1323\n1324\n1325\n1326\n1327\n1328\n1329\n1329\n1330\n1331\n1332\n1333\n1334\n1335\n1336\n1337\n1338\n1339\n1339\n1340\n1341\n1342\n1343\n1344\n1345\n1346\n1347\n1348\n1349\n1349\n1350\n1351\n1352\n1353\n1354\n1355\n1356\n1357\n1358\n1359\n1359\n1360\n1361\n1362\n1363\n1364\n1365\n1366\n1367\n1368\n1369\n1369\n1370\n1371\n1372\n1373\n1374\n1375\n1376\n1377\n1378\n1379\n1379\n1380\n1381\n1382\n1383\n1384\n1385\n1386\n1387\n1388\n1389\n1389\n1390\n1391\n1392\n1393\n1394\n1395\n1396\n1397\n1397\n1398\n1399\n1400\n1401\n1402\n1403\n1404\n1405\n1406\n1407\n1408\n1409\n1409\n1410\n1411\n1412\n1413\n1414\n1415\n1416\n1417\n1418\n1419\n1419\n1420\n1421\n1422\n1423\n1424\n1425\n1426\n1427\n1428\n1429\n1429\n1430\n1431\n1432\n1433\n1434\n1435\n1436\n1437\n1438\n1439\n1439\n1440\n1441\n1442\n1443\n1444\n1445\n1446\n1447\n1448\n1449\n1449\n1450\n1451\n1452\n1453\n1454\n1455\n1456\n1457\n1458\n1459\n1459\n1460\n1461\n1462\n1463\n1464\n1465\n1466\n1467\n1468\n1469\n1469\n1470\n1471\n1472\n1473\n1474\n1475\n1476\n1477\n1478\n1479\n1479\n1480\n1481\n1482\n1483\n1484\n1485\n1486\n1487\n1488\n1489\n1489\n1490\n1491\n1492\n1493\n1494\n1495\n1496\n1497\n1498\n1499\n1499\n1500\n1501\n1502\n1503\n1504\n1505\n1506\n1507\n1508\n1509\n1509\n1510\n1511\n1512\n1513\n1514\n1515\n1516\n1517\n1518\n1519\n1519\n1520\n1521\n1522\n1523\n1524\n1525\n1526\n1527\n1528\n1529\n1529\n1530\n1531\n1532\n1533\n1534\n1535\n1536\n1537\n1538\n1539\n1539\n1540\n1541\n1542\n1543\n1544\n1545\n1546\n1547\n1548\n1549\n1549\n1550\n1551\n1552\n1553\n1554\n1555\n1556\n1557\n1558\n1559\n1559\n1560\n1561\n1562\n1563\n1564\n1565\n1566\n1567\n1568\n1569\n1569\n1570\n1571\n1572\n1573\n1574\n1575\n1576\n1577\n1578\n1579\n1579\n1580\n1581\n1582\n1583\n1584\n1585\n1586\n1587\n1588\n1589\n1589\n1590\n1591\n1592\n1593\n1594\n1595\n1596\n1597\n1598\n1598\n1599\n1599\n1600\n1601\n1602\n1603\n1604\n1605\n1606\n1607\n1608\n1609\n1609\n1610\n1611\n1612\n1613\n1614\n1615\n1616\n1617\n1618\n1619\n1619\n1620\n1621\n1622\n1623\n1624\n1625\n1626\n1627\n1628\n1629\n1629\n1630\n1631\n1632\n1633\n1634\n1635\n1636\n1637\n1638\n1639\n1639\n1640\n1641\n1642\n1643\n1644\n1645\n1646\n1647\n1648\n1649\n1649\n1650\n1651\n1652\n1653\n1654\n1655\n1656\n1657\n1658\n1659\n1659\n1660\n1661\n1662\n1663\n1664\n1665\n1666\n1667\n1668\n1669\n1669\n1670\n1671\n1672\n1673\n1674\n1675\n1676\n1677\n1678\n1679\n1679\n1680\n1681\n1682\n1683\n1684\n1685\n1686\n1687\n1688\n1689\n1689\n1690\n1691\n1692\n1693\n1694\n1695\n1696\n1697\n1698\n1698\n1699\n1699\n1700\n1701\n1702\n1703\n1704\n1705\n1706\n1707\n1708\n1709\n1709\n1710\n1711\n1712\n1713\n1714\n1715\n1716\n1717\n1718\n1719\n1719\n1720\n1721\n1722\n1723\n1724\n1725\n1726\n1727\n1728\n1729\n1729\n1730\n1731\n1732\n1733\n1734\n1735\n1736\n1737\n1738\n1739\n1739\n1740\n1741\n1742\n1743\n1744\n1745\n1746\n1747\n1748\n1749\n1749\n1750\n1751\n1752\n1753\n1754\n1755\n1756\n1757\n1758\n1759\n1759\n1760\n1761\n1762\n1763\n1764\n1765\n1766\n1767\n1768\n1769\n1769\n1770\n1771\n1772\n1773\n1774\n1775\n1776\n1777\n1778\n1779\n1779\n1780\n1781\n1782\n1783\n1784\n1785\n1786\n1787\n1788\n1789\n1789\n1790\n1791\n1792\n1793\n1794\n1795\n1796\n1797\n1798\n1798\n1799\n1799\n1800\n1801\n1802\n1803\n1804\n1805\n1806\n1807\n1808\n1809\n1809\n1810\n1811\n1812\n1813\n1814\n1815\n1816\n1817\n1818\n1819\n1819\n1820\n1821\n1822\n1823\n1824\n1825\n1826\n1827\n1828\n1829\n1829\n1830\n1831\n1832\n1833\n1834\n1835\n1836\n1837\n1838\n1839\n1839\n1840\n1841\n1842\n1843\n1844\n1845\n1846\n1847\n1848\n1849\n1849\n1850\n1851\n1852\n1853\n1854\n1855\n1856\n1857\n1858\n1859\n1859\n1860\n1861\n1862\n1863\n1864\n1865\n1866\n1867\n1868\n1869\n1869\n1870\n1871\n1872\n1873\n1874\n1875\n1876\n1877\n1878\n1879\n1879\n1880\n1881\n1882\n1883\n1884\n1885\n1886\n1887\n1888\n1889\n1889\n1890\n1891\n1892\n1893\n1894\n1895\n1896\n1897\n1898\n1898\n1899\n1899\n1900\n1901\n1902\n1903\n1904\n1905\n1906\n1907\n1908\n1909\n1909\n1910\n1911\n1912\n1913\n1914\n1915\n1916\n1917\n1918\n1919\n1919\n1920\n1921\n1922\n1923\n1924\n1925\n1926\n1927\n1928\n1929\n1929\n1930\n1931\n1932\n1933\n1934\n1935\n1936\n1937\n1938\n1939\n1939\n1940\n1941\n1942\n1943\n1944\n1945\n1946\n1947\n1948\n1949\n1949\n1950\n1951\n1952\n1953\n1954\n1955\n1956\n1957\n1958\n1959\n1959\n1960\n1961\n1962\n1963\n1964\n1965\n1966\n1967\n1968\n1969\n1969\n1970\n1971\n1972\n1973\n1974\n1975\n1976\n1977\n1978\n1979\n1979\n1980\n1981\n1982\n1983\n1984\n1985\n1986\n1987\n1988\n1989\n1989\n1990\n1991\n1992\n1993\n1994\n1995\n1996\n1997\n1998\n1998\n1999\n1999\n2000\n2001\n2002\n2003\n2004\n2005\n2006\n2007\n2008\n2009\n2009\n2010\n2011\n2012\n2013\n2014\n2015\n2016\n2017\n2018\n2019\n2019\n2020\n2021\n2022\n2023\n2024\n2025\n2026\n2027\n2028\n2029\n2029\n2030\n2031\n2032\n2033\n2034\n2035\n2036\n2037\n2038\n2039\n2039\n2040\n2041\n2042\n2043\n2044\n2045\n2046\n2047\n2048\n2049\n2049\n2050\n2051\n2052\n2053\n2054\n2055\n2056\n2057\n2058\n2059\n2059\n2060\n2061\n2062\n2063\n2064\n2065\n2066\n2067\n2068\n2069\n2069\n2070\n2071\n2072\n2073\n2074\n2075\n2076\n2077\n2078\n2079\n2079\n2080\n2081\n2082\n2083\n2084\n2085\n2086\n2087\n2088\n2089\n2089\n2090\n2091\n2092\n2093\n2094\n2095\n2096\n2097\n2098\n2098\n2099\n2099\n2100\n2101\n2102\n2103\n2104\n2105\n2106\n2107\n2108\n2109\n2109\n2110\n2111\n2112\n2113\n2114\n2115\n2116\n2117\n2118\n2119\n2119\n2120\n2121\n2122\n2123\n2124\n2125\n2126\n2127\n2128\n2129\n2129\n2130\n2131\n2132\n2133\n2134\n2135\n2136\n2137\n2138\n2139\n2139\n2140\n2141\n2142\n2143\n2144\n2145\n2146\n2147\n2148\n2149\n2149\n2150\n2151\n2152\n2153\n2154\n2155\n2156\n2157\n2158\n2159\n2159\n2160\n2161\n2162\n2163\n2164\n2165\n2166\n2167\n2168\n2169\n2169\n2170\n2171\n2172\n2173\n2174\n2175\n2176\n2177\n2178\n2179\n2179\n2180\n2181\n2182\n2183\n2184\n2185\n2186\n2187\n2188\n2189\n2189\n2190\n2191\n2192\n2193\n2194\n2195\n2196\n2197\n2198\n2198\n2199\n2199\n2200\n2201\n2202\n2203\n2204\n2205\n2206\n2207\n2208\n2209\n2209\n2210\n2211\n2212\n2213\n2214\n2215\n2216\n2217\n2218\n2219\n2219\n2220\n2221\n2222\n2223\n2224\n2225\n2226\n2227\n2228\n2229\n2229\n2230\n2231\n2232\n2233\n2234\n2235\n2236\n2237\n2238\n2239\n2239\n2240\n2241\n2242\n2243\n2244\n2245\n2246\n2247\n2248\n2249\n2249\n2250\n2251\n2252\n2253\n2254\n2255\n2256\n2257\n2258\n2259\n2259\n2260\n2261\n2262\n2263\n2264\n2265\n2266\n2267\n2268\n2269\n2269\n2270\n2271\n2272\n2273\n2274\n2275\n2276\n2277\n2278\n2279\n2279\n2280\n2281\n2282\n2283\n2284\n2285\n2286\n2287\n2288\n2289\n2289\n2290\n2291\n2292\n2293\n2294\n2295\n2296\n2297\n2298\n2298\n2299\n2299\n2300\n2301\n2302\n2303\n2304\n2305\n2306\n2307\n2308\n2309\n2309\n2310\n2311\n2312\n2313\n2314\n2315\n2316\n2317\n2318\n2319\n2319\n2320\n2321\n2322\n2323\n2324\n2325\n2326\n2327\n2328\n2329\n2329\n2330\n2331\n2332\n2333\n2334\n2335\n2336\n2337\n2338\n2339\n2339\n2340\n2341\n2342\n2343\n2344\n2345\n2346\n2347\n2348\n2349\n2349\n2350\n2351\n2352\n2353\n2354\n2355\n2356\n2357\n2358\n2359\n2359\n2360\n2361\n2362\n2363\n2364\n2365\n2366\n2367\n2368\n2369\n2369\n2370\n2371\n2372\n2373\n2374\n2375\n2376\n2377\n2378\n2379\n2379\n2380\n2381\n2382\n2383\n2384\n2385\n2386\n2387\n2388\n2389\n2389\n2390\n2391\n2392\n2393\n2394\n2395\n2396\n2397\n2398\n2398\n2399\n2399\n2400\n2401\n2402\n2403\n2404\n2405\n2406\n2407\n2408\n2409\n2409\n2410\n2411\n2412\n2413\n2414\n2415\n2416\n2417\n2418\n2419\n2419\n2420\n2421\n2422\n2423\n2424\n2425\n2426\n2427\n2428\n2429\n2429\n2430\n2431\n2432\n2433\n2434\n2435\n2436\n2437\n2438\n2439\n2439\n2440\n2441\n2442\n2443
```

Image

Description: Final sequence $1 \times 197 \times 768$ with the first token being [CLS].

Explanation: We will slice index 0 to obtain the class representation.

S24 — Class token (final rep)

Image

Description: Class token representation $\text{cls_rep} \in 1 \times 768$.

Explanation: This vector summarizes the image; it goes into the classification head.

S25 — Logits

The screenshot shows a Python debugger interface with the following details:

- Code View:** The file `vit_transformer.py` is open, showing the `ViT` class definition. The code includes several `# BREAKPOINT` annotations. The `forward` method is shown, which takes `self` and `x` as inputs and returns a dictionary of outputs including `"S02_patches4d"`, `"S03_patches_flat"`, `"S04_patch_embed"`, `"final_seq"`, `"cls_rep"`, `"logits"`, and `"probs"`.
- Stack Trace:** A stack trace is visible in the bottom left, showing the call stack from `forward` down to `__main__.vit_transformer.py:227`.
- Variables View:** The `Threads & Variables` tab is selected, displaying a list of tensors and their values. One tensor, `cls Tok`, is highlighted and expanded to show its full 1D array of values.

Image

Description: Classification logits 1×1000 .

Explanation: Linear head maps cls_rep to 1000 ImageNet classes. Values are unnormalized scores.

S26 — Softmax probabilities

Image

Description: Class probabilities 1×1000 .

Explanation: Softmax converts logits to probabilities; the argmax gives the predicted class while the values reflect model confidence.

3 Guiding Questions and Answers

1. Why must the image be split into patches before embedding?

Vision Transformers operate on sequences of tokens, not on raw pixels. Splitting an image into 16×16 patches converts the 2D grid into a sequence of 196 patch tokens. Each patch is flattened into a vector and later projected to the model dimension. This allows the transformer to treat image patches like words in a sentence, enabling self-attention across the entire image.

2. Why is a class token added, and how does it affect the shape?

The [CLS] token is a special learned vector prepended to the patch sequence. During training, it learns to aggregate information from all patches so it can represent the entire image. Adding it increases the sequence length from 196 to 197 tokens, changing the shape from $(1, 196, 768)$ to $(1, 197, 768)$.

3. Why are positional encodings needed in ViT?

Self-attention alone is permutation-invariant, meaning it does not know the order or spatial arrangement of tokens. Positional encodings (sinusoidal or learned vectors) inject spatial order into token embeddings so the model can understand where each patch came from in the original image.

4. Why do Q , K , V have the same dimensions, and how do attention weights scale with patch count?

Queries (Q), Keys (K), and Values (V) are all derived from the same embedding dimension d_{model} and projected to the same head dimension $d_h = 64$ for stability in the dot-product attention calculation. This ensures the scaled dot-product $QK^\top / \sqrt{d_h}$ is well-defined. Attention weights form matrices of shape $(N \times N)$ per head, so the cost scales quadratically $\mathcal{O}(N^2)$ with the number of tokens $N = 197$.

5. How do residual connections preserve shape consistency across encoder blocks?

Each encoder block uses residual (skip) connections that add the input back to the output of the sub-layer (MHA or MLP). Since both input and output share the same dimensionality (1, 197, 768), the residual ensures consistent shapes across all layers. LayerNorm also preserves the same shape, so the data flows smoothly through the network depth.

6. Why is only the class token used for the final classification?

The [CLS] token attends to all patch tokens and is optimized to summarize the entire image. Using only this single token reduces the classification head to a simple linear layer ($768 \rightarrow 1000$) instead of requiring pooling over all patches, making it efficient and accurate for image-level predictions.

4 Reflection

Debugging the Vision Transformer step by step in PyCharm gave me a much clearer understanding of how the model works internally. By stopping at each breakpoint and checking tensor shapes, I saw how the raw image becomes patches, how the [CLS] token is added, and how positional encodings give the model a sense of spatial order. Watching the Q , K , V tensors and attention scores in real time helped me understand how self-attention redistributes information across all tokens. I also realized the importance of residual connections for keeping shapes consistent throughout the network.

This process did not only confirm the theoretical structure of ViTs but also improved my debugging skills in PyCharm, such as setting effective breakpoints and extracting value slices directly from the debugger instead of relying on print statements. Overall, this assignment deepened my intuition for transformers and gave me practical habits I can reuse when working with other deep learning architectures.