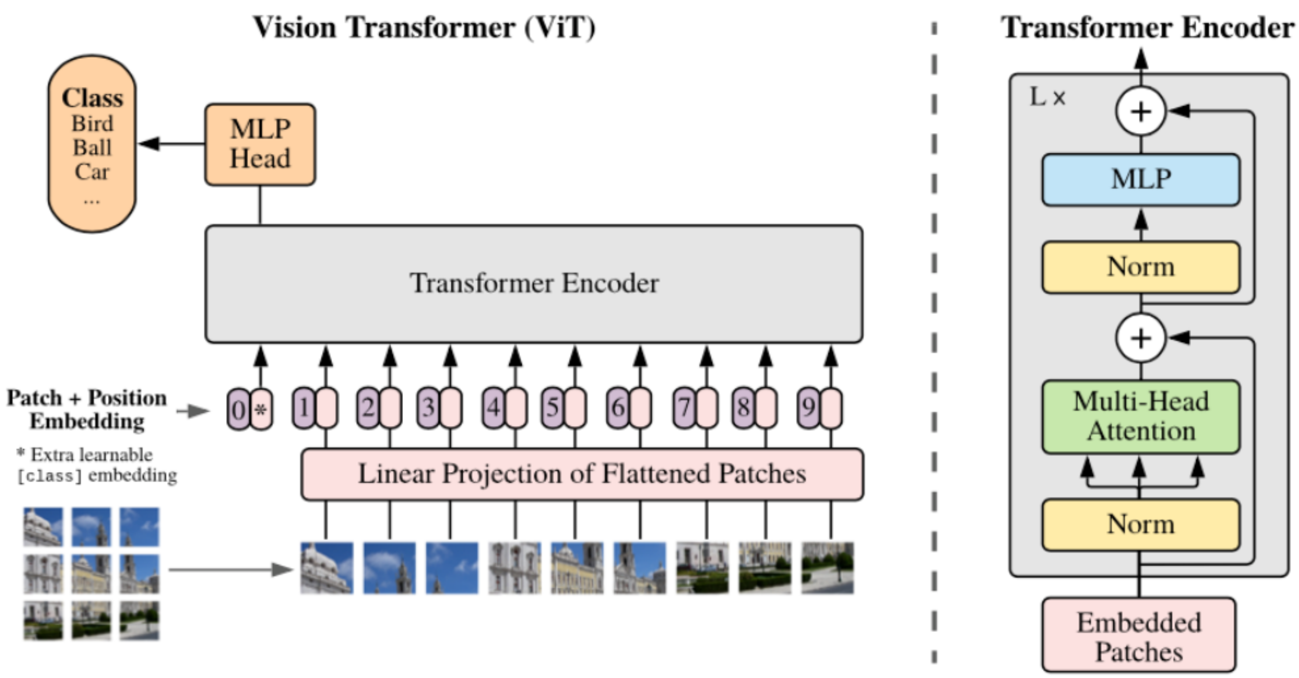
vision-transformers-assignment-1

March 2, 2025

[10]:
```
# prompt: unzip these files

!unzip /content/hymenoptera_data.zip
```

```
Archive:  /content/hymenoptera_data.zip
   creating: hymenoptera_data/train/
   creating: hymenoptera_data/train/ants/
  inflating: hymenoptera_data/train/ants/0013035.jpg
  inflating: hymenoptera_data/train/ants/1030023514_aad5c608f9.jpg
  inflating: hymenoptera_data/train/ants/1095476100_3906d8afde.jpg
  inflating: hymenoptera_data/train/ants/1099452230_d1949d3250.jpg
  inflating: hymenoptera_data/train/ants/116570827_e9c126745d.jpg
```




```
  inflating: hymenoptera_data/train/ants/178538489_bec7649292.jpg
  inflating: hymenoptera_data/train/ants/1804095607_0341701e1c.jpg
  inflating: hymenoptera_data/train/ants/1808777855_2a895621d7.jpg
  inflating: hymenoptera_data/train/ants/188552436_605cc9b36b.jpg
  inflating: hymenoptera_data/train/ants/1917341202_d00a7f9af5.jpg
  inflating: hymenoptera_data/train/ants/1924473702_daa9aacdbe.jpg
  inflating: hymenoptera_data/train/ants/196057951_63bf063b92.jpg
```

```
inflating: hymenoptera_data/train/ants/196757565_326437f5fe.jpg
inflating: hymenoptera_data/train/ants/201558278_fe4caecc76.jpg
inflating: hymenoptera_data/train/ants/201790779_527f4c0168.jpg
inflating: hymenoptera_data/train/ants/2019439677_2db655d361.jpg
inflating: hymenoptera_data/train/ants/207947948_3ab29d7207.jpg
inflating: hymenoptera_data/train/ants/20935278_9190345f6b.jpg
inflating: hymenoptera_data/train/ants/224655713_3956f7d39a.jpg
inflating: hymenoptera_data/train/ants/2265824718_2c96f485da.jpg
inflating: hymenoptera_data/train/ants/2265825502_fff99cfd2d.jpg
inflating: hymenoptera_data/train/ants/226951206_d6bf946504.jpg
inflating: hymenoptera_data/train/ants/2278278459_6b99605e50.jpg
inflating: hymenoptera_data/train/ants/2288450226_a6e96e8fdf.jpg
inflating: hymenoptera_data/train/ants/2288481644_83ff7e4572.jpg
inflating: hymenoptera_data/train/ants/2292213964_ca51ce4bef.jpg
inflating: hymenoptera_data/train/ants/24335309_c5ea483bb8.jpg
inflating: hymenoptera_data/train/ants/245647475_9523dfd13e.jpg
inflating: hymenoptera_data/train/ants/255434217_1b2b3fe0a4.jpg
inflating: hymenoptera_data/train/ants/258217966_d9d90d18d3.jpg
inflating: hymenoptera_data/train/ants/275429470_b2d7d9290b.jpg
inflating: hymenoptera_data/train/ants/28847243_e79fe052cd.jpg
inflating: hymenoptera_data/train/ants/318052216_84dff3f98a.jpg
inflating: hymenoptera_data/train/ants/334167043_cbd1adaeb9.jpg
inflating: hymenoptera_data/train/ants/339670531_94b75ae47a.jpg
inflating: hymenoptera_data/train/ants/342438950_a3da61deab.jpg
inflating: hymenoptera_data/train/ants/36439863_0bec9f554f.jpg
inflating: hymenoptera_data/train/ants/374435068_7eee412ec4.jpg
inflating: hymenoptera_data/train/ants/382971067_0bfd33afe0.jpg
inflating: hymenoptera_data/train/ants/384191229_5779cf591b.jpg
inflating: hymenoptera_data/train/ants/386190770_672743c9a7.jpg
inflating: hymenoptera_data/train/ants/392382602_1b7bed32fa.jpg
inflating: hymenoptera_data/train/ants/403746349_71384f5b58.jpg
inflating: hymenoptera_data/train/ants/408393566_b5b694119b.jpg
inflating: hymenoptera_data/train/ants/424119020_6d57481dab.jpg
inflating: hymenoptera_data/train/ants/424873399_47658a91fb.jpg
inflating: hymenoptera_data/train/ants/450057712_771b3bfc91.jpg
inflating: hymenoptera_data/train/ants/45472593_bfd624f8dc.jpg
inflating: hymenoptera_data/train/ants/459694881_ac657d3187.jpg
inflating: hymenoptera_data/train/ants/460372577_f2f6a8c9fc.jpg
inflating: hymenoptera_data/train/ants/460874319_0a45ab4d05.jpg
inflating: hymenoptera_data/train/ants/466430434_4000737de9.jpg
inflating: hymenoptera_data/train/ants/470127037_513711fd21.jpg
inflating: hymenoptera_data/train/ants/474806473_ca6caab245.jpg
inflating: hymenoptera_data/train/ants/475961153_b8c13fd405.jpg
inflating: hymenoptera_data/train/ants/484293231_e53cfc0c89.jpg
inflating: hymenoptera_data/train/ants/49375974_e28ba6f17e.jpg
inflating: hymenoptera_data/train/ants/506249802_207cd979b4.jpg
inflating: hymenoptera_data/train/ants/506249836_717b73f540.jpg
inflating: hymenoptera_data/train/ants/512164029_c0a66b8498.jpg
```

```
inflating: hymenoptera_data/train/ants/512863248_43c8ce579b.jpg
inflating: hymenoptera_data/train/ants/518773929_734dbc5ff4.jpg
inflating: hymenoptera_data/train/ants/522163566_fec115ca66.jpg
inflating: hymenoptera_data/train/ants/522415432_2218f34bf8.jpg
inflating: hymenoptera_data/train/ants/531979952_bde12b3bc0.jpg
inflating: hymenoptera_data/train/ants/533848102_70a85ad6dd.jpg
inflating: hymenoptera_data/train/ants/535522953_308353a07c.jpg
inflating: hymenoptera_data/train/ants/540889389_48bb588b21.jpg
inflating: hymenoptera_data/train/ants/541630764_dbd285d63c.jpg
inflating: hymenoptera_data/train/ants/543417860_b14237f569.jpg
inflating: hymenoptera_data/train/ants/560966032_988f4d7bc4.jpg
inflating: hymenoptera_data/train/ants/5650366_e22b7e1065.jpg
inflating: hymenoptera_data/train/ants/6240329_72c01e663e.jpg
inflating: hymenoptera_data/train/ants/6240338_93729615ec.jpg
inflating: hymenoptera_data/train/ants/649026570_e58656104b.jpg
inflating: hymenoptera_data/train/ants/662541407_ff8db781e7.jpg
inflating: hymenoptera_data/train/ants/67270775_e9fdf77e9d.jpg
inflating: hymenoptera_data/train/ants/6743948_2b8c096dda.jpg
inflating: hymenoptera_data/train/ants/684133190_35b62c0c1d.jpg
inflating: hymenoptera_data/train/ants/69639610_95e0de17aa.jpg
inflating: hymenoptera_data/train/ants/707895295_009cf23188.jpg
inflating: hymenoptera_data/train/ants/7759525_1363d24e88.jpg
inflating: hymenoptera_data/train/ants/795000156_a9900a4a71.jpg
inflating: hymenoptera_data/train/ants/822537660_caf4ba5514.jpg
inflating: hymenoptera_data/train/ants/82852639_52b7f7f5e3.jpg
inflating: hymenoptera_data/train/ants/841049277_b28e58ad05.jpg
inflating: hymenoptera_data/train/ants/886401651_f878e888cd.jpg
inflating: hymenoptera_data/train/ants/892108839_f1aad4ca46.jpg
inflating: hymenoptera_data/train/ants/938946700_ca1c669085.jpg
inflating: hymenoptera_data/train/ants/957233405_25c1d1187b.jpg
inflating: hymenoptera_data/train/ants/9715481_b3cb4114ff.jpg
inflating: hymenoptera_data/train/ants/998118368_6ac1d91f81.jpg
inflating: hymenoptera_data/train/ants/ant photos.jpg
inflating: hymenoptera_data/train/ants/Ant_1.jpg
inflating: hymenoptera_data/train/ants/army-ants-red-picture.jpg
inflating: hymenoptera_data/train/ants/formica.jpeg
inflating: hymenoptera_data/train/ants/hormiga_co_por.jpg
inflating: hymenoptera_data/train/ants/imageNotFound.gif
inflating: hymenoptera_data/train/ants/kurokusa.jpg
inflating: hymenoptera_data/train/ants/MehdiabadiAnt2_600.jpg
inflating: hymenoptera_data/train/ants/Nepenthes_rafflesiana_ant.jpg
inflating: hymenoptera_data/train/ants/swiss-army-ant.jpg
inflating: hymenoptera_data/train/ants/termite-vs-ant.jpg
inflating: hymenoptera_data/train/ants/trap-jaw-ant-insect-bg.jpg
inflating: hymenoptera_data/train/ants/VietnameseAntMimicSpider.jpg
 creating: hymenoptera_data/train/bees/
inflating: hymenoptera_data/train/bees/1092977343_cb42b38d62.jpg
inflating: hymenoptera_data/train/bees/1093831624_fb5fbe2308.jpg
```

```
inflating: hymenoptera_data/train/bees/1097045929_1753d1c765.jpg
inflating: hymenoptera_data/train/bees/1232245714_f862fbe385.jpg
inflating: hymenoptera_data/train/bees/129236073_0985e91c7d.jpg
inflating: hymenoptera_data/train/bees/1295655112_7813f37d21.jpg
inflating: hymenoptera_data/train/bees/132511197_0b86ad0fff.jpg
inflating: hymenoptera_data/train/bees/132826773_dbbcb117b9.jpg
inflating: hymenoptera_data/train/bees/150013791_969d9a968b.jpg
inflating: hymenoptera_data/train/bees/1508176360_2972117c9d.jpg
inflating: hymenoptera_data/train/bees/154600396_53e1252e52.jpg
inflating: hymenoptera_data/train/bees/16838648_415acd9e3f.jpg
inflating: hymenoptera_data/train/bees/1691282715_0addfdf5e8.jpg
inflating: hymenoptera_data/train/bees/17209602_fe5a5a746f.jpg
inflating: hymenoptera_data/train/bees/174142798_e5ad6d76e0.jpg
inflating: hymenoptera_data/train/bees/1799726602_8580867f71.jpg
inflating: hymenoptera_data/train/bees/1807583459_4fe92b3133.jpg
inflating: hymenoptera_data/train/bees/196430254_46bd129ae7.jpg
inflating: hymenoptera_data/train/bees/196658222_3fffd79c67.jpg
inflating: hymenoptera_data/train/bees/198508668_97d818b6c4.jpg
inflating: hymenoptera_data/train/bees/2031225713_50ed499635.jpg
inflating: hymenoptera_data/train/bees/2037437624_2d7bce461f.jpg
inflating: hymenoptera_data/train/bees/2053200300_8911ef438a.jpg
inflating: hymenoptera_data/train/bees/205835650_e6f2614bee.jpg
inflating: hymenoptera_data/train/bees/208702903_42fb4d9748.jpg
inflating: hymenoptera_data/train/bees/21399619_3e61e5bb6f.jpg
inflating: hymenoptera_data/train/bees/2227611847_ec72d40403.jpg
inflating: hymenoptera_data/train/bees/2321139806_d73d899e66.jpg
inflating: hymenoptera_data/train/bees/2330918208_8074770c20.jpg
inflating: hymenoptera_data/train/bees/2345177635_caf07159b3.jpg
inflating: hymenoptera_data/train/bees/2358061370_9daabbd9ac.jpg
inflating: hymenoptera_data/train/bees/2364597044_3c3e3fc391.jpg
inflating: hymenoptera_data/train/bees/2384149906_2cd8b0b699.jpg
inflating: hymenoptera_data/train/bees/2397446847_04ef3cd3e1.jpg
inflating: hymenoptera_data/train/bees/2405441001_b06c36fa72.jpg
inflating: hymenoptera_data/train/bees/2445215254_51698ff797.jpg
inflating: hymenoptera_data/train/bees/2452236943_255bfd9e58.jpg
inflating: hymenoptera_data/train/bees/2467959963_a7831e9ff0.jpg
inflating: hymenoptera_data/train/bees/2470492904_837e97800d.jpg
inflating: hymenoptera_data/train/bees/2477324698_3d4b1b1cab.jpg
inflating: hymenoptera_data/train/bees/2477349551_e75c97cf4d.jpg
inflating: hymenoptera_data/train/bees/2486729079_62df0920be.jpg
inflating: hymenoptera_data/train/bees/2486746709_c43cec0e42.jpg
inflating: hymenoptera_data/train/bees/2493379287_4100e1dacc.jpg
inflating: hymenoptera_data/train/bees/2495722465_879acf9d85.jpg
inflating: hymenoptera_data/train/bees/2528444139_fa728b0f5b.jpg
inflating: hymenoptera_data/train/bees/2538361678_9da84b77e3.jpg
inflating: hymenoptera_data/train/bees/2551813042_8a070aeb2b.jpg
inflating: hymenoptera_data/train/bees/2580598377_a4caecdb54.jpg
inflating: hymenoptera_data/train/bees/2601176055_8464e6aa71.jpg
```

```
inflating: hymenoptera_data/train/bees/2610833167_79bf0bcae5.jpg
inflating: hymenoptera_data/train/bees/2610838525_fe8e3cae47.jpg
inflating: hymenoptera_data/train/bees/2617161745_fa3ebe85b4.jpg
inflating: hymenoptera_data/train/bees/2625499656_e3415e374d.jpg
inflating: hymenoptera_data/train/bees/2634617358_f32fd16bea.jpg
inflating: hymenoptera_data/train/bees/2638074627_6b3ae746a0.jpg
inflating: hymenoptera_data/train/bees/2645107662_b73a8595cc.jpg
inflating: hymenoptera_data/train/bees/2651621464_a2fa8722eb.jpg
inflating: hymenoptera_data/train/bees/2652877533_a564830cbf.jpg
inflating: hymenoptera_data/train/bees/266644509_d30bb16a1b.jpg
inflating: hymenoptera_data/train/bees/2683605182_9d2a0c66cf.jpg
inflating: hymenoptera_data/train/bees/2704348794_eb5d5178c2.jpg
inflating: hymenoptera_data/train/bees/2707440199_cd170bd512.jpg
inflating: hymenoptera_data/train/bees/2710368626_cb42882dc8.jpg
inflating: hymenoptera_data/train/bees/2722592222_258d473e17.jpg
inflating: hymenoptera_data/train/bees/2728759455_ce9bb8cd7a.jpg
inflating: hymenoptera_data/train/bees/2756397428_1d82a08807.jpg
inflating: hymenoptera_data/train/bees/2765347790_da6cf6cb40.jpg
inflating: hymenoptera_data/train/bees/2781170484_5d61835d63.jpg
inflating: hymenoptera_data/train/bees/279113587_b4843db199.jpg
inflating: hymenoptera_data/train/bees/2792000093_e8ae0718cf.jpg
inflating: hymenoptera_data/train/bees/2801728106_833798c909.jpg
inflating: hymenoptera_data/train/bees/2822388965_f6dca2a275.jpg
inflating: hymenoptera_data/train/bees/2861002136_52c7c6f708.jpg
inflating: hymenoptera_data/train/bees/2908916142_a7ac8b57a8.jpg
inflating: hymenoptera_data/train/bees/29494643_e3410f0d37.jpg
inflating: hymenoptera_data/train/bees/2959730355_416a18c63c.jpg
inflating: hymenoptera_data/train/bees/2962405283_22718d9617.jpg
inflating: hymenoptera_data/train/bees/3006264892_30e9cced70.jpg
inflating: hymenoptera_data/train/bees/3030189811_01d095b793.jpg
inflating: hymenoptera_data/train/bees/3030772428_8578335616.jpg
inflating: hymenoptera_data/train/bees/3044402684_3853071a87.jpg
inflating: hymenoptera_data/train/bees/3074585407_9854eb3153.jpg
inflating: hymenoptera_data/train/bees/3079610310_ac2d0ae7bc.jpg
inflating: hymenoptera_data/train/bees/3090975720_71f12e6de4.jpg
inflating: hymenoptera_data/train/bees/3100226504_c0d4f1e3f1.jpg
inflating: hymenoptera_data/train/bees/342758693_c56b89b6b6.jpg
inflating: hymenoptera_data/train/bees/354167719_22dca13752.jpg
inflating: hymenoptera_data/train/bees/359928878_b3b418c728.jpg
inflating: hymenoptera_data/train/bees/365759866_b15700c59b.jpg
inflating: hymenoptera_data/train/bees/36900412_92b81831ad.jpg
inflating: hymenoptera_data/train/bees/39672681_1302d204d1.jpg
inflating: hymenoptera_data/train/bees/39747887_42df2855ee.jpg
inflating: hymenoptera_data/train/bees/421515404_e87569fd8b.jpg
inflating: hymenoptera_data/train/bees/444532809_9e931e2279.jpg
inflating: hymenoptera_data/train/bees/446296270_d9e8b93ecf.jpg
inflating: hymenoptera_data/train/bees/452462677_7be43af8ff.jpg
inflating: hymenoptera_data/train/bees/452462695_40a4e5b559.jpg
```

```
inflating: hymenoptera_data/train/bees/457457145_5f86eb7e9c.jpg
inflating: hymenoptera_data/train/bees/465133211_80e0c27f60.jpg
inflating: hymenoptera_data/train/bees/469333327_358ba8fe8a.jpg
inflating: hymenoptera_data/train/bees/472288710_2abee16fa0.jpg
inflating: hymenoptera_data/train/bees/473618094_8ffdcab215.jpg
inflating: hymenoptera_data/train/bees/476347960_52edd72b06.jpg
inflating: hymenoptera_data/train/bees/478701318_bbd5e557b8.jpg
inflating: hymenoptera_data/train/bees/507288830_f46e8d4cb2.jpg
inflating: hymenoptera_data/train/bees/509247772_2db2d01374.jpg
inflating: hymenoptera_data/train/bees/513545352_fd3e7c7c5d.jpg
inflating: hymenoptera_data/train/bees/522104315_5d3cb2758e.jpg
inflating: hymenoptera_data/train/bees/537309131_532bfa59ea.jpg
inflating: hymenoptera_data/train/bees/586041248_3032e277a9.jpg
inflating: hymenoptera_data/train/bees/760526046_547e8b381f.jpg
inflating: hymenoptera_data/train/bees/760568592_45a52c847f.jpg
inflating: hymenoptera_data/train/bees/774440991_63a4aa0cbe.jpg
inflating: hymenoptera_data/train/bees/85112639_6e860b0469.jpg
inflating: hymenoptera_data/train/bees/873076652_eb098dab2d.jpg
inflating: hymenoptera_data/train/bees/90179376_abc234e5f4.jpg
inflating: hymenoptera_data/train/bees/92663402_37f379e57a.jpg
inflating: hymenoptera_data/train/bees/95238259_98470c5b10.jpg
inflating: hymenoptera_data/train/bees/969455125_58c797ef17.jpg
inflating: hymenoptera_data/train/bees/98391118_bdb1e80cce.jpg
 creating: hymenoptera_data/val/
 creating: hymenoptera_data/val/ants/
inflating: hymenoptera_data/val/ants/10308379_1b6c72e180.jpg
inflating: hymenoptera_data/val/ants/1053149811_f62a3410d3.jpg
inflating: hymenoptera_data/val/ants/1073564163_225a64f170.jpg
inflating: hymenoptera_data/val/ants/1119630822_cd325ea21a.jpg
inflating: hymenoptera_data/val/ants/1124525276_816a07c17f.jpg
inflating: hymenoptera_data/val/ants/11381045_b352a47d8c.jpg
inflating: hymenoptera_data/val/ants/119785936_dd428e40c3.jpg
inflating: hymenoptera_data/val/ants/1247887232_edcb61246c.jpg
inflating: hymenoptera_data/val/ants/1262751255_c56c042b7b.jpg
inflating: hymenoptera_data/val/ants/1337725712_2eb53cd742.jpg
inflating: hymenoptera_data/val/ants/1358854066_5ad8015f7f.jpg
inflating: hymenoptera_data/val/ants/1440002809_b268d9a66a.jpg
inflating: hymenoptera_data/val/ants/147542264_79506478c2.jpg
inflating: hymenoptera_data/val/ants/152286280_411648ec27.jpg
inflating: hymenoptera_data/val/ants/153320619_2aeb5fa0ee.jpg
inflating: hymenoptera_data/val/ants/153783656_85f9c3ac70.jpg
inflating: hymenoptera_data/val/ants/157401988_d0564a9d02.jpg
inflating: hymenoptera_data/val/ants/159515240_d5981e20d1.jpg
inflating: hymenoptera_data/val/ants/161076144_124db762d6.jpg
inflating: hymenoptera_data/val/ants/161292361_c16e0bf57a.jpg
inflating: hymenoptera_data/val/ants/170652283_ecdaff5d1a.jpg
inflating: hymenoptera_data/val/ants/17081114_79b9a27724.jpg
inflating: hymenoptera_data/val/ants/172772109_d0a8e15fb0.jpg
```

```
  inflating: hymenoptera_data/val/ants/1743840368_b5ccda82b7.jpg
  inflating: hymenoptera_data/val/ants/181942028_961261ef48.jpg
  inflating: hymenoptera_data/val/ants/183260961_64ab754c97.jpg
  inflating: hymenoptera_data/val/ants/2039585088_c6f47c592e.jpg
  inflating: hymenoptera_data/val/ants/205398178_c395c5e460.jpg
  inflating: hymenoptera_data/val/ants/208072188_f293096296.jpg
  inflating: hymenoptera_data/val/ants/209615353_eeb38ba204.jpg
  inflating: hymenoptera_data/val/ants/2104709400_8831b4fc6f.jpg
  inflating: hymenoptera_data/val/ants/212100470_b485e7b7b9.jpg
  inflating: hymenoptera_data/val/ants/2127908701_d49dc83c97.jpg
  inflating: hymenoptera_data/val/ants/2191997003_379df31291.jpg
  inflating: hymenoptera_data/val/ants/2211974567_ee4606b493.jpg
  inflating: hymenoptera_data/val/ants/2219621907_47bc7cc6b0.jpg
  inflating: hymenoptera_data/val/ants/2238242353_52c82441df.jpg
  inflating: hymenoptera_data/val/ants/2255445811_dabcdf7258.jpg
  inflating: hymenoptera_data/val/ants/239161491_86ac23b0a3.jpg
  inflating: hymenoptera_data/val/ants/263615709_cfb28f6b8e.jpg
  inflating: hymenoptera_data/val/ants/308196310_1db5ffa01b.jpg
  inflating: hymenoptera_data/val/ants/319494379_648fb5a1c6.jpg
  inflating: hymenoptera_data/val/ants/35558229_1fa4608a7a.jpg
  inflating: hymenoptera_data/val/ants/412436937_4c2378efc2.jpg
  inflating: hymenoptera_data/val/ants/436944325_d4925a38c7.jpg
  inflating: hymenoptera_data/val/ants/445356866_6cb3289067.jpg
  inflating: hymenoptera_data/val/ants/459442412_412fecf3fe.jpg
  inflating: hymenoptera_data/val/ants/470127071_8b8ee2bd74.jpg
  inflating: hymenoptera_data/val/ants/477437164_bc3e6e594a.jpg
  inflating: hymenoptera_data/val/ants/488272201_c5aa281348.jpg
  inflating: hymenoptera_data/val/ants/502717153_3e4865621a.jpg
  inflating: hymenoptera_data/val/ants/518746016_bcc28f8b5b.jpg
  inflating: hymenoptera_data/val/ants/540543309_ddbb193ee5.jpg
  inflating: hymenoptera_data/val/ants/562589509_7e55469b97.jpg
  inflating: hymenoptera_data/val/ants/57264437_a19006872f.jpg
  inflating: hymenoptera_data/val/ants/573151833_ebbc274b77.jpg
  inflating: hymenoptera_data/val/ants/649407494_9b6bc4949f.jpg
  inflating: hymenoptera_data/val/ants/751649788_78dd7d16ce.jpg
  inflating: hymenoptera_data/val/ants/768870506_8f115d3d37.jpg
  inflating: hymenoptera_data/val/ants/800px-
Meat_eater_ant_qeen_excavating_hole.jpg
  inflating: hymenoptera_data/val/ants/8124241_36b290d372.jpg
  inflating: hymenoptera_data/val/ants/8398478_50ef10c47a.jpg
  inflating: hymenoptera_data/val/ants/854534770_31f6156383.jpg
  inflating: hymenoptera_data/val/ants/892676922_4ab37dce07.jpg
  inflating: hymenoptera_data/val/ants/94999827_36895faade.jpg
  inflating: hymenoptera_data/val/ants/Ant-1818.jpg
  inflating: hymenoptera_data/val/ants/ants-devouring-remains-of-large-dead-
insect-on-red-tile-in-Stellenbosch-South-Africa-closeup-1-DHD.jpg
  inflating: hymenoptera_data/val/ants/desert_ant.jpg
  inflating: hymenoptera_data/val/ants/F.pergan.28(f).jpg
```

```
inflating: hymenoptera_data/val/ants/Hormiga.jpg
 creating: hymenoptera_data/val/bees/
inflating: hymenoptera_data/val/bees/1032546534_06907fe3b3.jpg
inflating: hymenoptera_data/val/bees/10870992_eebeeb3a12.jpg
inflating: hymenoptera_data/val/bees/1181173278_23c36fac71.jpg
inflating: hymenoptera_data/val/bees/1297972485_33266a18d9.jpg
inflating: hymenoptera_data/val/bees/1328423762_f7a88a8451.jpg
inflating: hymenoptera_data/val/bees/1355974687_1341c1face.jpg
inflating: hymenoptera_data/val/bees/144098310_a4176fd54d.jpg
inflating: hymenoptera_data/val/bees/1486120850_490388f84b.jpg
inflating: hymenoptera_data/val/bees/149973093_da3c446268.jpg
inflating: hymenoptera_data/val/bees/151594775_ee7dc17b60.jpg
inflating: hymenoptera_data/val/bees/151603988_2c6f7d14c7.jpg
inflating: hymenoptera_data/val/bees/1519368889_4270261ee3.jpg
inflating: hymenoptera_data/val/bees/152789693_220b003452.jpg
inflating: hymenoptera_data/val/bees/177677657_a38c97e572.jpg
inflating: hymenoptera_data/val/bees/1799729694_0c40101071.jpg
inflating: hymenoptera_data/val/bees/181171681_c5a1a82ded.jpg
inflating: hymenoptera_data/val/bees/187130242_4593a4c610.jpg
inflating: hymenoptera_data/val/bees/203868383_0fcbb48278.jpg
inflating: hymenoptera_data/val/bees/2060668999_e11edb10d0.jpg
inflating: hymenoptera_data/val/bees/2086294791_6f3789d8a6.jpg
inflating: hymenoptera_data/val/bees/2103637821_8d26ee6b90.jpg
inflating: hymenoptera_data/val/bees/2104135106_a65eede1de.jpg
inflating: hymenoptera_data/val/bees/215512424_687e1e0821.jpg
inflating: hymenoptera_data/val/bees/2173503984_9c6aaaa7e2.jpg
inflating: hymenoptera_data/val/bees/220376539_20567395d8.jpg
inflating: hymenoptera_data/val/bees/224841383_d050f5f510.jpg
inflating: hymenoptera_data/val/bees/2321144482_f3785ba7b2.jpg
inflating: hymenoptera_data/val/bees/238161922_55fa9a76ae.jpg
inflating: hymenoptera_data/val/bees/2407809945_fb525ef54d.jpg
inflating: hymenoptera_data/val/bees/2415414155_1916f03b42.jpg
inflating: hymenoptera_data/val/bees/2438480600_40a1249879.jpg
inflating: hymenoptera_data/val/bees/2444778727_4b781ac424.jpg
inflating: hymenoptera_data/val/bees/2457841282_7867f16639.jpg
inflating: hymenoptera_data/val/bees/2470492902_3572c90f75.jpg
inflating: hymenoptera_data/val/bees/2478216347_535c8fe6d7.jpg
inflating: hymenoptera_data/val/bees/2501530886_e20952b97d.jpg
inflating: hymenoptera_data/val/bees/2506114833_90a41c5267.jpg
inflating: hymenoptera_data/val/bees/2509402554_31821cb0b6.jpg
inflating: hymenoptera_data/val/bees/2525379273_dcb26a516d.jpg
inflating: hymenoptera_data/val/bees/26589803_5ba7000313.jpg
inflating: hymenoptera_data/val/bees/2668391343_45e272cd07.jpg
inflating: hymenoptera_data/val/bees/2670536155_c170f49cd0.jpg
inflating: hymenoptera_data/val/bees/2685605303_9eed79d59d.jpg
inflating: hymenoptera_data/val/bees/2702408468_d9ed795f4f.jpg
inflating: hymenoptera_data/val/bees/2709775832_85b4b50a57.jpg
inflating: hymenoptera_data/val/bees/2717418782_bd83307d9f.jpg
```

```
  inflating: hymenoptera_data/val/bees/272986700_d4d4bf8c4b.jpg
  inflating: hymenoptera_data/val/bees/2741763055_9a7bb00802.jpg
  inflating: hymenoptera_data/val/bees/2745389517_250a397f31.jpg
  inflating: hymenoptera_data/val/bees/2751836205_6f7b5eff30.jpg
  inflating: hymenoptera_data/val/bees/2782079948_8d4e94a826.jpg
  inflating: hymenoptera_data/val/bees/2809496124_5f25b5946a.jpg
  inflating: hymenoptera_data/val/bees/2815838190_0a9889d995.jpg
  inflating: hymenoptera_data/val/bees/2841437312_789699c740.jpg
  inflating: hymenoptera_data/val/bees/2883093452_7e3a1eb53f.jpg
  inflating: hymenoptera_data/val/bees/290082189_f66cb80bfc.jpg
  inflating: hymenoptera_data/val/bees/296565463_d07a7bed96.jpg
  inflating: hymenoptera_data/val/bees/3077452620_548c79fda0.jpg
  inflating: hymenoptera_data/val/bees/348291597_ee836fbb1a.jpg
  inflating: hymenoptera_data/val/bees/350436573_41f4ecb6c8.jpg
  inflating: hymenoptera_data/val/bees/353266603_d3eac7e9a0.jpg
  inflating: hymenoptera_data/val/bees/372228424_16da1f8884.jpg
  inflating: hymenoptera_data/val/bees/400262091_701c00031c.jpg
  inflating: hymenoptera_data/val/bees/416144384_961c326481.jpg
  inflating: hymenoptera_data/val/bees/44105569_16720a960c.jpg
  inflating: hymenoptera_data/val/bees/456097971_860949c4fc.jpg
  inflating: hymenoptera_data/val/bees/464594019_1b24a28bb1.jpg
  inflating: hymenoptera_data/val/bees/485743562_d8cc6b8f73.jpg
  inflating: hymenoptera_data/val/bees/540976476_844950623f.jpg
  inflating: hymenoptera_data/val/bees/54736755_c057723f64.jpg
  inflating: hymenoptera_data/val/bees/57459255_752774f1b2.jpg
  inflating: hymenoptera_data/val/bees/576452297_897023f002.jpg
  inflating: hymenoptera_data/val/bees/586474709_ae436da045.jpg
  inflating: hymenoptera_data/val/bees/590318879_68cf112861.jpg
  inflating: hymenoptera_data/val/bees/59798110_2b6a3c8031.jpg
  inflating: hymenoptera_data/val/bees/603709866_a97c7cfc72.jpg
  inflating: hymenoptera_data/val/bees/603711658_4c8cd2201e.jpg
  inflating: hymenoptera_data/val/bees/65038344_52a45d090d.jpg
  inflating:
hymenoptera_data/val/bees/6a00d8341c630a53ef00e553d0beb18834-800wi.jpg
  inflating: hymenoptera_data/val/bees/72100438_73de9f17af.jpg
  inflating: hymenoptera_data/val/bees/759745145_e8bc776ec8.jpg
  inflating: hymenoptera_data/val/bees/936182217_c4caa5222d.jpg
  inflating: hymenoptera_data/val/bees/abeja.jpg
```

[2]:
```
!unzip /content/models.zip
!unzip /content/utils.zip
```

```
Archive:  /content/models.zip
   creating: models/__pycache__/
  inflating: models/__pycache__/configs.cpython-312.pyc
  inflating: models/__pycache__/modeling.cpython-312.pyc
  inflating: models/__pycache__/modeling_resnet.cpython-312.pyc
  inflating: models/configs.py
```

```
  inflating: models/modeling.py
  inflating: models/modeling_resnet.py
  inflating: utils/data_utils.py
  inflating: utils/dist_util.py
  inflating: utils/scheduler.py
Archive:  /content/utils.zip
replace utils/data_utils.py? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: utils/data_utils.py
replace utils/dist_util.py? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: utils/dist_util.py
replace utils/scheduler.py? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: utils/scheduler.py
```

[3]:
```
# prompt: install the the requirements libs /content/requirements.txt

!pip install -r /content/requirements.txt
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages
(from -r /content/requirements.txt (line 1)) (2.5.1+cu124)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from -r /content/requirements.txt (line 2)) (1.26.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from -r /content/requirements.txt (line 3)) (4.67.1)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.11/dist-
packages (from -r /content/requirements.txt (line 4)) (2.18.0)
Collecting ml-collections (from -r /content/requirements.txt (line 5))
  Downloading ml_collections-1.0.0-py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from torch->-r /content/requirements.txt (line 1)) (3.17.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.11/dist-packages (from torch->-r
/content/requirements.txt (line 1)) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-
packages (from torch->-r /content/requirements.txt (line 1)) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages
(from torch->-r /content/requirements.txt (line 1)) (3.1.5)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages
(from torch->-r /content/requirements.txt (line 1)) (2024.10.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch->-r
/content/requirements.txt (line 1))
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch->-r
/content/requirements.txt (line 1))
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch->-r
/content/requirements.txt (line 1))
```

```
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch->-r /content/requirements.txt
(line 1))
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch->-r
/content/requirements.txt (line 1))
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch->-r /content/requirements.txt
(line 1))
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch->-r
/content/requirements.txt (line 1))
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch->-r
/content/requirements.txt (line 1))
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch->-r
/content/requirements.txt (line 1))
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch->-r
/content/requirements.txt (line 1)) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch->-r
/content/requirements.txt (line 1)) (12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch->-r
/content/requirements.txt (line 1))
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/dist-
packages (from torch->-r /content/requirements.txt (line 1)) (3.1.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-
packages (from torch->-r /content/requirements.txt (line 1)) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch->-r
/content/requirements.txt (line 1)) (1.3.0)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.11/dist-
packages (from tensorboard->-r /content/requirements.txt (line 4)) (1.4.0)
Requirement already satisfied: grpcio>=1.48.2 in /usr/local/lib/python3.11/dist-
packages (from tensorboard->-r /content/requirements.txt (line 4)) (1.70.0)
Requirement already satisfied: markdown>=2.6.8 in
```

```
/usr/local/lib/python3.11/dist-packages (from tensorboard->-r
/content/requirements.txt (line 4)) (3.7)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from tensorboard->-r /content/requirements.txt (line 4)) (24.2)
Requirement already satisfied: protobuf!=4.24.0,>=3.19.6 in
/usr/local/lib/python3.11/dist-packages (from tensorboard->-r
/content/requirements.txt (line 4)) (4.25.6)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.11/dist-packages (from tensorboard->-r
/content/requirements.txt (line 4)) (75.1.0)
Requirement already satisfied: six>1.9 in /usr/local/lib/python3.11/dist-
packages (from tensorboard->-r /content/requirements.txt (line 4)) (1.17.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.11/dist-packages (from tensorboard->-r
/content/requirements.txt (line 4)) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from tensorboard->-r
/content/requirements.txt (line 4)) (3.1.3)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.11/dist-packages
(from ml-collections->-r /content/requirements.txt (line 5)) (6.0.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard->-r
/content/requirements.txt (line 4)) (3.0.2)
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4
MB)
                         363.4/363.4 MB
3.8 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (13.8 MB)
                         13.8/13.8 MB
110.2 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (24.6 MB)
                         24.6/24.6 MB
84.0 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (883 kB)
                         883.7/883.7 kB
62.4 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl
(664.8 MB)
                         664.8/664.8 MB
2.2 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl
(211.5 MB)
                         211.5/211.5 MB
5.7 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
```

```
manylinux2014_x86_64.whl (56.3 MB)
                         56.3/56.3 MB
12.3 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl (127.9 MB)
                         127.9/127.9 MB
7.0 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl (207.5 MB)
                         207.5/207.5 MB
7.2 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (21.1 MB)
                         21.1/21.1 MB
94.2 MB/s eta 0:00:00
Downloading ml_collections-1.0.0-py3-none-any.whl (76 kB)
                         76.5/76.5 kB
8.4 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12,
nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-
cuda-cupti-cu12, nvidia-cublas-cu12, ml-collections, nvidia-cusparse-cu12,
nvidia-cudnn-cu12, nvidia-cusolver-cu12
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
```

```
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed ml-collections-1.0.0 nvidia-cublas-cu12-12.4.5.8 nvidia-
cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-
cu12-12.4.127 nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3 nvidia-
curand-cu12-10.3.5.147 nvidia-cusolver-cu12-11.6.1.9 nvidia-cusparse-
cu12-12.3.1.170 nvidia-nvjitlink-cu12-12.4.127
```

1 Vision Transformer for Binary Classification

2 This notebook demonstrates the use of various Vision Transformer (ViT) configurations for binary classification of images (bees vs. ants).

3 We will train and evaluate models using different configurations and compare their performance.

[4]:
```
# Import necessary libraries
import torch
import torch.nn as nn
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
from models.modeling import VisionTransformer
import os
import models.configs as configs
```

[5]:
```
import random
import matplotlib.pyplot as plt
import numpy as np
```

[6]:
```
import ml_collections
```

4 ## Data Preparation

5 We will load the dataset and apply necessary transformations to prepare it for training and evaluation.

[7]:
```
# Define data transformations
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(degrees=15), # Add rotation
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2,
 ↪hue=0.1), # Add color jitter
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

6 Load the dataset

[11]:
```
# Load the dataset
data_dir = 'hymenoptera_data'
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
                                          data_transforms[x])
                  for x in ['train', 'val']}
dataloaders = {x: DataLoader(image_datasets[x], batch_size=32,
                             shuffle=True, num_workers=4)
              for x in ['train', 'val']}
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val']}
class_names = image_datasets['train'].classes
```

```
/usr/local/lib/python3.11/dist-packages/torch/utils/data/dataloader.py:617:
UserWarning: This DataLoader will create 4 worker processes in total. Our
suggested max number of worker in current system is 2, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  warnings.warn(
```

[ ]:
```
##ViT-B/16 configuration
```

7 ## Model Training and Evaluation

8 We will train and evaluate models using different ViT (Vision Transformers) configurations and compare their performance.

8.1 Configuration: ViT-B/16

- **Purpose**: Represents the base Vision Transformer model.
- **Characteristics**:
  - Patch size: 16x16
  - Hidden size: 768
  - Attention heads: 12
  - Transformer layers: 12
- **Use Case**: Standard configuration for moderate complexity tasks.

[12]:
```
def get_b16_config():
    """
    Returns the ViT-B/16 configuration.

    This configuration represents the base Vision Transformer model with
    a patch size of 16x16, a hidden size of 768, 12 attention heads, and
    12 transformer layers. It is a standard configuration for moderate
    complexity tasks.

    Returns:
        ConfigDict: A configuration dictionary for ViT-B/16.
    """
    config = ml_collections.ConfigDict()
    config.patches = ml_collections.ConfigDict({'size': (16, 16)})
    config.hidden_size = 768
    config.transformer = ml_collections.ConfigDict()
    config.transformer.mlp_dim = 3072
    config.transformer.num_heads = 12
    config.transformer.num_layers = 12
    config.transformer.attention_dropout_rate = 0.0
    config.transformer.dropout_rate = 0.1
    config.classifier = 'token'
    config.representation_size = None
    return config
```

[13]:
```
config =  get_b16_config()
```

Initialize the ViT model for binary classification

[14]:
```
# Modify the ViT model for binary classification
model = VisionTransformer(config=config, num_classes=2)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
```

[15]:

```
# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

[16]:

```
# Training loop
num_epochs = 20
for epoch in range(num_epochs):
    print(f'Epoch {epoch+1}/{num_epochs}')
    print('-' * 10)
    for phase in ['train', 'val']:
        if phase == 'train':
            model.train()
        else:
            model.eval()
        running_loss = 0.0
        running_corrects = 0
        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            with torch.set_grad_enabled(phase == 'train'):
                # Assuming the first element of the tuple is the output we need
                outputs = model(inputs)[0] # Access the first element of the␣
 ↪tuple
                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)
                if phase == 'train':
                    loss.backward()
                    optimizer.step()
            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)
        epoch_loss = running_loss / dataset_sizes[phase]
        epoch_acc = running_corrects.double() / dataset_sizes[phase]
        print(f'{phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')
```

```
Epoch 1/20
----------
train Loss: 2.8023 Acc: 0.5451
val Loss: 0.9151 Acc: 0.4575
Epoch 2/20
----------
train Loss: 0.8231 Acc: 0.5410
val Loss: 0.9932 Acc: 0.4575
Epoch 3/20
----------
train Loss: 0.8276 Acc: 0.5123
val Loss: 0.7546 Acc: 0.5425
```

```
Epoch 4/20
----------
train Loss: 0.7902 Acc: 0.4467
val Loss: 0.7137 Acc: 0.4575
Epoch 5/20
----------
train Loss: 0.7408 Acc: 0.4795
val Loss: 0.7147 Acc: 0.4575
Epoch 6/20
----------
train Loss: 0.7164 Acc: 0.5041
val Loss: 0.6909 Acc: 0.5425
Epoch 7/20
----------
train Loss: 0.7001 Acc: 0.5123
val Loss: 0.7262 Acc: 0.4575
Epoch 8/20
----------
train Loss: 0.7157 Acc: 0.4877
val Loss: 0.7001 Acc: 0.5425
Epoch 9/20
----------
train Loss: 0.6953 Acc: 0.5369
val Loss: 0.7362 Acc: 0.4575
Epoch 10/20
----------
train Loss: 0.7084 Acc: 0.4836
val Loss: 0.6898 Acc: 0.5686
Epoch 11/20
----------
train Loss: 0.6918 Acc: 0.4754
val Loss: 0.6920 Acc: 0.5359
Epoch 12/20
----------
train Loss: 0.7042 Acc: 0.4549
val Loss: 0.6954 Acc: 0.4510
Epoch 13/20
----------
train Loss: 0.6987 Acc: 0.4631
val Loss: 0.6873 Acc: 0.5556
Epoch 14/20
----------
train Loss: 0.7047 Acc: 0.5041
val Loss: 0.7067 Acc: 0.4575
Epoch 15/20
----------
train Loss: 0.7012 Acc: 0.4836
val Loss: 0.7116 Acc: 0.4575
```

```
Epoch 16/20
----------
train Loss: 0.7110 Acc: 0.4590
val Loss: 0.7225 Acc: 0.4575
Epoch 17/20
----------
train Loss: 0.7074 Acc: 0.5041
val Loss: 0.7239 Acc: 0.4575
Epoch 18/20
----------
train Loss: 0.7055 Acc: 0.5246
val Loss: 0.6913 Acc: 0.5490
Epoch 19/20
----------
train Loss: 0.6961 Acc: 0.4877
val Loss: 0.6639 Acc: 0.5948
Epoch 20/20
----------
train Loss: 0.6758 Acc: 0.5697
val Loss: 0.6468 Acc: 0.6471
```

[17]:
```
# Evaluate the model
model.eval()
running_corrects = 0
for inputs, labels in dataloaders['val']:
    inputs = inputs.to(device)
    labels = labels.to(device)
    # Assuming the first element of the tuple is the output we need
    outputs = model(inputs)[0]  # Access the first element of the tuple
    _, preds = torch.max(outputs, 1)
    running_corrects += torch.sum(preds == labels.data)
accuracy = running_corrects.double() / dataset_sizes['val']
print(f'Validation Accuracy: {accuracy:.4f}')
```

```
Validation Accuracy: 0.6471
```

[ ]:
```
# Test on 5 random images
```

[18]:
```
model_name = "ViT-B/16 configuration model"
```

[19]:
```
# Function to display images with predictions
def imshow(inp, title=None):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
```

```
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  #
```

Testing on Random Images #We will test the model on 5 random images from the validation set and display the results.

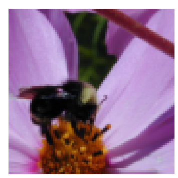
[21]:
```
print(f"\nTesting 5 random images for {model_name}:")
model.eval()
images_so_far = 0
fig = plt.figure(figsize=(15, 10))
with torch.no_grad():
    for i, (inputs, labels) in enumerate(dataloaders['val']):
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        # Access the first element of the tuple, which is the output we need
        outputs = outputs[0]
        _, preds = torch.max(outputs, 1)

        for j in range(inputs.size()[0]):
            images_so_far += 1
            ax = plt.subplot(1, 5, images_so_far)
            ax.axis('off')
            ax.set_title(f'True: {class_names[labels[j]]}\nPred:␣
 ↪{class_names[preds[j]]}')
            imshow(inputs.cpu().data[j])

            if images_so_far == 5:
                break
        if images_so_far == 5:
            break
plt.show()
```

```
Testing 5 random images for ViT-B/16 configuration model:
```

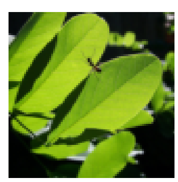
```
/usr/local/lib/python3.11/dist-packages/torch/utils/data/dataloader.py:617:
UserWarning: This DataLoader will create 4 worker processes in total. Our
suggested max number of worker in current system is 2, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  warnings.warn(
```

True: ants
Pred: bees

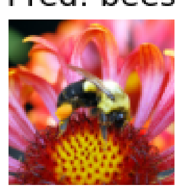
True: bees
Pred: bees

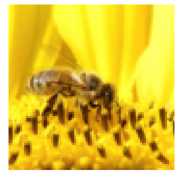


True: ants
Pred: bees



True: bees
Pred: bees






```
[ ]: ## simple input for testing purpose ###
```

8.2 Configuration: Testing

- **Purpose**: Designed for quick tests and debugging.
- **Characteristics**: Minimal resources with a very small hidden size, MLP dimension, and number of heads and layers.
- **Use Case**: Suitable for verifying code functionality without heavy computation.

```
[22]: def get_testing():
          """
          Returns a minimal configuration for testing.

          This configuration is designed for quick tests and debugging.
          It uses minimal resources with a very small hidden size, MLP dimension,
          and number of heads and layers.

          Returns:
              ConfigDict: A configuration dictionary with minimal settings.
          """
          config = ml_collections.ConfigDict()
          config.patches = ml_collections.ConfigDict({'size': (16, 16)})
          config.hidden_size = 1
          config.transformer = ml_collections.ConfigDict()
          config.transformer.mlp_dim = 1
          config.transformer.num_heads = 1
          config.transformer.num_layers = 1
          config.transformer.attention_dropout_rate = 0.0
          config.transformer.dropout_rate = 0.1
          config.classifier = 'token'
          config.representation_size = None
          return config
```

```
[23]: config =  get_testing()
```

[24]:
```
# Modify the ViT model for binary classification
model = VisionTransformer(config=config, num_classes=2)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
```

[25]:
```
# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

[26]:
```
# Training loop
num_epochs = 20
for epoch in range(num_epochs):
    print(f'Epoch {epoch+1}/{num_epochs}')
    print('-' * 10)
    for phase in ['train', 'val']:
        if phase == 'train':
            model.train()
        else:
            model.eval()
        running_loss = 0.0
        running_corrects = 0
        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            with torch.set_grad_enabled(phase == 'train'):
                # Assuming the first element of the tuple is the output we need
                outputs = model(inputs)[0] # Access the first element of the␣
↪tuple

                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)
                if phase == 'train':
                    loss.backward()
                    optimizer.step()
            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)
        epoch_loss = running_loss / dataset_sizes[phase]
        epoch_acc = running_corrects.double() / dataset_sizes[phase]
        print(f'{phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')
```

```
Epoch 1/20
----------
train Loss: 0.6988 Acc: 0.5041
val Loss: 0.7087 Acc: 0.4575
Epoch 2/20
----------
train Loss: 0.6981 Acc: 0.5041
```

```
val Loss: 0.7075 Acc: 0.4575
Epoch 3/20
----------
train Loss: 0.6976 Acc: 0.5041
val Loss: 0.7063 Acc: 0.4575
Epoch 4/20
----------
train Loss: 0.6969 Acc: 0.5041
val Loss: 0.7053 Acc: 0.4575
Epoch 5/20
----------
train Loss: 0.6964 Acc: 0.5041
val Loss: 0.7044 Acc: 0.4575
Epoch 6/20
----------
train Loss: 0.6961 Acc: 0.5041
val Loss: 0.7034 Acc: 0.4575
Epoch 7/20
----------
train Loss: 0.6958 Acc: 0.5041
val Loss: 0.7025 Acc: 0.4575
Epoch 8/20
----------
train Loss: 0.6953 Acc: 0.5041
val Loss: 0.7019 Acc: 0.4575
Epoch 9/20
----------
train Loss: 0.6950 Acc: 0.5041
val Loss: 0.7012 Acc: 0.4575
Epoch 10/20
----------
train Loss: 0.6948 Acc: 0.5041
val Loss: 0.7005 Acc: 0.4575
Epoch 11/20
----------
train Loss: 0.6946 Acc: 0.5041
val Loss: 0.6999 Acc: 0.4575
Epoch 12/20
----------
train Loss: 0.6943 Acc: 0.5041
val Loss: 0.6994 Acc: 0.4575
Epoch 13/20
----------
train Loss: 0.6942 Acc: 0.5041
val Loss: 0.6988 Acc: 0.4575
Epoch 14/20
----------
train Loss: 0.6940 Acc: 0.5041
```

```
val Loss: 0.6984 Acc: 0.4575
Epoch 15/20
----------
train Loss: 0.6939 Acc: 0.5041
val Loss: 0.6980 Acc: 0.4575
Epoch 16/20
----------
train Loss: 0.6939 Acc: 0.5041
val Loss: 0.6975 Acc: 0.4575
Epoch 17/20
----------
train Loss: 0.6937 Acc: 0.5041
val Loss: 0.6973 Acc: 0.4575
Epoch 18/20
----------
train Loss: 0.6936 Acc: 0.5041
val Loss: 0.6971 Acc: 0.4575
Epoch 19/20
----------
train Loss: 0.6936 Acc: 0.5041
val Loss: 0.6970 Acc: 0.4575
Epoch 20/20
----------
train Loss: 0.6936 Acc: 0.5041
val Loss: 0.6966 Acc: 0.4575
```

[27]:
```
# Evaluate the model
model.eval()
running_corrects = 0
for inputs, labels in dataloaders['val']:
    inputs = inputs.to(device)
    labels = labels.to(device)
    # Assuming the first element of the tuple is the output we need
    outputs = model(inputs)[0]  # Access the first element of the tuple
    _, preds = torch.max(outputs, 1)
    running_corrects += torch.sum(preds == labels.data)
accuracy = running_corrects.double() / dataset_sizes['val']
print(f'Validation Accuracy: {accuracy:.4f}')
```

```
Validation Accuracy: 0.4575
```

[28]:
```
model_name = "minimum-configuration model"
```

[29]:
```
# Function to display images with predictions
def imshow(inp, title=None):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
```

```
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  #
```

Testing on Random Images #We will test the model on 5 random images from the validation set and display the results.

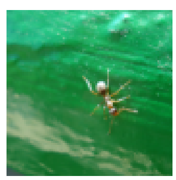
[30]:
```
print(f"\nTesting 5 random images for {model_name}:")
model.eval()
images_so_far = 0
fig = plt.figure(figsize=(15, 10))
with torch.no_grad():
    for i, (inputs, labels) in enumerate(dataloaders['val']):
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        # Access the first element of the tuple, which is the output we need
        outputs = outputs[0]
        _, preds = torch.max(outputs, 1)

        for j in range(inputs.size()[0]):
            images_so_far += 1
            ax = plt.subplot(1, 5, images_so_far)
            ax.axis('off')
            ax.set_title(f'True: {class_names[labels[j]]}\nPred:␣
 ↪{class_names[preds[j]]}')
            imshow(inputs.cpu().data[j])

            if images_so_far == 5:
                break
        if images_so_far == 5:
            break
plt.show()
```

```
Testing 5 random images for minimum-configuration model:
```

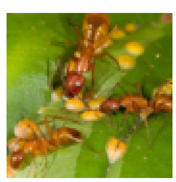
True: ants
Pred: ants

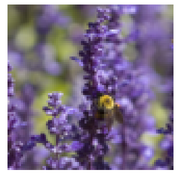
True: bees
Pred: ants

True: ants
Pred: ants



True: ants
Pred: ants






```
[ ]: ##Tetsing on ViT-B/32 configuration
```

8.3 Configuration: ViT-B/32

- **Purpose**: Similar to ViT-B/16 but with a larger patch size.
- **Characteristics**:
  - Patch size: 32x32
- **Use Case**: Reduces the number of patches and computational complexity.

```
[31]: def get_b32_config():
          """
          Returns the ViT-B/32 configuration.

          This configuration is similar to ViT-B/16 but uses a larger patch
          size of 32x32, which reduces the number of patches and computational
          complexity.

          Returns:
              ConfigDict: A configuration dictionary for ViT-B/32.
          """
          config = get_b16_config()
          config.patches.size = (32, 32)
          return config
```

```
[ ]: config =  get_b32_config()
```

```
[32]: # Modify the ViT model for binary classification
      model = VisionTransformer(config=config, num_classes=2)
      device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
      model = model.to(device)
```

```
[33]: # Define loss function and optimizer
      criterion = nn.CrossEntropyLoss()
      optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

[34]:

```python
# Training loop
num_epochs = 20
for epoch in range(num_epochs):
    print(f'Epoch {epoch+1}/{num_epochs}')
    print('-' * 10)
    for phase in ['train', 'val']:
        if phase == 'train':
            model.train()
        else:
            model.eval()
        running_loss = 0.0
        running_corrects = 0
        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            with torch.set_grad_enabled(phase == 'train'):
                # Assuming the first element of the tuple is the output we need
                outputs = model(inputs)[0] # Access the first element of the␣
↪tuple
                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)
                if phase == 'train':
                    loss.backward()
                    optimizer.step()
            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)
        epoch_loss = running_loss / dataset_sizes[phase]
        epoch_acc = running_corrects.double() / dataset_sizes[phase]
        print(f'{phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')
```

```
Epoch 1/20
----------
train Loss: 0.7028 Acc: 0.4959
val Loss: 0.6904 Acc: 0.5425
Epoch 2/20
----------
train Loss: 0.7019 Acc: 0.4959
val Loss: 0.6901 Acc: 0.5425
Epoch 3/20
----------
train Loss: 0.7010 Acc: 0.4959
val Loss: 0.6899 Acc: 0.5425
Epoch 4/20
----------
train Loss: 0.7002 Acc: 0.4959
val Loss: 0.6898 Acc: 0.5425
```

```
Epoch 5/20
----------
train Loss: 0.6994 Acc: 0.4959
val Loss: 0.6897 Acc: 0.5425
Epoch 6/20
----------
train Loss: 0.6989 Acc: 0.4959
val Loss: 0.6896 Acc: 0.5425
Epoch 7/20
----------
train Loss: 0.6983 Acc: 0.4959
val Loss: 0.6895 Acc: 0.5425
Epoch 8/20
----------
train Loss: 0.6976 Acc: 0.4959
val Loss: 0.6895 Acc: 0.5425
Epoch 9/20
----------
train Loss: 0.6971 Acc: 0.4959
val Loss: 0.6895 Acc: 0.5425
Epoch 10/20
----------
train Loss: 0.6968 Acc: 0.4959
val Loss: 0.6896 Acc: 0.5425
Epoch 11/20
----------
train Loss: 0.6963 Acc: 0.4959
val Loss: 0.6897 Acc: 0.5425
Epoch 12/20
----------
train Loss: 0.6960 Acc: 0.4959
val Loss: 0.6898 Acc: 0.5425
Epoch 13/20
----------
train Loss: 0.6957 Acc: 0.4959
val Loss: 0.6899 Acc: 0.5425
Epoch 14/20
----------
train Loss: 0.6953 Acc: 0.4959
val Loss: 0.6900 Acc: 0.5425
Epoch 15/20
----------
train Loss: 0.6951 Acc: 0.4959
val Loss: 0.6901 Acc: 0.5425
Epoch 16/20
----------
train Loss: 0.6948 Acc: 0.4959
val Loss: 0.6902 Acc: 0.5425
```

```
Epoch 17/20
----------
train Loss: 0.6947 Acc: 0.4959
val Loss: 0.6903 Acc: 0.5425
Epoch 18/20
----------
train Loss: 0.6944 Acc: 0.4959
val Loss: 0.6905 Acc: 0.5425
Epoch 19/20
----------
train Loss: 0.6945 Acc: 0.4959
val Loss: 0.6907 Acc: 0.5425
Epoch 20/20
----------
train Loss: 0.6941 Acc: 0.4959
val Loss: 0.6908 Acc: 0.5425
```

[35]:
```
# Evaluate the model
model.eval()
running_corrects = 0
for inputs, labels in dataloaders['val']:
    inputs = inputs.to(device)
    labels = labels.to(device)
    # Assuming the first element of the tuple is the output we need
    outputs = model(inputs)[0]  # Access the first element of the tuple
    _, preds = torch.max(outputs, 1)
    running_corrects += torch.sum(preds == labels.data)
accuracy = running_corrects.double() / dataset_sizes['val']
print(f'Validation Accuracy: {accuracy:.4f}')
```

```
Validation Accuracy: 0.5425
```

[36]:
```
model_name = "ViT-B/32 configuration model"
```

[37]:
```
# Function to display images with predictions
def imshow(inp, title=None):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  #
```

Testing on Random Images #We will test the model on 5 random images from the validation

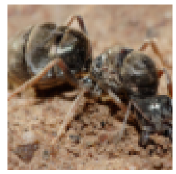
set and display the results.

[38]:
```
print(f"\nTesting 5 random images for {model_name}:")
model.eval()
images_so_far = 0
fig = plt.figure(figsize=(15, 10))
with torch.no_grad():
    for i, (inputs, labels) in enumerate(dataloaders['val']):
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        # Access the first element of the tuple, which is the output we need
        outputs = outputs[0]
        _, preds = torch.max(outputs, 1)

        for j in range(inputs.size()[0]):
            images_so_far += 1
            ax = plt.subplot(1, 5, images_so_far)
            ax.axis('off')
            ax.set_title(f'True: {class_names[labels[j]]}\nPred: {class_names[preds[j]]}')
            imshow(inputs.cpu().data[j])

            if images_so_far == 5:
                break
        if images_so_far == 5:
            break
plt.show()
```

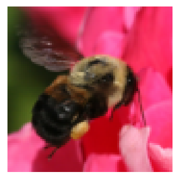
```
Testing 5 random images for ViT-B/32 configuration model:
```

True: bees
Pred: bees

True: ants
Pred: bees

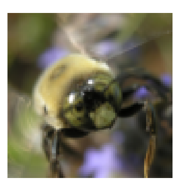


True: bees
Pred: bees



True: bees
Pred: bees

True: bees
Pred: bees



[ ]:
```
## Testing on ViT-L/16 configuration.
```

8.4 Configuration: ViT-L/16

- **Purpose**: Represents a larger Vision Transformer model.
- **Characteristics**:
  - Patch size: 16x16
  - Hidden size: 1024
  - Attention heads: 16
  - Transformer layers: 24
- **Use Case**: Suitable for more complex tasks requiring higher capacity.

[39]:
```python
def get_l16_config():
    """
    Returns the ViT-L/16 configuration.

    This configuration represents a larger Vision Transformer model with
    a patch size of 16x16, a hidden size of 1024, 16 attention heads, and
    24 transformer layers. It is suitable for more complex tasks requiring
    higher capacity.

    Returns:
        ConfigDict: A configuration dictionary for ViT-L/16.
    """
    config = ml_collections.ConfigDict()
    config.patches = ml_collections.ConfigDict({'size': (16, 16)})
    config.hidden_size = 1024
    config.transformer = ml_collections.ConfigDict()
    config.transformer.mlp_dim = 4096
    config.transformer.num_heads = 16
    config.transformer.num_layers = 24
    config.transformer.attention_dropout_rate = 0.0
    config.transformer.dropout_rate = 0.1
    config.classifier = 'token'
    config.representation_size = None
    return config
```

[40]:
```python
config = get_l16_config()
```

[41]:
```python
# Modify the ViT model for binary classification
model = VisionTransformer(config=config, num_classes=2)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
```

[42]:
```python
# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

[44]:
```python
# Training loop
num_epochs = 20
```

```python
for epoch in range(num_epochs):
    print(f'Epoch {epoch+1}/{num_epochs}')
    print('-' * 10)
    for phase in ['train', 'val']:
        if phase == 'train':
            model.train()
        else:
            model.eval()
        running_loss = 0.0
        running_corrects = 0
        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            with torch.set_grad_enabled(phase == 'train'):
                # Assuming the first element of the tuple is the output we need
                outputs = model(inputs)[0] # Access the first element of the␣
↪tuple
                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)
                if phase == 'train':
                    loss.backward()
                    optimizer.step()
            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)
        epoch_loss = running_loss / dataset_sizes[phase]
        epoch_acc = running_corrects.double() / dataset_sizes[phase]
        print(f'{phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')
```

```
Epoch 1/20
----------
```

```
---------------------------------------------------------------------------
OutOfMemoryError                          Traceback (most recent call last)
<ipython-input-44-787d05bdc1da> in <cell line: 0>()
     17             with torch.set_grad_enabled(phase == 'train'):
     18                 # Assuming the first element of the tuple is the output␣
↪we need
---> 19                 outputs = model(inputs)[0] # Access the first element o␣
↪the tuple
     20                 _, preds = torch.max(outputs, 1)
     21                 loss = criterion(outputs, labels)

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
↪_wrapped_call_impl(self, *args, **kwargs)
   1734             return self._compiled_call_impl(*args, **kwargs)  # type:␣
↪ignore[misc]
```

```
   1735         else:
-> 1736             return self._call_impl(*args, **kwargs)
   1737 
   1738     # torchrec tests the code consistency with the following code

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
 ↪_call_impl(self, *args, **kwargs)
   1745                 or _global_backward_pre_hooks or _global_backward_hooks
   1746                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1747             return forward_call(*args, **kwargs)
   1748 
   1749         result = None

/content/models/modeling.py in forward(self, x, labels)
    271 
    272     def forward(self, x, labels=None):
--> 273         x, attn_weights = self.transformer(x)
    274         logits = self.head(x[:, 0])
    275 

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
 ↪_wrapped_call_impl(self, *args, **kwargs)
   1734             return self._compiled_call_impl(*args, **kwargs)  # type:␣
 ↪ignore[misc]
   1735         else:
-> 1736             return self._call_impl(*args, **kwargs)
   1737 
   1738     # torchrec tests the code consistency with the following code

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
 ↪_call_impl(self, *args, **kwargs)
   1745                 or _global_backward_pre_hooks or _global_backward_hooks
   1746                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1747             return forward_call(*args, **kwargs)
   1748 
   1749         result = None

/content/models/modeling.py in forward(self, input_ids)
    256     def forward(self, input_ids):
    257         embedding_output = self.embeddings(input_ids)
--> 258         encoded, attn_weights = self.encoder(embedding_output)
    259         return encoded, attn_weights
    260 

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
 ↪_wrapped_call_impl(self, *args, **kwargs)
   1734             return self._compiled_call_impl(*args, **kwargs)  # type:␣
 ↪ignore[misc]
```

```
   1735         else:
-> 1736             return self._call_impl(*args, **kwargs)
   1737 
   1738     # torchrec tests the code consistency with the following code

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
 ↪_call_impl(self, *args, **kwargs)
   1745                 or _global_backward_pre_hooks or _global_backward_hooks
   1746                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1747             return forward_call(*args, **kwargs)
   1748 
   1749         result = None

/content/models/modeling.py in forward(self, hidden_states)
    241         attn_weights = []
    242         for layer_block in self.layer:
--> 243             hidden_states, weights = layer_block(hidden_states)
    244             if self.vis:
    245                 attn_weights.append(weights)

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
 ↪_wrapped_call_impl(self, *args, **kwargs)
   1734             return self._compiled_call_impl(*args, **kwargs)  # type:␣
 ↪ignore[misc]
   1735         else:
-> 1736             return self._call_impl(*args, **kwargs)
   1737 
   1738     # torchrec tests the code consistency with the following code

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
 ↪_call_impl(self, *args, **kwargs)
   1745                 or _global_backward_pre_hooks or _global_backward_hooks
   1746                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1747             return forward_call(*args, **kwargs)
   1748 
   1749         result = None

/content/models/modeling.py in forward(self, x)
    181         h = x
    182         x = self.attention_norm(x)
--> 183         x, weights = self.attn(x)
    184         x = x + h
    185 

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
 ↪_wrapped_call_impl(self, *args, **kwargs)
   1734             return self._compiled_call_impl(*args, **kwargs)  # type:␣
 ↪ignore[misc]
```

```
   1735         else:
-> 1736             return self._call_impl(*args, **kwargs)
   1737 
   1738     # torchrec tests the code consistency with the following code

/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py in␣
↪_call_impl(self, *args, **kwargs)
   1745                 or _global_backward_pre_hooks or _global_backward_hooks
   1746                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1747             return forward_call(*args, **kwargs)
   1748 
   1749         result = None

/content/models/modeling.py in forward(self, hidden_states)
     82         value_layer = self.transpose_for_scores(mixed_value_layer)
     83 
---> 84         attention_scores = torch.matmul(query_layer, key_layer.
↪transpose(-1, -2))
     85         attention_scores = attention_scores / math.sqrt(self.
↪attention_head_size)
     86         attention_probs = self.softmax(attention_scores)

OutOfMemoryError: CUDA out of memory. Tried to allocate 26.00 MiB. GPU 0 has a␣
↪total capacity of 14.74 GiB of which 10.12 MiB is free. Process 5330 has 14.73␣
↪GiB memory in use. Of the allocated memory 13.84 GiB is allocated by PyTorch,␣
↪and 770.06 MiB is reserved by PyTorch but unallocated. If reserved but␣
↪unallocated memory is large try setting␣
↪PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True to avoid fragmentation.  See␣
↪documentation for Memory Management  (https://pytorch.org/docs/stable/notes/
↪cuda.html#environment-variables)
```

[ ]:
```
# Evaluate the model
model.eval()
running_corrects = 0
for inputs, labels in dataloaders['val']:
    inputs = inputs.to(device)
    labels = labels.to(device)
    # Assuming the first element of the tuple is the output we need
    outputs = model(inputs)[0]  # Access the first element of the tuple
    _, preds = torch.max(outputs, 1)
    running_corrects += torch.sum(preds == labels.data)
accuracy = running_corrects.double() / dataset_sizes['val']
print(f'Validation Accuracy: {accuracy:.4f}')
```

[ ]:
```
model_name = "ViT-L/16 configuration model"
```

[ ]:
```
# Function to display images with predictions
def imshow(inp, title=None):
```

```
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  #
```

Testing on Random Images #We will test the model on 5 random images from the validation set and display the results.

```
[ ]: print(f"\nTesting 5 random images for {model_name}:")
model.eval()
images_so_far = 0
fig = plt.figure(figsize=(15, 10))
with torch.no_grad():
    for i, (inputs, labels) in enumerate(dataloaders['val']):
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        # Access the first element of the tuple, which is the output we need
        outputs = outputs[0]
        _, preds = torch.max(outputs, 1)

        for j in range(inputs.size()[0]):
            images_so_far += 1
            ax = plt.subplot(1, 5, images_so_far)
            ax.axis('off')
            ax.set_title(f'True: {class_names[labels[j]]}\nPred:␣
↪{class_names[preds[j]]}')
            imshow(inputs.cpu().data[j])

            if images_so_far == 5:
                break
        if images_so_far == 5:
            break
plt.show()
```

8.5 Configuration: ViT-H/14

- **Purpose**: Represents a high-capacity Vision Transformer model.
- **Characteristics**:
  - Patch size: 14x14
  - Hidden size: 1280

- Attention heads: 16
- Transformer layers: 32
- **Use Case**: Designed for very complex tasks requiring significant computational resources.

[ ]:
```
##testing on ViT-H/14 configuration.
```

[ ]:
```
def get_h14_config():
    """
    Returns the ViT-H/14 configuration.

    This configuration represents a high-capacity Vision Transformer model
    with a patch size of 14x14, a hidden size of 1280, 16 attention heads,
    and 32 transformer layers. It is designed for very complex tasks
    requiring significant computational resources.

    Returns:
        ConfigDict: A configuration dictionary for ViT-H/14.
    """
    config = ml_collections.ConfigDict()
    config.patches = ml_collections.ConfigDict({'size': (14, 14)})
    config.hidden_size = 1280
    config.transformer = ml_collections.ConfigDict()
    config.transformer.mlp_dim = 5120
    config.transformer.num_heads = 16
    config.transformer.num_layers = 32
    config.transformer.attention_dropout_rate = 0.0
    config.transformer.dropout_rate = 0.1
    config.classifier = 'token'
    config.representation_size = None
    return config
```

[ ]:
```
config = get_h14_config()
```

[ ]:
```
# Modify the ViT model for binary classification
model = VisionTransformer(config=config, num_classes=2)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
```

[ ]:
```
# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

[ ]:
```
# Training loop
num_epochs = 50
for epoch in range(num_epochs):
    print(f'Epoch {epoch+1}/{num_epochs}')
    print('-' * 10)
```

```
    for phase in ['train', 'val']:
        if phase == 'train':
            model.train()
        else:
            model.eval()
        running_loss = 0.0
        running_corrects = 0
        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            with torch.set_grad_enabled(phase == 'train'):
                # Assuming the first element of the tuple is the output we need
                outputs = model(inputs)[0] # Access the first element of the␣
 ↪tuple
                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)
                if phase == 'train':
                    loss.backward()
                    optimizer.step()
            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)
        epoch_loss = running_loss / dataset_sizes[phase]
        epoch_acc = running_corrects.double() / dataset_sizes[phase]
        print(f'{phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')
```

[ ]:
```
# Evaluate the model
model.eval()
running_corrects = 0
for inputs, labels in dataloaders['val']:
    inputs = inputs.to(device)
    labels = labels.to(device)
    # Assuming the first element of the tuple is the output we need
    outputs = model(inputs)[0]  # Access the first element of the tuple
    _, preds = torch.max(outputs, 1)
    running_corrects += torch.sum(preds == labels.data)
accuracy = running_corrects.double() / dataset_sizes['val']
print(f'Validation Accuracy: {accuracy:.4f}')
```

[ ]:
```
model_name = " ViT-L/14 configuration model"
```

[ ]:
```
# Function to display images with predictions
def imshow(inp, title=None):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
```

```
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  #
```

Testing on Random Images #We will test the model on 5 random images from the validation set and display the results.

```
[ ]: print(f"\nTesting 5 random images for {model_name}:")
model.eval()
images_so_far = 0
fig = plt.figure(figsize=(15, 10))
with torch.no_grad():
    for i, (inputs, labels) in enumerate(dataloaders['val']):
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        # Access the first element of the tuple, which is the output we need
        outputs = outputs[0]
        _, preds = torch.max(outputs, 1)

        for j in range(inputs.size()[0]):
            images_so_far += 1
            ax = plt.subplot(1, 5, images_so_far)
            ax.axis('off')
            ax.set_title(f'True: {class_names[labels[j]]}\nPred:␣
↪{class_names[preds[j]]}')
            imshow(inputs.cpu().data[j])

            if images_so_far == 5:
                break
        if images_so_far == 5:
            break
plt.show()
```