Model reports:

Epoch 1: train\_acc = 0.49239999055862427, train\_loss = 1.4454176425933838, val\_acc = 0.546999990940094, val\_loss = 1.2732207775115967

Epoch 2: train\_acc = 0.5564000010490417, train\_loss = 1.2631537914276123, val\_acc = 0.5645999908447266, val\_loss = 1.2316303253173828

Epoch 3: train\_acc = 0.5760999917984009, train\_loss = 1.2067872285842896, val\_acc = 0.5764999985694885, val\_loss = 1.2226899862289429

Epoch 4: train\_acc = 0.5889599919319153, train\_loss = 1.1756646633148193, val\_acc = 0.5922999978065491, val\_loss = 1.15999436378479

Epoch 5: train\_acc = 0.5978000164031982, train\_loss = 1.1415517330169678, val\_acc = 0.5958999991416931, val\_loss = 1.1553505659103394

Epoch 6: train\_acc = 0.6071000099182129, train\_loss = 1.121337890625, val\_acc = 0.5952000021934509, val\_loss = 1.154342770576477

Epoch 7: train\_acc = 0.6115400195121765, train\_loss = 1.1105847358703613, val\_acc = 0.5993000268936157, val\_loss = 1.1524652242660522

Epoch 8: train\_acc = 0.6171600222587585, train\_loss = 1.0885009765625, val\_acc = 0.603600025177002, val\_loss = 1.1494483947753906

Epoch 9: train\_acc = 0.6207399964332581, train\_loss = 1.076804518699646, val\_acc = 0.6061999797821045, val\_loss = 1.1354519128799438

Epoch 10: train\_acc = 0.6283000111579895, train\_loss = 1.0581824779510498, val\_acc = 0.608299970626831, val\_loss = 1.12826406955719

Epoch 11: train\_acc = 0.633679986000061, train\_loss = 1.042823076248169, val\_acc = 0.6080999970436096, val\_loss = 1.1275713443756104

Epoch 12: train\_acc = 0.6355400085449219, train\_loss = 1.034339189529419, val\_acc = 0.6068999767303467, val\_loss = 1.1327663660049438

Epoch 13: train\_acc = 0.638700008392334, train\_loss = 1.0210871696472168, val\_acc = 0.6087999939918518, val\_loss = 1.140114188194275

Epoch 14: train\_acc = 0.6426600217819214, train\_loss = 1.0095363855361938, val\_acc = 0.6032999753952026, val\_loss = 1.131926417350769

Epoch 15: train\_acc = 0.644860029220581, train\_loss = 1.0065206289291382, val\_acc = 0.6164000034332275, val\_loss = 1.1145144701004028

Epoch 16: train\_acc = 0.6507200002670288, train\_loss = 0.9839068055152893, val\_acc = 0.607699990272522, val\_loss = 1.1372569799423218

Epoch 17: train\_acc = 0.6523399949073792, train\_loss = 0.9817643761634827, val\_acc = 0.6116999983787537, val\_loss = 1.140908122062683

Epoch 18: train\_acc = 0.6562600135803223, train\_loss = 0.9702702164649963, val\_acc = 0.6140000224113464, val\_loss = 1.1231436729431152

Epoch 19: train\_acc = 0.6588000059127808, train\_loss = 0.959434986114502, val\_acc = 0.6137999892234802, val\_loss = 1.1232237815856934

Epoch 20: train\_acc = 0.663100004196167, train\_loss = 0.9491212368011475, val\_acc = 0.6151000261306763, val\_loss = 1.140295147895813

Epoch 21: train\_acc = 0.6640999913215637, train\_loss = 0.9486017823219299, val\_acc = 0.6186000108718872, val\_loss = 1.129909634590149

Epoch 22: train\_acc = 0.665880024433136, train\_loss = 0.9384665489196777, val\_acc = 0.6118000149726868, val\_loss = 1.1325682401657104

Epoch 23: train\_acc = 0.6675000190734863, train\_loss = 0.932896077632904, val\_acc = 0.6126000285148621, val\_loss = 1.1381505727767944

Epoch 24: train\_acc = 0.6765400171279907, train\_loss = 0.9155426621437073, val\_acc = 0.6209999918937683, val\_loss = 1.1362708806991577

Epoch 25: train\_acc = 0.6761400103569031, train\_loss = 0.907611072063446, val\_acc = 0.6115999817848206, val\_loss = 1.1350665092468262

Epoch 26: train\_acc = 0.6784800291061401, train\_loss = 0.9031552076339722, val\_acc = 0.6100000143051147, val\_loss = 1.154421329498291

Epoch 27: train\_acc = 0.6803399920463562, train\_loss = 0.8972927331924438, val\_acc = 0.6152999997138977, val\_loss = 1.1450783014297485

Epoch 28: train\_acc = 0.6828799843788147, train\_loss = 0.8882137537002563, val\_acc = 0.6148999929428101, val\_loss = 1.1281012296676636

Epoch 29: train\_acc = 0.684499979019165, train\_loss = 0.8842329978942871, val\_acc = 0.6136999726295471, val\_loss = 1.1688389778137207

Epoch 30: train\_acc = 0.6851199865341187, train\_loss = 0.882375180721283, val\_acc = 0.6166999936103821, val\_loss = 1.1311962604522705

Epoch 31: train\_acc = 0.6900799870491028, train\_loss = 0.8664125204086304, val\_acc = 0.6168000102043152, val\_loss = 1.1392914056777954

Epoch 32: train\_acc = 0.6973000168800354, train\_loss = 0.8493896722793579, val\_acc = 0.6108999848365784, val\_loss = 1.1641671657562256

Epoch 33: train\_acc = 0.6967200040817261, train\_loss = 0.847392201423645, val\_acc = 0.6248999834060669, val\_loss = 1.1323264837265015

Epoch 34: train\_acc = 0.6943600177764893, train\_loss = 0.8479257225990295, val\_acc = 0.6236000061035156, val\_loss = 1.1310278177261353

Epoch 35: train\_acc = 0.6989799737930298, train\_loss = 0.8407227396965027, val\_acc = 0.613099992275238, val\_loss = 1.1600738763809204

Epoch 36: train\_acc = 0.7044600248336792, train\_loss = 0.8303477168083191, val\_acc = 0.621999979019165, val\_loss = 1.1242709159851074

Epoch 37: train\_acc = 0.7064399719238281, train\_loss = 0.8175704479217529, val\_acc = 0.6176999807357788, val\_loss = 1.139251708984375

Epoch 38: train\_acc = 0.7092999815940857, train\_loss = 0.814103364944458, val\_acc = 0.6187999844551086, val\_loss = 1.1693387031555176

Epoch 39: train\_acc = 0.7095000147819519, train\_loss = 0.8102928400039673, val\_acc = 0.6182000041007996, val\_loss = 1.1569620370864868

Epoch 40: train\_acc = 0.7107200026512146, train\_loss = 0.8107662796974182, val\_acc = 0.619700014591217, val\_loss = 1.1482346057891846

Compare the performance gap between the pre-trained DCNN model in the condition of transfer learning, and **your customized model in assignment 2** in the condition of training from scratch. Explain why you obtained such results, and give a brief discussion about it.

**Answer:**

First, let’s talk about our customs model works. We train two model architectures. The 1st architecture got good results. We got 70% accuracy on our test or validation dataset. Our 2nd architecture was not soo much good and we got an accuracy of only the 60% on the validation. Our both architectures were different in the learning rates, batch sizes, and epochs. So we can say that the learning rates matter a lot for accuracy purposes. In our 2nd architecture, we got fewer accuracies may be due to the learning rates, we stuck on some local minima and got fewer accuracies. And in 1st architecture, we succeed to reach the global minima.

Now we train the pre-trained model our results are not good as compared to the customs models.

After this we extract features using deep feature techniques and then apply the SVM model and then also apply the custom model work to see the results so it is also providing the same accuracies as the 2nd architectures.

So, in short, we can say that our 1st architecture was working fine as compared to the other. It can be due to the best learning rate selection and best parameters selection that suit our dataset. As we already discussed the importance of the learning rate. So we select the best learning rates and we succeed to reach global minima instead of being stuck at the local minima.