# 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
```

**Q.** 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.

**A :** Let's first discuss how our customs model functions. Two model architectures are trained. The initial architecture was successful. In the test or validation dataset, we achieved a 70% accuracy rate. Our second architecture wasn't very good, and the validation accuracy was barely 60%. There were differences between our two architectures in terms of learning rates, batch sizes, and epochs. So, we might conclude that learning rates are crucial for accuracy. Because to the learning rates, we may have remained on some local minima in our second architecture, which is why we obtained fewer accuracy values. And we are able to reach the global minima in the first architecture.

The pre-trained model is now being trained. Compared to the custom models, our outcomes are subpar.

Afterwards, we use the SVM model and the custom model work to see the results, extracting features using deep feature techniques in order to achieve accuracy levels comparable to those of the second architectures.

In summary, we may state that our first architecture performed better than the others. It can be because we chose the best learning rate and the optimal settings for our dataset. As already mentioned, learning pace is crucial. In order to reach the global minima rather than remain at the local minima, we choose the best learning rates.