Assignment 2: Experiment 6
Team 10

Apply fine tuning for your team's network to the CIFAR 10 dataset.

Team's network: DenseNet169

Fine tuning is a technique to use pre-trained models and how to extract features from them for training a model for a different task.

In fine tuning we freeze the initial layers and retain the later layers for our task.As explained here, the initial layers learn very general features and as we go higher up the network, the layers tend to learn patterns more specific to the task it is being trained on.

In addition to the base DenseNet169 a dense layer and a softmax were added to classify the output.

Experiments performed:

1. **Training only the last convolutional layer:**
   Results: Best test accuracy and test loss obtained in this method (using various dropouts) was 0.4408 and 2.7471.
2. **Training the last two convolutional layers:**
   Results: Best test accuracy and test loss obtained were 0.4672 and 2.8262 respectively.
3. **Training the last three convolutional layers:**
   Results: An accuracy of 0.4296 and a loss of 2.8412 were observed.

   Results from other experiments were listed below.

Assignment 2: Experiment 6
Team 10

Experiments:

Base Model: Trainable layers: base_model.layers[:-4]

Unfreeze the last conv layer, pooling and norm layers.

Base model: model.compile(loss='categorical_crossentropy',

optimizer = optimizers.Adam(lr=1e-4), metrics=['acc'])

Model 1:

Added: ##Add new Dense layer and a softmax to classify.

model.add(layers.Flatten())

model.add(layers.Dense(1024, activation = 'relu'))

model.add(layers.Dense(10,activation='softmax'))

Result after 10 epochs:

```
Epoch 8/10
50000/50000 [==============================] - 24s 479us/step - loss: 1.0363 - acc: 0.6457 - val_loss: 3.0180 - val_acc: 0.4350
Epoch 9/10
50000/50000 [==============================] - 24s 478us/step - loss: 1.0033 - acc: 0.6568 - val_loss: 3.0200 - val_acc: 0.4331
Epoch 10/10
50000/50000 [==============================] - 24s 480us/step - loss: 0.9780 - acc: 0.6635 - val_loss: 2.9883 - val_acc: 0.4334
```

Model 2:

model.add(layers.Flatten())

model.add(layers.Dense(1024, activation = 'relu'))

model.add(layers.Dropout(0.2)) **##Added a dropout to base model**

model.add(layers.Dense(10,activation='softmax'))

```
50000/50000 [==============================] - 24s 482us/step - loss: 1.0515 - acc: 0.6362 - val_loss: 3.0558 - val_acc: 0.4264
Epoch 9/10
50000/50000 [==============================] - 24s 482us/step - loss: 1.0201 - acc: 0.6487 - val_loss: 2.8506 - val_acc: 0.4354
Epoch 10/10
50000/50000 [==============================] - 24s 482us/step - loss: 0.9941 - acc: 0.6558 - val_loss: 2.7471 - val_acc: 0.4408
```

Model 3: All same as model 2 but changed Adam to **RMSprop**

```
Epoch 8/10
50000/50000 [==============================] - 24s 480us/step - loss: 0.8558 - acc: 0.7040 - val_loss: 2.7552 - val_acc: 0.4346
Epoch 9/10
50000/50000 [==============================] - 24s 481us/step - loss: 0.8397 - acc: 0.7112 - val_loss: 2.7599 - val_acc: 0.4330
Epoch 10/10
50000/50000 [==============================] - 24s 480us/step - loss: 0.8283 - acc: 0.7165 - val_loss: 2.8143 - val_acc: 0.4281
```

Model 4: All same as base but learning rate is **1e-5, base_model.layers[:-8] ##Unfreeze the last 2 conv2D layers of base model**

```
Epoch 8/10
50000/50000 [==============================] - 25s 497us/step - loss: 1.6623 - acc: 0.4321 - val_loss: 2.8135 - val_acc: 0.3512
Epoch 9/10
50000/50000 [==============================] - 25s 496us/step - loss: 1.5946 - acc: 0.4562 - val_loss: 2.8753 - val_acc: 0.3597
Epoch 10/10
50000/50000 [==============================] - 25s 497us/step - loss: 1.5389 - acc: 0.4712 - val_loss: 2.9412 - val_acc: 0.3664
```

Assignment 2: Experiment 6
Team 10

Model 5: Same as 3 with **dropout 0.3**

```
Epoch 8/10
50000/50000 [==============================] - 24s 482us/step - loss: 1.0490 - acc: 0.6379 - val_loss: 2.7414 - val_acc: 0.4468
Epoch 9/10
50000/50000 [==============================] - 24s 483us/step - loss: 1.0240 - acc: 0.6465 - val_loss: 2.8056 - val_acc: 0.4366
Epoch 10/10
50000/50000 [==============================] - 24s 483us/step - loss: 0.9961 - acc: 0.6527 - val_loss: 2.7466 - val_acc: 0.4352
```

Model 6: All same as base but learning rate is **1e-5, base_model.layers[:-7]** and 30 epochs

```
Epoch 26/30
50000/50000 [==============================] - 23s 459us/step - loss: 1.0234 - acc: 0.6388 - val_loss: 2.7690 - val_acc: 0.4678
Epoch 27/30
50000/50000 [==============================] - 23s 460us/step - loss: 1.0160 - acc: 0.6431 - val_loss: 2.8146 - val_acc: 0.4657
Epoch 28/30
50000/50000 [==============================] - 23s 459us/step - loss: 1.0064 - acc: 0.6454 - val_loss: 2.8397 - val_acc: 0.4651
Epoch 29/30
50000/50000 [==============================] - 23s 459us/step - loss: 0.9969 - acc: 0.6520 - val_loss: 2.8170 - val_acc: 0.4670
Epoch 30/30
50000/50000 [==============================] - 23s 460us/step - loss: 0.9905 - acc: 0.6520 - val_loss: 2.8262 - val_acc: 0.4672
```

Model 7: All same as base with unfreeze the last 3 conv layers:

```
Epoch 28/30
50000/50000 [==============================] - 23s 465us/step - loss: 1.0826 - acc: 0.6219 - val_loss: 2.8498 - val_acc: 0.4282
Epoch 29/30
50000/50000 [==============================] - 23s 466us/step - loss: 1.0724 - acc: 0.6266 - val_loss: 2.8935 - val_acc: 0.4253
Epoch 30/30
50000/50000 [==============================] - 23s 465us/step - loss: 1.0579 - acc: 0.6315 - val_loss: 2.8412 - val_acc: 0.4296
```