سوال اول

Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. Intuitively, underfitting occurs when the model or the algorithm does not fit the data well enough. Specifically, underfitting occurs if the model or algorithm shows low variance but high bias.
A model is underfitting the training data when the model performs poorly on the training data. This is because the model is unable to capture the relationship between the input examples and the target values.
A model is overfitting the training data when you see that the model performs well on the training data but does not perform well on the evaluation data. This is because the model is memorizing the data it has seen and is unable to generalize to unseen examples. A model that is underfit will have high training and high testing error while an overfit model will have extremely low training error but a high testing error. If your model is overfitting the training data, it makes sense to take actions that reduce model flexibility. To reduce model flexibility, these can be done:

- Feature selection: consider using fewer feature combinations, decrease n-grams size, and decrease the number of numeric attribute bins.
- Increase the amount of regularization used.
- Increase the amount of training data examples.
- Increase the number of passes on the existing training data.

سوال دوم

In deep learning, transfer learning is a technique whereby a neural network model is first trained on a problem similar to the problem that is being solved. One or more layers from the trained model are then used in a new model trained on the problem of interest. Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error. They are also, super simple to incorporate, achieve solid (same or even better) model performance quickly, there's not as much labeled data required and have versatile uses cases from transfer learning, prediction, and feature extraction. The weights in re-used layers may be used as the starting point for the training process and adapted in response to the new problem. This usage treats transfer learning as a type of weight initialization scheme.

https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/

سوال سوم

Normal convolution:
(width of filter * height filter * depth +1) * number of filters = (5 * 5 * 3 + 1) x 128 = 76 * 128 = 9728

Depth-wise Separable Convolution:
(5*5 + 1) * 2 + (3*2 + 1) * 128 = 948

The main difference is this: in the normal convolution, we are transforming the image 128 times. And every transformation uses up 5x5x3x(28-5+1)x(28-5+1) multiplications. In the separable convolution, we only really transform the image once — in the depth-wise convolution. Then, we take the transformed image and simply elongate it to 128 channels. Without having to transform the image over and over again, we can save up on computational power. Also, having too many parameters forces function to memorize rather than learn and thus leads to over-fitting. Depth-wise separable convolution prevents that.

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

سوال چهارم

Colab Link: https://drive.google.com/file/d/1EzZnQEx667oZHJGGx5zarRlDM3Kzw5PK/view?usp=sharing

Part A)
The model is a typical Resnet50 model using average pooling, with random weights specified by "weights = None" in its declaration. The data generator uses the Resnet50 default preprocessor function and since we don't want augmented data, nothing else needs to be added. The same is true for the test data. In this part, because we're not using augmented data, the model gets overfitted and is not expected to accurately predict the test labels.

Part B)
The model is the same as Part A, but the training data is augmented. The results are expected to be more accurate than Part A because there's less chance of the model being overfitted. The best difference between the accuracy of these two parts has been the following:

```
101/101 [==============================] - 70s 679ms/step - loss: 6.4001 -
accuracy: 0.2022
[6.400067329406738, 0.20221365988254547]
```

```
101/101 [==============================] - 73s 717ms/step - loss: 2.9122 -
accuracy: 0.2365
[2.912196731567383, 0.25653773963451385]
```

Part C)
In this part the model changes to a sequential model including a pretrained Resnet50 with imagenet, GlobalAveragePooling, and two Dense layers to correctly classify the data into the number of classes we want. As expected, fine tuning results in a better accuracy on the test data than freezing the ResNet50 weights. Comparing to the previous parts, the overfitted model gives the worst result, and fine tuning has the best one. Using

augmented train data on a pretrained Resnet yields better results on a randomly weighted Resnet.
And as expected, using a partially-pretrained model took much less time to get to the same accuracy levels as

Links used (besides the ones in the assignment doc):

https://medium.com/@kenneth.ca95/a-guide-to-transfer-learning-with-keras-using-resnet50-a81a4a28084b

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/transfer_learning.ipynb#scrollTo=4nzcagVitLQm

https://github.com/Kenneth-ca/holbertonschool-machine_learning/blob/master/supervised_learning/0x09-transfer_learning/0_transfer.ipynb

https://cv-tricks.com/keras/understand-implement-resnets/

زمان کل تمرین
بدون در نظر گرفتن زمان train: حدود هشت ساعت.