DataLab Preparation (Week 4, DataLab II, Thursday)

3. Data Augmentation

**3c Considering the following cases, which comprise the original image on the left and a** augmented image on the right, indicate what argument of the ImageDataGenerator was used to create the augmented image.

Case 1 - tf.keras.layers.RandomContrast

Case 2 - horizontal\_flip=True

Case 3 - rotation\_range=30

Case 4 - width\_shift\_range=0.2

Case 5 - horizontal\_flip=True and height\_shift\_range=0.2

4. Transfer Learning

**4a Explain how transfer learning based on feature extraction from a pretrained model works.**

Transfer learning based on feature extraction from a pre-trained model consists of using the convolutional base of the model to extract interesting features from new samples.

If the new dataset differs a lot from the dataset on which the original model was trained, it is more useful using only the first few layers of the model to do feature extraction, rather than using the entire convolutional base simply because representations found in densely connected layers are not as generic, and therefore not as reusable.

We begin by instantiating the convolutional base of our chosen model trained on a particular dataset, specifying the weight checkpoint from which to initialize the model, wether or not to include the densely connected classifier on top of the network, and the input shape of the image tensors we will feed into the network (\*optional).

The final feature map after extracting the convolutional base is where we will stick a densely connected classifier.

Afterwards, there are two ways to proceed. Either we run the convolutional base over our dataset and then use the data as input to a standalone, densely connected classifier, which is fast and cheap to run, but you cannot use data augmentation, or extend the model by adding Dense layers on top and run the whole thing from end to end on the input data, which will allow us to use data aug., but it is far more expensive.

Because the first option does not allow us to perform data augmentation, the model is more prone to overfitting with small image datasets. On the other hand, it can be run even without a GPU.

With the second option, we will get much better results in terms of accuracy and we can also train it using more epochs since we can apply data augmentation.

The main difference in applying this method consists in the extra step for the second option in which you first freeze the convolutional base to prevent the layers' weights from being updated during training.

**4b Explain how transfer learning based on fine-tuning a pretrained model works.**

Fine-tuning consists of unfreezing a few of the top layers of a frozen model base used for feature extraction, and jointly training both the newly added part of the model and these top layers.

It is only possible to fine-tune the top layers of the convolutional base once the classifier on top has already been trained. If the classifier is not already trained, the error signal propagating though the network during training will be too large, and the representations previously learned by the layers being fine-tuned will be destroyed.

Steps to fine0tuning a pre-trained model:

1. Add custom network on top of an already-trained base network

2. Freeze the base network

3. Train the needed part

4. Unfreeze some layers in the base network. Batch Normalization layers should not be unfrozen.

5. Jointly train both layers and the added part

It is generally better to fine-tune more specialized features because these are the ones repurposed to solve the problem at hand. Earlier layers in the conv base encode more generic, reusable features. Additionally, the more parameters are trained, the higher the risk of overfitting.

**4c Keras provides several pre-trained models for image classification (see the list of models here). Pick 5 of these models and describe their main features.**

VGG16

1. include\_top: whether to include the 3 fully-connected layers at the top of the network.
2. weights: one of None (random initialization), "imagenet" (pre-training on ImageNet), or the path to the weights file to be loaded.
3. input\_tensor: optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model.
4. input\_shape: optional shape tuple, only to be specified if include\_top is False (otherwise the input shape has to be (224, 224, 3) (with channels\_last data format) or (3, 224, 224) (with "channels\_first" data format). It should have exactly 3 input channels, and width and height should be no smaller than 32. E.g. (200, 200, 3) would be one valid value.
5. pooling: Optional pooling mode for feature extraction when include\_top is False.
6. None means that the output of the model will be the 4D tensor output of the last convolutional block.
7. avg means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
8. max means that global max pooling will be applied.
9. classes: optional number of classes to classify images into, only to be specified if include\_top is True, and if no weights argument is specified.
10. classifier\_activation: A str or callable. The activation function to use on the "top" layer. Ignored unless include\_top=True. Set classifier\_activation=None to return the logits of the "top" layer. When loading pretrained weights, classifier\_activation can only be None or "softmax".

Xception

1. include\_top: whether to include the 3 fully-connected layers at the top of the network.
2. weights: one of None (random initialization), "imagenet" (pre-training on ImageNet), or the path to the weights file to be loaded.
3. input\_tensor: optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model.
4. input\_shape: optional shape tuple, only to be specified if include\_top is False (otherwise the input shape has to be (299, 299, 3). It should have exactly 3 inputs channels, and width and height should be no smaller than 71. E.g. (150, 150, 3) would be one valid value.
5. pooling: Optional pooling mode for feature extraction when include\_top is False.
6. None means that the output of the model will be the 4D tensor output of the last convolutional block.
7. avg means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
8. max means that global max pooling will be applied.
9. classes: optional number of classes to classify images into, only to be specified if include\_top is True, and if no weights argument is specified.
10. classifier\_activation: A str or callable. The activation function to use on the "top" layer. Ignored unless include\_top=True. Set classifier\_activation=None to return the logits of the "top" layer. When loading pretrained weights, classifier\_activation can only be None or "softmax".

ResNet

1. include\_top: whether to include the fully-connected layer at the top of the network.
2. weights: one of None (random initialization), "imagenet" (pre-training on ImageNet), or the path to the weights file to be loaded.
3. input\_tensor: optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model.
4. input\_shape: optional shape tuple, only to be specified if include\_top is False (otherwise the input shape has to be (224, 224, 3) (with "channels\_last" data format) or (3, 224, 224) (with "channels\_first" data format). It should have exactly 3 inputs channels, and width and height should be no smaller than 32. E.g. (200, 200, 3) would be one valid value.
5. pooling: Optional pooling mode for feature extraction when include\_top is False.
6. None means that the output of the model will be the 4D tensor output of the last convolutional block.
7. avg means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
8. max means that global max pooling will be applied.
9. classes: optional number of classes to classify images into, only to be specified if include\_top is True, and if no weights argument is specified.
10. classifier\_activation: A str or callable. The activation function to use on the "top" layer. Ignored unless include\_top=True. Set classifier\_activation=None to return the logits of the "top" layer. When loading pretrained weights, classifier\_activation can only be None or "softmax".

MobileNet

1. input\_shape: Optional shape tuple, only to be specified if include\_top is False (otherwise the input shape has to be (224, 224, 3) (with "channels\_last" data format) or (3, 224, 224) (with "channels\_first" data format). It should have exactly 3 inputs channels, and width and height should be no smaller than 32. E.g. (200, 200, 3) would be one valid value. Defaults to None. input\_shape will be ignored if the input\_tensor is provided.
2. alpha: Controls the width of the network. This is known as the width multiplier in the MobileNet paper.
3. If alpha < 1.0, proportionally decreases the number of filters in each layer.
4. If alpha > 1.0, proportionally increases the number of filters in each layer.
5. If alpha == 1, default number of filters from the paper are used at each layer. Defaults to 1.0.
6. depth\_multiplier: Depth multiplier for depthwise convolution. This is called the resolution multiplier in the MobileNet paper. Defaults to 1.0.
7. dropout: Dropout rate. Defaults to 0.001.
8. include\_top: Boolean, whether to include the fully-connected layer at the top of the network. Defaults to True.
9. weights: One of None (random initialization), "imagenet" (pre-training on ImageNet), or the path to the weights file to be loaded. Defaults to "imagenet".
10. input\_tensor: Optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model. input\_tensor is useful for sharing inputs between multiple different networks. Defaults to None.
11. pooling: Optional pooling mode for feature extraction when include\_top is False.
12. None (default) means that the output of the model will be the 4D tensor output of the last convolutional block.
13. avg means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
14. max means that global max pooling will be applied.
15. classes: Optional number of classes to classify images into, only to be specified if include\_top is True, and if no weights argument is specified. Defaults to 1000.
16. classifier\_activation: A str or callable. The activation function to use on the "top" layer. Ignored unless include\_top=True. Set classifier\_activation=None to return the logits of the "top" layer. When loading pretrained weights, classifier\_activation can only be None or "softmax".

EfficientNet B0

1. include\_top: Whether to include the fully-connected layer at the top of the network. Defaults to True.
2. weights: One of None (random initialization), "imagenet" (pre-training on ImageNet), or the path to the weights file to be loaded. Defaults to "imagenet".
3. input\_tensor: Optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model.
4. input\_shape: Optional shape tuple, only to be specified if include\_top is False. It should have exactly 3 inputs channels.
5. pooling: Optional pooling mode for feature extraction when include\_top is False. Defaults to None.
6. None means that the output of the model will be the 4D tensor output of the last convolutional layer.
7. avg means that global average pooling will be applied to the output of the last convolutional layer, and thus the output of the model will be a 2D tensor.
8. max means that global max pooling will be applied.
9. classes: Optional number of classes to classify images into, only to be specified if include\_top is True, and if no weights argument is specified. 1000 is how many ImageNet classes there are. Defaults to 1000.
10. classifier\_activation: A str or callable. The activation function to use on the "top" layer. Ignored unless include\_top=True. Set classifier\_activation=None to return the logits of the "top" layer. Defaults to 'softmax'. When loading pretrained weights, classifier\_activation can only be None or "softmax".