**SWS3009 Summer Workshop**

**Lab 4 – Answer Book**

**SUBMISSION DEADLINE: SUNDAY 9 JULY 2023, 11.59 PM**

Name 1:YANG SIMIN

Name 2:PAN DEYU

**QUESTION 1.**

**First,** **the Sequential API is a straightforward and simple way to build neural network models where the data flows sequentially through the layers. It allows to create models by stacking layers on top of each other. It is suitable for building models that have a single input and a single output. It is easier to use and understand, especially for beginners, as it follows a linear, sequential structure.**

**Second, the Model API is a more flexible and powerful way to build neural network models. It allows for more complex architectures with multiple inputs and outputs, shared layers, and branching models. It provides a functional API where you can define the connections between layers more explicitly. It is suitable for building models that have multiple inputs or outputs, models with shared layers, or models with complex branching architectures.**

**In summary, the Sequential API is used for simpler, sequential models with a single input and output, while the Model API provides more flexibility and control over the model architecture, allowing for more complex models with multiple inputs and outputs, shared layers, and branching structures. The choice between the two depends on the requirements and complexity of the model that the programmers want to build.**

**QUESTION 2.**

**We do this for several reasons :**

**First,** **increased Data Variety: Data augmentation allows us to generate new samples with variations that are likely to be encountered in real-world scenarios. By applying transformations such as rotations, translations, or flips, we can simulate different angles, positions, or orientations of objects in images. This helps the model generalize better and improves its ability to handle different variations in the test data.**

**Second, improved Model Robustness: Data augmentation helps to make the model more robust to variations and noise in the input data. By exposing the model to a wider range of variations, it becomes more resilient to changes in lighting conditions, viewpoints, or other factors that might affect the appearance of objects in images.**

**Two other transformations that can be done using ImageDataGenerator:**

**First, Random Zoom:**

**The zoom\_range parameter in ImageDataGenerator allows us to randomly zoom in or out on the images. It takes a floating-point value or a tuple of two floating-point values as input. For example, zoom\_range=0.2 means the images can be zoomed in or out by up to 20%. This transformation can help the model learn to recognize objects at different scales and can simulate the effect of objects being closer or farther away.**

**Second, Horizontal Flip:**

**The horizontal\_flip parameter in ImageDataGenerator enables random horizontal flipping of the images. By setting horizontal\_flip=True, each image has a 50% chance of being flipped horizontally. This transformation can be useful when the orientation of objects in the images doesn't affect their classification. For example, if we are training a model to classify cats and dogs, flipping the images horizontally will not change the label since cats and dogs remain cats and dogs regardless of their orientation.**

**QUESTION 3.**

**The array returned by the model. predict call represents the predicted probabilities for each class in the classification problem.**

**The array [[0.231701, 0.6328776, 0.01869423, 0.0764939, 0.04023323]] corresponds to a single prediction for a given input image. In this case, there are five classes, so each element in the array represents the predicted probability for one of the classes.**

**The first number, 0.231701, represents the predicted probability for the class labeled as 0 (daisy).**

**The second number, 0.6328776, represents the predicted probability for the class labeled as 1 (dandelion).**

**The third number, 0.01869423, represents the predicted probability for the class labeled as 2 (roses).**

**The fourth number, 0.0764939, represents the predicted probability for the class labeled as 3 (sunflowers).**

**The fifth number, 0.04023323, represents the predicted probability for the class labeled as 4 (tulips).**

**These probabilities indicate the model's confidence in assigning each input image to a specific class. The class with the highest probability is typically considered as the predicted class for the given input.**

**QUESTION 4.**

**np. argmax is a NumPy function that returns the index of the maximum value in an array along a specified axis.**

**np.argmax(result) is called to find the index of the maximum value in the result array. The result array is the output of the model's prediction, which contains the predicted probabilities for each class. By finding the index of the maximum probability, we can determine the predicted class.**

**The last line return (dict[themax], result[0][themax], themax) returns a tuple with three values:**

**dict[themax] returns the class label corresponding to the predicted class index (themax). The dict dictionary maps the class index to its label.**

**result[0][themax] returns the probability associated with the predicted class. result[0] represents the first row of the result array (as there is only one prediction), and result[0][themax] retrieves the probability value for the predicted class.**

**themax represents the index of the predicted class.**

**In summary, the last line of code returns the predicted class label, the associated probability, and the index of the predicted class for a given image.**