DataLab Preparation (Week 4, DataLab I, Monday)

**3. Image Modeling with Keras**

**3a How color images are stored when we load them in a numpy array?**

Color images are stored in 3-dimensional arrays (rank-3 tensor).

(no. of samples, height, width, color depth)

**3b How can you access the color of a pixel in an image that was loaded in a numpy array? Show a code example.**

To access a particular pixel in an image, we index both of the spatial dimensions of the image as such:

data[1000, 1500] - i.e., pixel at row 1000, column 1500 of the image array

**3c How can you set all the red values of the pixels in an image to zero in a numpy array?**

We can set all the red values to zero by changing the array data as such:

data[: , : , 3] = 0

plt.imshow(data)

plt.show()

**3d What is one-hot encoding? Describe an example using the image labels of your creative brief dataset.**

One-hot encoding is the process of transforming image data into an array that can be understood by the algorithm. In each row, the values are set to 0 except in the column corresponding to the class from which the image was taken.

For example, for images from my dataset, this would be the output of one-hot encoding them:

[1, 0, 0, 0, 0], <- lemon

[0, 1, 0, 0, 0], <-alcoholic beverage

[0, 0, 1, 0, 0], <-non-alcoholic beverage

[1, 0, 0, 0, 0], <-lemon

**3e How can we use a one-hot encoding array to determine how many predictions were correct in the output of a model?**

We can determine the number of correct predictions using one-hot encoding by taking the encoded array of labels in the test set with that in the predicted set, multiplying them, and then calculating the sum of the product. This will give us the number of correct classifications.

**3f Explain in your own words what is a Convolution layer.**

A convolution layer is similar to a dense layer, but it connects to the previous layer using a convolutional kernel, meaning that each image input is put through a convolutional operation. Another main difference is that a convolutional layer has one weight for each pixel in the kernel, meaning fewer weights in total.

**3g What is the purpose of a Flatten Layer after a Convolution layer?**

It serves as a connector between a convolutional layer and the following densely connected layers. The flatten layer takes the output of the convolutional layer and ‘flattens’ it into a one-dimensional array.

**3h What is defined by the parameter kernel\_size in the Conv2D function in Keras?**

The parameter kernel\_size indicates the dimensions of the convolutional kernel or filter (height and width).

**3i Explain in your own words what is zero padding in the convolution process.**

Zero padding consists of adding a border of 0s around the central image in order to have the output image size match the input image size after the convolutional operation of the kernel.

**3j Explain in your own words what is stride in the convolution process.**

In the convolutional process, the size of the step the kernel takes between input pixels is called the stride. The higher the stride size, the smaller the output image.

**3k How to use zero padding and stride in Keras?**

model.add(Conv2D(10, kernel\_size=3, activation='relu',

input\_shape=(img\_rows, img\_cols, 1),

padding = 'same, strides = 1))

**3l How to calculate the size of the output of a Convolution layer that uses zero padding and stride?**

output = (image size - kernel size + 2\* padding size)/ stride size + 1

**3m Explain what is dilated convolution and how to use it in Keras.**

Dilated convolution is a type of convolution where there is spaces between the kernel pixels, so that it covers more pixels of the input image, but it not all of them affect the output pixel.

How to use it in Keras:

model.add(Conv2D(10, kernel\_size=3, activation='relu',

input\_shape=(img\_rows, img\_cols, 1),

dilation\_rate = 2))

**3n When a CNN starts with sequence of two Convolution Layers, does the second layer operates directly on the input image? Explain your answer.**

No, the second layer does not operate directly on the input image. Instead, it operates on the feature map created by the first convolutional layer.

**3o Why having several Convolution Layers in a CNN is useful? Why kind of patterns can be identified by a sequence of Convolution Layers?**

Having several convolutional layers in a CNN is useful because it can detect more complex features the more layers it has. They allow the network to gradually build up representations of objects in the images from simple features to more complex features and up to sensitivity to distinct categories of objects. The layers in the early part of the network detect more simple features, such as lines or simple textures. Intermediate layers respond to more complex features that include simple objects, such as eyes. By the time the information travels up to higher layers of the network, the feature maps tend to extract specific types of objects. This allows the fully connected layers at the top of the network to extract useful information for object classification based on the responses of these layers.

**3p Explain how the number of parameters of a neural network that contains only Dense layers is calculated. Provide an example.**

Each unit of the first layer is connected to each one of the pixels of the image through a weight. The next layer's units are connected to all the units in the first layer, and so on. The parameters in a layer are calculated as every pixel in the image times the number of units in the layer plus the number of units in the layer for bias terms.

For the input layer:

model.add(Dense(10, activation = 'relu', input\_shape =(784,)))

- parameters = 784 \* 10 + 10 = 7850

For an intermediate layer:

model.add(Dense(10, activation = 'relu')

- parameters = 10 \* 10 + 10

For an output layer:

model.add(Dense(3, activation = 'softmax'))

- parameters = 3 \* 10 +3

**3q Explain how the number of parameters of a CNN (with Convolution Layers) is calculated. Provide an example.**

The number of parameters of a CNN are calculated similarily, only this time we multiply the number of units by the kernel size. For the first layer, we multiply the number of kernels by kernel size and add the number of kernels in the layer for bias terms. For intermediate layers, each unit is connected through a convolutional kernel to each feature map in the first layer. The flatten layer has no parameters.

For the input layer:

model.add(Conv2D(10, kernel\_size=3, activation='relu',

input\_shape=(img\_rows, img\_cols, 1)))

-parameters = 10 \* 9 (for the kernel size which is a 3x3 array) + 10

For intermediate layers:

model.add(Conv2D(10, kernel\_size=3, activation='relu', padding = 'same'))

- parameters = 10\* 9 \* 10 + 10;

Because there is zero padding here, the convolutions leaves the same number of pixels in each subsequent layer.

**3r What is the purpose of a pooling operation in a CNN?**

The purpose of a pooling operation in a CNN is to reduce the number of parameters.

**3s Explain in your own words how the max pooling operation works.**

Max pooling consists of extracting a small grid of pixels from the image, calculating its maximal value, and replacing the grid itself in the image with a singular pixel that has that maximal value. We end up with an image made up of a quarter of the original number of pixels, that retains only the brightest feature in each part of the image.

**3t How do we define the size of the window used by the MaxPooling2D layer in Keras?**

The size of the window used by the MaxPooling2D layer is defined by the 'pool\_size' parameter. This parameter specifies the height and width of the pooling window.

**3u Explain how to store the best parameters (weights) of a CNN before the network starts over-fitting.**

We store the best parameters using the callback ModelCheckpoint, which stores the weight of the network at the end of each epoch. The callback monitors the validation loss, and will only overwrite the weights whenever the validation loss declines. This means that if the network overfits, the weights will be stored for the epoch at which the validation loss was the smallest, before it started rising back up.

**3v Explain what is Dropout in the context of CNNs. How to use Dropout in Keras?**

Dropout is a regularisation strategy to help overcome the problem of overfitting. In each step of learning, we choose a random subset of the units in a layer and we ignore it. This group of units would be ignored both on the forward pass through the network, as well as in the back-propagation stage.

Dropout in Keras:

model.add(Dropout(0.2))

**3x Explain what is Batch Normalization in the context of CNNs. How to use Batch Normalization in Keras?**

Batch Normalization takes the output a layer and rescales it so that it always has a mean of 0 and a standard deviation of 1 in every batch of training.

Batch Normalization in Keras:

model.add(BatchNormalization())

**3z Why we should not combine Batch Normalization and Dropout?**

Dropout slows down learning, making it more incremental and careful, while batch normalization tends to speed up the process of learning. Therefore, their effects together may in fact counter each other.