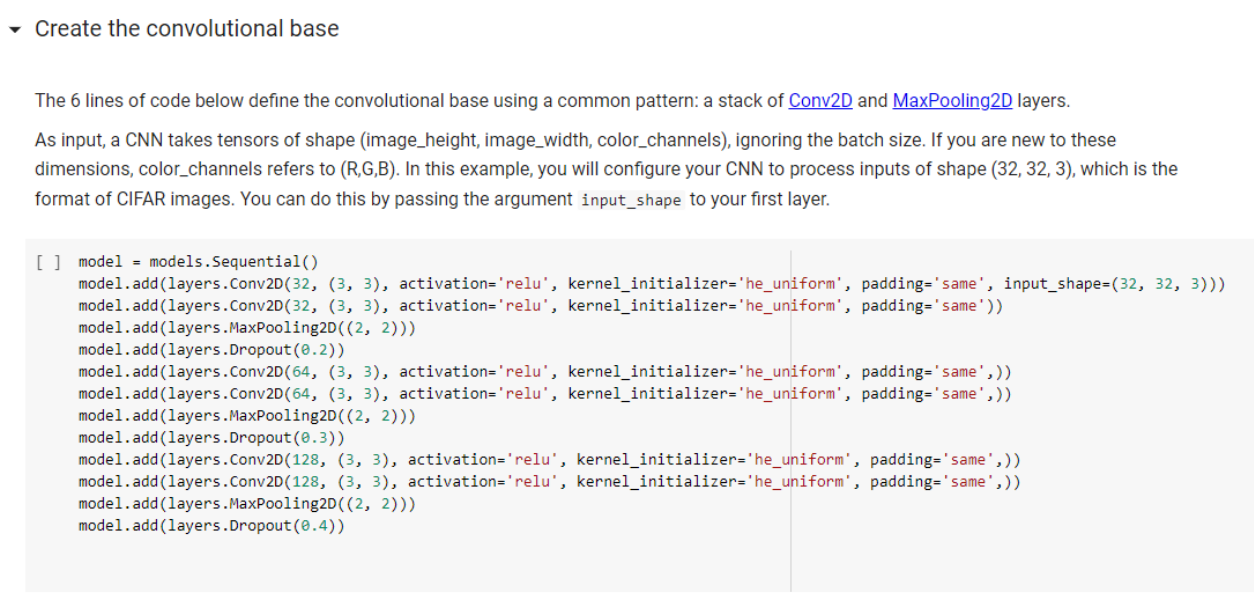
For this lab, we utilize a base neural network code that is default at a 70% validation accuracy. The goal of this lab was to attempt to get the validation accuracy to 90%. In this documentation, I will explain some basic concepts that will be needed to understand and run the code yourself through Google Colab. I will not be going into neural networks in detail as that should be researched before attempting this lab or else the basic concepts I discuss will not make any sense.

Baseline Model + Increasing Dropout


▾ Create the convolutional base

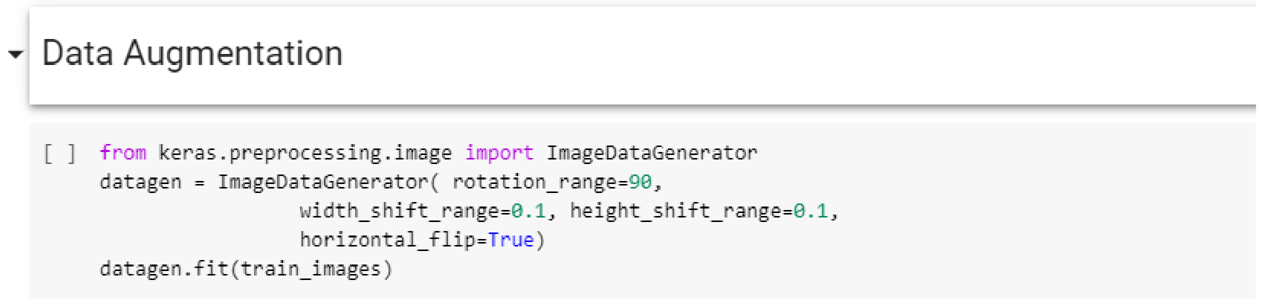
The 6 lines of code below define the convolutional base using a common pattern: a stack of Conv2D and MaxPooling2D layers.

As input, a CNN takes tensors of shape (image_height, image_width, color_channels), ignoring the batch size. If you are new to these dimensions, color_channels refers to (R,G,B). In this example, you will configure your CNN to process inputs of shape (32, 32, 3), which is the format of CIFAR images. You can do this by passing the argument `input_shape` to your first layer.

```
[ ]  model = models.Sequential()
     model.add(layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
     model.add(layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dropout(0.2))
     model.add(layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',))
     model.add(layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dropout(0.3))
     model.add(layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',))
     model.add(layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dropout(0.4))
```

In the above image, we can see that there are multiple layers called convolutional layers, pooling layers, and dropout layers. We adjust the dropout layer in order to try to adjust the end validation accuracy result. The parameter within the dropout layer is the percentage of features that are dropped when passing through neurons. So a dropout of 0.4 as seen on the last line of code would be a 40% drop of features passed through the neurons.

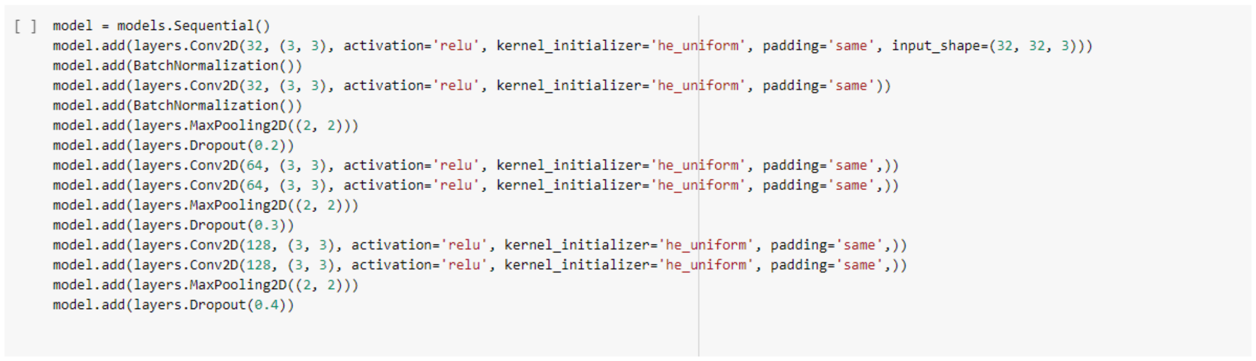
Baseline Model + Increasing Dropout + Data Augmentation


▾ Data Augmentation

```
[ ]  from keras.preprocessing.image import ImageDataGenerator
     datagen = ImageDataGenerator( rotation_range=90,
                      width_shift_range=0.1, height_shift_range=0.1,
                      horizontal_flip=True)
     datagen.fit(train_images)
```

The small piece of code that is seen above is how we apply data augmentation to our model. We can see that the data augmentation code resizes images by adjusting multiple factors such as width and height. This ultimately helps the model when training images.

Baseline Model + Increasing Dropout + Data Augmentation + Batch Normalization



```
[ ]  model = models.Sequential()
     model.add(layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
     model.add(BatchNormalization())
     model.add(layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
     model.add(BatchNormalization())
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dropout(0.2))
     model.add(layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',))
     model.add(layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dropout(0.3))
     model.add(layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',))
     model.add(layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dropout(0.4))
```

We can see in the image above that there are 2 lines of batch normalization code that were added compared to the initial Baseline Model + Increasing Dropout code. This was done in order to attempt to lower the number of epochs that is required for the training. The epochs are lowered because batch normalization helps to stabilize the learning process.