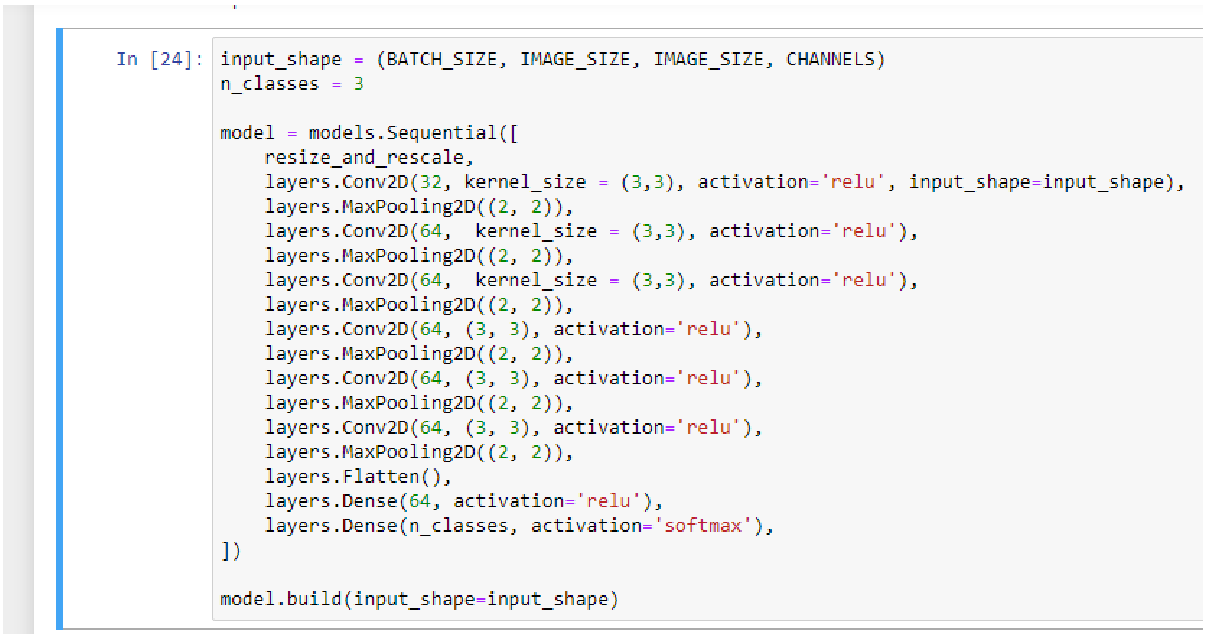
4.4.1 CNN Codes and Summary of the Model

Figure 4.9 codes define a Convolutional Neural Network (CNN) model. This model is composed of several layers. The first layer is a resize and rescale layer, which resizes the input images to a standard size (in this case, IMAGE_SIZE) and rescales them to a range of 0-1 using a max scaler. This ensures that all images have the same size and are properly normalized. The next layers are convolution layers, which are used to detect features in the images. These layers use a 3×3 filter window and a ReLU activation function to detect features in the images. Each convolution layer is followed by a max pooling layer, which reduces the size of the feature maps. This helps reduce the amount of computation required for the model and helps to reduce overfitting. The model then uses a Flatten layer to flatten the feature maps, followed by two dense layers. The first dense layer has 64 neurons and uses a ReLU activation function, while the second dense layer has neurons and uses a SoftMax activation function. The SoftMax activation function is used to classify the images into one of the three classes (under weld, good weld, and over weld).



```
In [24]: input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
         n_classes = 3

         model = models.Sequential([
             resize_and_rescale,
             layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64, (3, 3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64, (3, 3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64, (3, 3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Flatten(),
             layers.Dense(64, activation='relu'),
             layers.Dense(n_classes, activation='softmax'),
         ])

         model.build(input_shape=input_shape)
```

Figure 4.9 CNN model

Model Summary:

Table 4.1 Model summary after applying CNN model.

Layer Type	Output shape	Parameters
Sequential	(300, 300)	0
Conv 1	(298, 298)	896
Max pooling	(149,149)	0
Conv 2	(147,147)	18496
Max pooling	(73,73)	0
Conv 3	(71, 71)	36928
Max pooling	(35, 35)	0
Conv 4	(33, 33)	36928
Max pooling	(16, 16)	0
Conv 5	(14, 14)	36928
Max pooling	(7, 7)	0
Conv 6	(5, 5)	36928
Max pooling	(2, 2)	0
Flatten	256	0
Dense 1	64	16448
Dense 2	3	195
Total parameters = 183,747 Trainable parameters = 183,747 Non trainable parameters = 0		

4.5 Data Augmentation

Data augmentation is an approach employed to expand the size of an existing training dataset by manipulating the existing samples in the dataset. There exist numerous ways this can be achieved, including cropping, rotating, shifting, shearing, and flipping. This strategy can be extremely beneficial for deep learning models that may suffer from underfitting due to a lack of data. Many computers vision-related tasks, including object detection, segmentation, and classification, overly rely on data augmentation to maximize the accuracy of their corresponding models. As shown in Figure 4.10 represent the augmented images created from the original images.

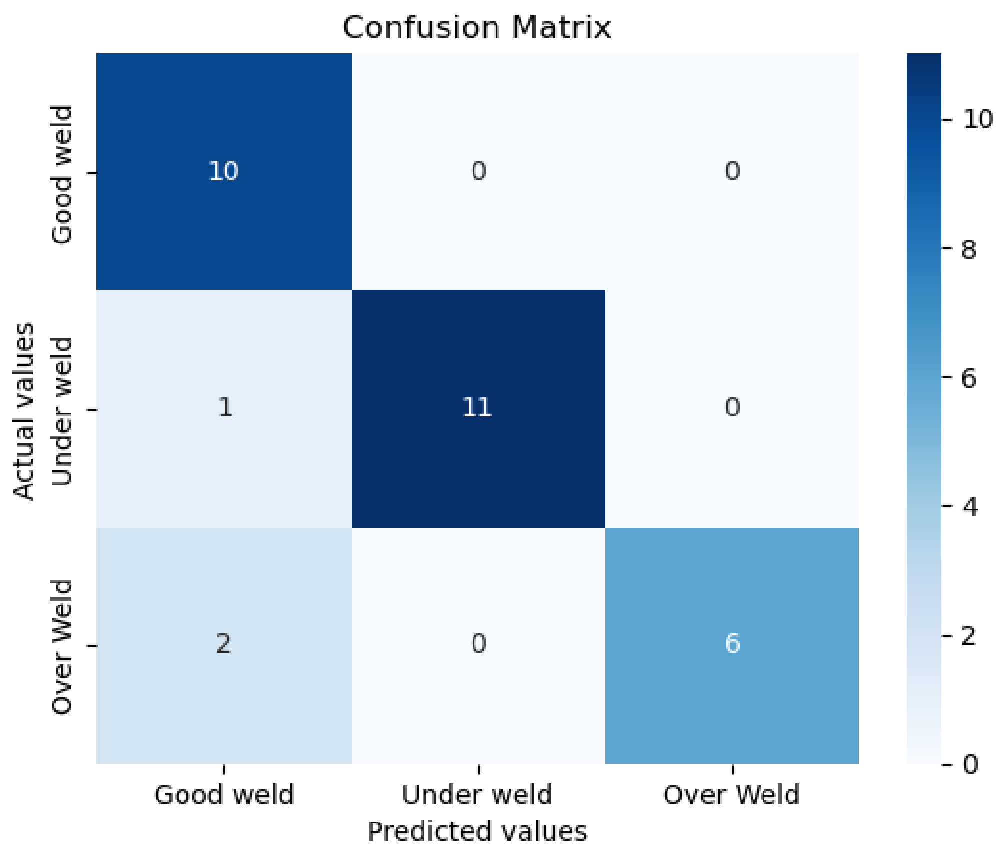


Figure 5.3 Confusion matrix of the test dataset