1. **Identity Block:**

```
Result = [0.         0.         1.2319232  0.         0.04842055
0.39851195 1.703964   0.32821387]
```

2. **Convolution Block:**

```
Result = [0.        1.3407363 1.4184607 0.        0.        2.2878466
0.8547394 0.5266884]
```

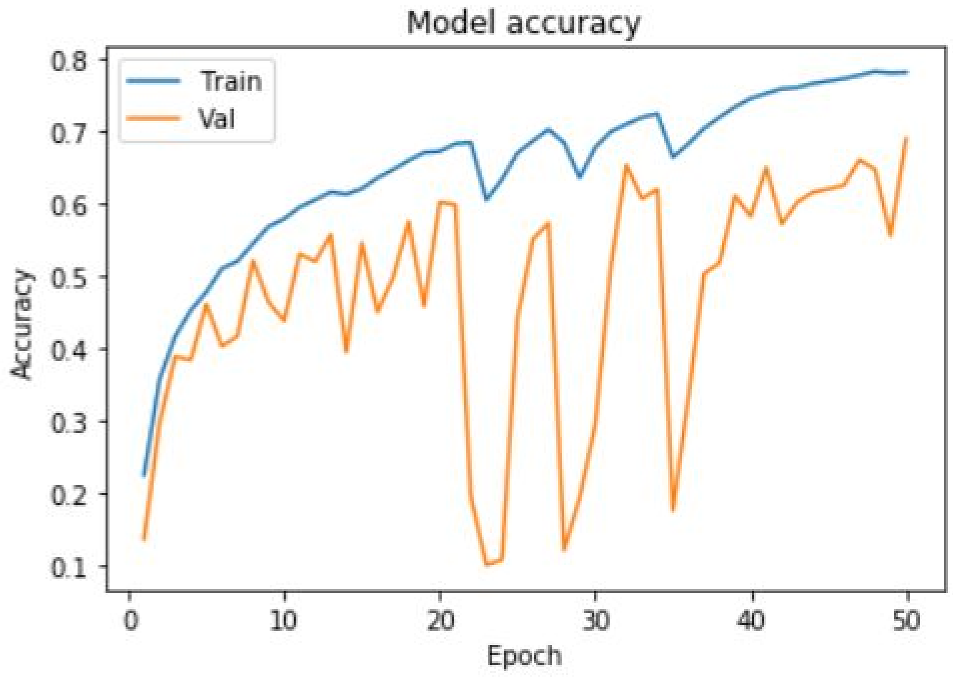
3. **ResNET Block tested on CIFAR10 Dataset:**

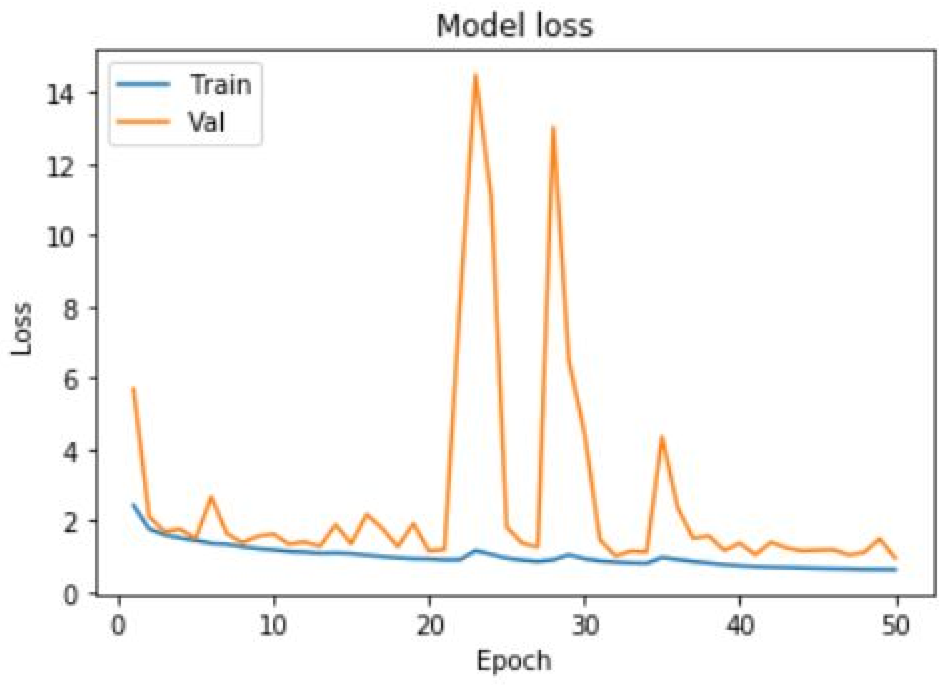
```
50000 train samples
10000 test samples

Epoch 1/50
25/25 [==============================] - 90s 4s/step - loss: 2.4100 - acc:
0.2254 - val_loss: 5.6821 - val_acc: 0.1358
                                   ...
Epoch 50/50
25/25 [==============================] - 44s 2s/step - loss: 0.6151 - acc:
0.7810 - val_loss: 0.9365 - val_acc: 0.6887

10000/10000 [==============================] - 10s 991us/step
Test loss: 0.9365417770385742
Test accuracy: 0.6887
```

4. **Plotting the Learning Curve:**





5. Analysis:

Observing the training plots from our model, overfitting is visible in both graphs. Between epochs 21-24, 28-31, and in 35, the validation error vastly differs from the training error. Since the validation accuracy in these epochs falls to around 10%-20%, we know our model does not generalize well. In these same epochs, the model fits the training data well enough to have accuracy above 50%. Therefore our model overfits our training data, and fails to generalize to test data well.

There are a few different ways to prevent overfitting in Neural Net Architectures. Since our model trains itself over 50 epochs and converges to a final training error, one way to prevent our model from overfitting would be to reduce the number of epochs. Doing this, more training images would pass through our neural net each epoch, and our model would have less iterations to fit training images as well. Another way to help prevent overfitting our data would be to add a dropout layer to our convolutional neural network. Dropout layers prevent overfitting by dropping units based on a probability, so the model cannot depend too much on any particular feature.