

---

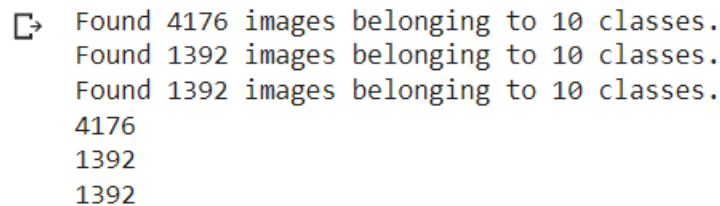
# Implement Convolutional Neural Networks - PA5

---

Gopi, Jagan Mohan Reddy, Pavan Kuamr  
{sthota2,jaganmoh,ptammine}@buffalo.edu

## 1 Dataset

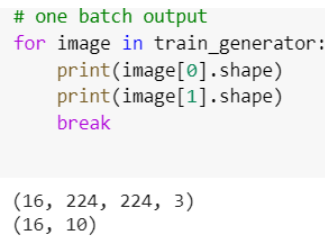
- Imported the necessary libraries from various packages like keras, tensorflow, matplotlib and numpy.
- Set the seed value to the tensorflow to reproduce the same result.
- With the given image height(224), width(224) and batch size (16), the images were read from the different folders like train, validate and test using ImageDataGenerator() function and stored in the respective generator variables. Finally, we displayed the number of samples of train, validation, and test is shown in Figure 1.



```
Found 4176 images belonging to 10 classes.  
Found 1392 images belonging to 10 classes.  
Found 1392 images belonging to 10 classes.  
4176  
1392  
1392
```

Figure 1: Number of samples of each folder

- **Understanding the nature of samples in each folder, using ImageDataGenerator():** All the samples in each folder, partitioned based on the batch size(16). We also display each sample distribution and the class label.



```
# one batch output  
for image in train_generator:  
    print(image[0].shape)  
    print(image[1].shape)  
    break  
  
(16, 224, 224, 3)  
(16, 10)
```

Figure 2: Number of samples of each folder

From the above Figure 2, one batch has 16 samples, height & width of an image is 224 and the last value is indicating the type of image (RGB).

## 2 Build your Neural Network and Train

We have built a Convolutional Neural Network (CNN) with 3 hidden layers without regularization methods, which includes Conv2D layer, “ReLU” as activation function and the output layer’s

Table 1: Distributiun of batch sizes among type of data

Type of data	No_of_input_images	Batch_size	No_batches
Train	4176	16	$4176/16 = 261$
Validation	1392	16	$1392/16 = 87$
Test	1392	16	$1392/16 = 87$

activation function used is “Softmax” as this problem deals with multi-class classification. The layer configuration and its description is given below.

#### Base Model

- Convolution Layer 1: The hyperparameters:- Number of filters: 96, kernel size is 3x3, input shape = 224, default stride is (1,1). The activation function “ReLU” and the pooling is max with 2x2 matrix.
- Convolution Layer 2: The hyperparameters:- Number of filters: 256, kernel size is 3x3, default stride is (1,1). The activation function “ReLU” and the pooling is max with 2x2 matrix.
- Convolution Layer 2: The hyperparameters:- Number of filters: 512, kernel size is 3x3, default stride is (1,1). The activation function “ReLU” and the pooling is max with 2x2 matrix.
- Fully Connected Dense layer: 1024 neurons and the activation function “ReLU” is used.
- Output Layer: The output layer mapped to 10 classes from the FC layer with activation function as “softmax”.

#### Model Hyperparameters used:

- optimizer: Adam
- loss: categorical\_crossentropy (as it is a multi-class classification problem)
- metric: accuracy

The training process including loss and accuracy show in Figure 3.

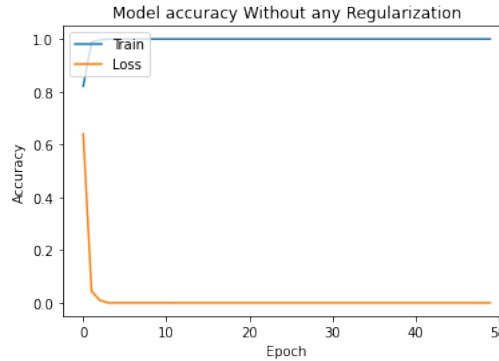


Figure 3: Base Model with Train accuracy vs loss

#### Part 2.3 - L1 Regularization

- Convolution Layer 1: The hyperparameters:- Number of filters: 96, kernel size is 3x3, input shape = 224, default stride is (1,1) and the kernel  $l_1$ -regularization is 0.001. The activation function “ReLU” and the pooling is max with 2x2 matrix.
- Convolution Layer 2: The hyperparameters:- Number of filters: 256, kernel size is 3x3, default stride is (1,1). The activation function “ReLU” and the pooling is max with 2x2 matrix.

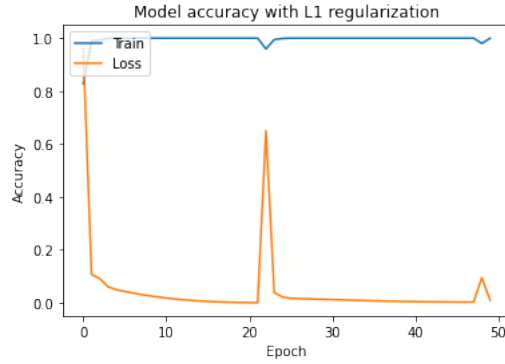


Figure 4:  $l_1$ -Regularization Model with Train accuracy vs loss

- Convolution Layer 3: The hyperparameters:- Number of filters: 512, kernel size is 3x3, default stride is (1,1). The activation function “ReLU”, the pooling is max with 2x2 matrix and then flattened the layers.
- Fully Connected Dense layer: 1024 neurons and the activation function “ReLU” is used.
- Output Layer: The output layer mapped to 10 classes from the FC layer with activation function as “softmax”.

#### Model Hyperparameters used:

- optimizer: adam
- loss: categorical\_crossentropy (as it is a multi-class classification problem)
- metric: accuracy

The training process including loss and accuracy show in Figure 4.

#### Part 2.4 - L2 Regularization

- Convolution Layer 1: The hyperparameters:- Number of filters: 96, kernel size is 3x3, input shape = 224, default stride is (1,1) and the kernel  $l_2$ -regularization is 0.001. The activation function “ReLU” and the pooling is max with 2x2 matrix.
- Convolution Layer 2: The hyperparameters:- Number of filters: 256, kernel size is 3x3, default stride is (1,1). The activation function “ReLU” and the pooling is max with 2x2 matrix.
- Convolution Layer 3: The hyperparameters:- Number of filters: 512, kernel size is 3x3, default stride is (1,1). The activation function “ReLU”, the pooling is max with 2x2 matrix and then flattened the layers.

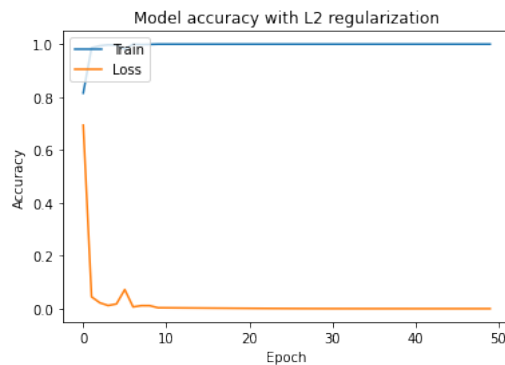


Figure 5:  $l_2$ -Regularization Model with Train accuracy vs loss

- Fully Connected Dense layer: 1024 neurons and the activation function “ReLU” is used.
- Output Layer: The output layer mapped to 10 classes from the FC layer with activation function as “softmax”.

#### Model Hyperparameters used:

- optimizer: adam
- loss: categorical\_crossentropy (as it is a multi-class classification problem)
- metric: accuracy

The training process including loss and accuracy show in Figure 5.

#### Part 2.5 - Dropout Regularization

- Convolution Layer 1: The hyperparameters:- Number of filters: 96, kernel size is 3x3, input shape = 224, default stride is (1,1). The activation function “ReLU” and the pooling is max with 2x2 matrix.
- Convolution Layer 2: The hyperparameters:- Number of filters: 256, kernel size is 3x3, default stride is (1,1). The activation function “ReLU” and the pooling is max with 2x2 matrix.
- Convolution Layer 3: The hyperparameters:- Number of filters: 512, kernel size is 3x3, default stride is (1,1). The activation function “ReLU” and the pooling is max with 2x2 matrix. Flatten the network and Dropout with 0.2 probability.
- Fully Connected Dense layer: 1024 neurons and the activation function “ReLU” is used.
- Output Layer: The output layer mapped to 10 classes from the FC layer with activation function as “softmax”.

#### Model Hyperparameters used:

- optimizer: adam
- loss: categorical\_crossentropy (as it is a multi-class classification problem)
- metric: accuracy

The training process including loss and accuracy show in Figure 6.

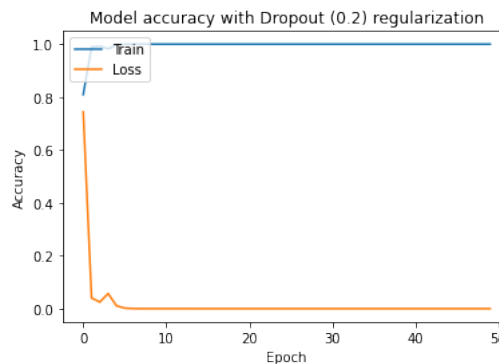


Figure 6: Dropout-Regularization Model with Train accuracy vs loss

### 3 Part 3 - ResNet-50

- As part of this section, we used the ResNet50 model with different hyperparameters to train, validate and test the given dataset. The weights are chosen from “imagenet”, input shape is “224” and pooling is “max” for ReseNet50.
- The first 103 layers were frozen in ResNet50 model.

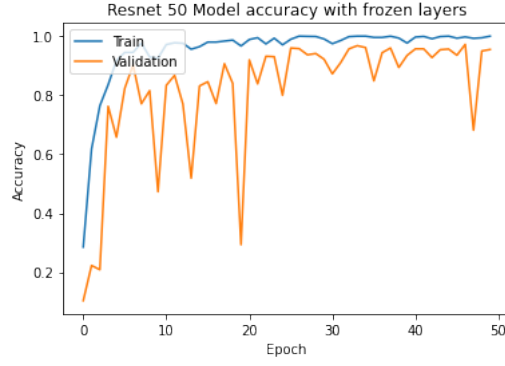


Figure 7: resNet50 Model with Train accuracy vs loss

- Flatten the model to construct the Neural network and the output layer map to 10 classes.

The training process including loss and accuracy show in Figure 7.

### 3.1 Comparison of Models

Table 2: Comparison of Different Regularization methods

Model Name	optimizer	Regularization Method	Train Accuracy	Train Loss	Test Accuracy	Test Loss
Model without Regularization (Vanilla)	adam	None	0.9961	0.0139	0.8384	1.7715
Model with L1 regularization	adam	L1(0.001)	0.9948	0.0497	0.8484	1.5626
Model with L2 regularization	adam	L2(0.001)	0.9953	0.0185	0.8642	1.7693
Model with Dropout regularization	adam	Dropout(0.2)	0.9954	0.0176	0.8484	1.9266
resNet50	None	None	0.9533	0.3407	0.9561	0.6130