# Object Classification by Deep CNNs



## Overview

The goal of this assignment is to build a visual recognition system that leverages deep Convolutional Neural Networks (CNNs) for object classification.

You will work with CIFAR10 dataset [1] which consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. CIFAR10 is one of the most commonly used datasets in the computer vision community that was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton in 2009. It is still very popular and actively used by researchers to report and compare the performance of various supervised and unsupervised methods.

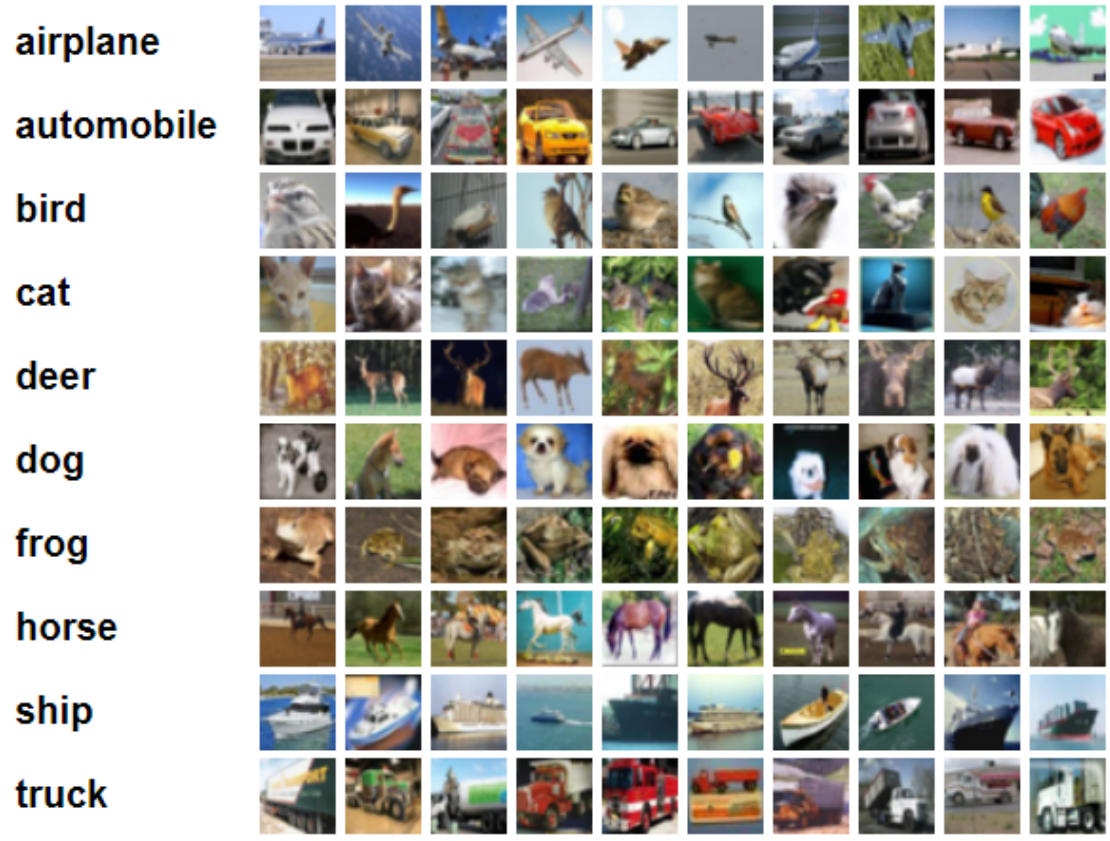


Figure 1: The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

In this homework, you will implement and train a Residual Network (ResNet) from scratch in Tensorflow/Keras by using Google Colab infrastructures (e.g. GPUs). A starter code in Google Colab accompanies this homework. It is highly recommended that you follow the instructions in this document.

[1] https://www.cs.toronto.edu/ kriz/cifar.html/

## Deep Residual Learning

Very deep networks are useful since they have enough abstractions to represent highly nonlinear and complex functions. However, increasing the depth of a network comes with the cost of possibly vanishing or exploding the gradients during back-propagation. Simply put, the gradient from the final layers of a very deep architecture may exponentially decrease to a very small value or exponentially increase to very large values and explode. As a result, this can impede the process of training. Residual Networks, or in short ResNets, overcome this issue by using a shortcut that allows the gradient to be directly backpropagated to earlier layers. Figure 2 shows the concept of residual learning.

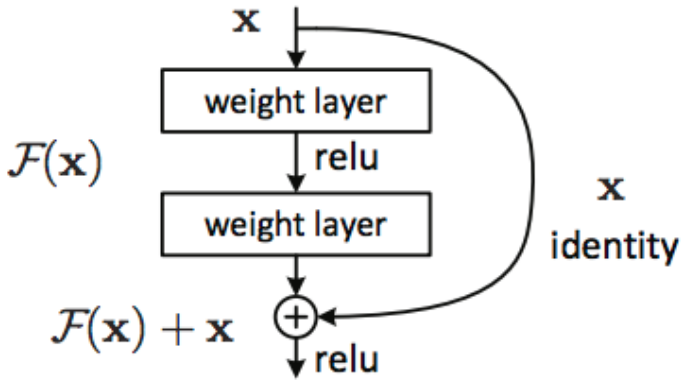


Figure 2: In a residual block, the input is added to the output of the block before passing into the final acitvation.

ResNet utilizes two basic blocks, with skip connections, that are repeated in their architecture. These blocks are called the identity block and the convolutional block, and one of the main goals of this homework is to implement them in Tensorflow/Keras. We will discuss each of them in details as follows.

### Identiy Block

In an identity block, as shown in Figure 3, the original input to the block is simply added to the final output of the block.

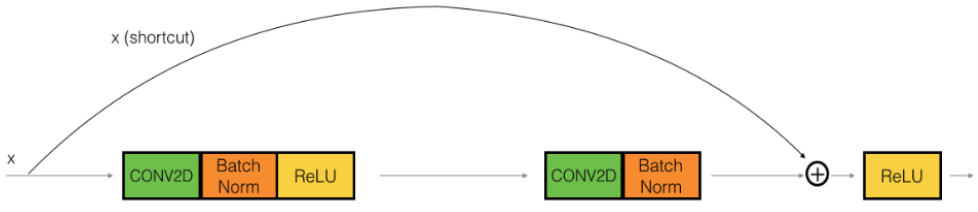


Figure 3: Simple identity block.

The upper portion is simply a shortcut wheres the main path consists of standard operations such as convolutions ( as denoted by CONV2D) and activation functions (e.g. ReLu). Batch normalization is also added to speed up the training process.

Although the block shown in Figure 3 is a standard representation of identify block, in this homework, we use a slightly modified version of it in order to make our ResNet more powerful. As such, we add one more convolution followed by batch normalization and ReLU activation function in the main path. Figure 4 shows the identity block used in this homework. In addition, details of identity block (e.g. number of filters, strides, etc) have been presented in the starter code (Google Colab document).

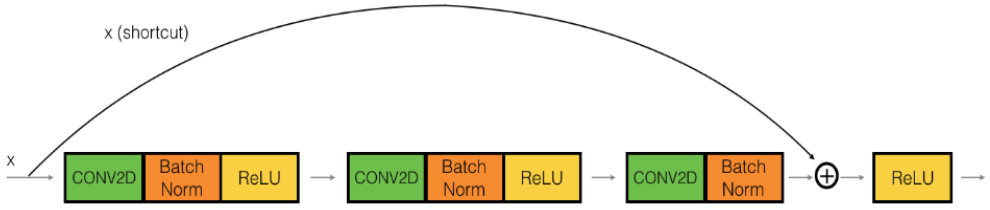


Figure 4: The identity block used in this homework.

## Convolutional Block

Since adding tensors is one of the most commonly used operations in a ResNet, it is good to remember that two tensors must have the same size across all dimensions to be added together. In ResNet, a convolutional block is used when the dimentions of input and output to the block are not the same. As shown in Figure 5, a convolutional Block is very similar to the identity block that was demonstrated in Figure 4, except for the fact that the shortcut path comprises of a convolutional layer followed by a batch norm. This allows to resize the input to the block to a different dimension. Note that we don't employ any activation function such as ReLu in the shortcut path since our goal is to simply resize the input. Details of convolutional block (e.g. number of filers,stride etc.) have been presented in the starter code (Google Colab document).

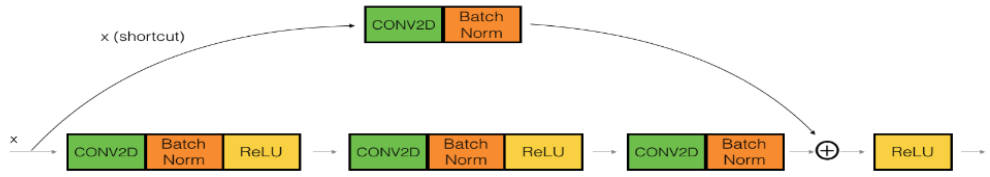


Figure 5: The convolutional block. It can be used when dimensions of input and output of the block are not the same.

## ResNet Architecture

Now that we have learned about the building blocks of a ResNet architecture, we further learn how to build the ResNet and implement it. Figure 6 shows the ResNet model that we will implement in

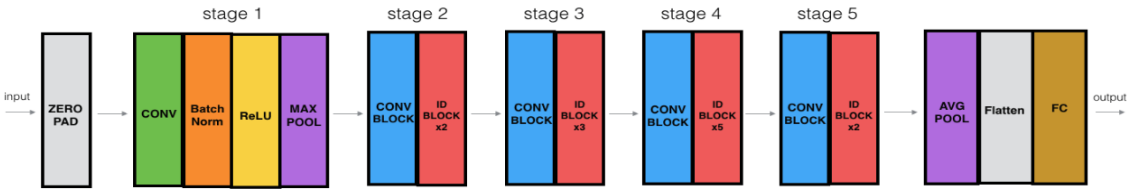


Figure 6: ResNet architecture.

this homework. It comprises of five different stages, excluding the last layer. From stage 2 to stage 5, covolutional and identity blocks (as denoted ID) are used interchangeably. The last layer comprises of an averaging pooling layer,followed by a flattening operation and a final fully connected layer to make predictions. Since this is a multi-class classification problem, we use a softmax function as the last activation layer. Exact details of the size of the feature maps as well as the number of filters that are used in each stage have been detailed in the starter code ( Google Colab document).

## Dataset

In this homework, we will used the CIFAR10 dataset [2]. CIFAR10 is a labeled subset of 80 million tiny images dataset and it contains 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We used Keras built-in functionalities to download and feed this dataset into our CNN. In addition, we also use data augmentation to improve the performance of training. All the relevant codes have been presented in the starter code and can be readily used.

## Tasks

1. Implement the identity block shown in Figure 4. To test your implementation, in the starter-code, we provide a tensor as an input and use predictable random initialization method for the convolutional kernels. Your output numbers should exactly match the desired output. Report the generated output numbers as the answer to this question (**20 points**)

2. Implement the convolutional block shown in Figure 5. Similar to part 1, your output numbers should exactly match the desired output. Report the generated output numbers as the answer to this question (**20 points**)

3. Implement and train the ResNet architecture shown in Figure 6 on CIFAR10 dataset for **50 epochs**. Report the accuracy of your model on CIFAR10 **test set** as the answer to this question. (**30 points**)

4. Plot the accuracy of training and validation over the entire 50 epochs on the same plot using the provided helper function for plotting. (**5 points**)

[2] https://www.cs.toronto.edu/ kriz/cifar.html/

5. Considering the overall trend of training and validation accuracy, is your model over-fitting or under-fitting? explain the reasoning behind your answer and identify one possible solution to overcome the issue (**15 points**)

## Submission

Your submission will consist of a single tarball, "*UID*.tar.gz" where *UID* is the university ID of the submitter. It will be submitted on CCLE. Your tarball will consist of several files, listed below. Please respect the filenames and formatting **exactly**. Your tarball should include:

- README: a .txt file
  - Line 1: Full name, UID, email address of first group member (comma separated)
  - Line 2: Full name, UID, email address of second group member (comma separated) if any, empty otherwise
  - Use the rest of the file to list the sources you used to complete this project
- code/: a directory containing all the .py files that you used for your project. You can save your Google Colab as a .py file for this purpose and your code should run without any issues.
- PDF/: You should include a PDF file that contains the answers to all the questions listed under Tasks (e.g. numbers,plot etc.)
- The homework is due on Monday December 9, 11:59 PM PST. This is a **hard** deadline. No late submissions will be accepted.

## Important Points to Remember

- Upload the starter code (starter.ipynb) to your Google Colab[3]
- Click on Runtime and then Change runtime type to set the hardware accelerator to **GPU**.
- You can run the entire code blocks by clicking on Runtime and then Run all (or simply by Ctrl+F9)
- If during training, your session disconnects, in most cases you should be able to simply reconnect and restart the training where you left off ( unless the session has expired)
- Follow the suggested Conv2D parameters as indicated in the starter code, or your output numbers may differ from desired values.
- Don't overwhelm your Google Colab quota by running many instances of training at the same time. It is recommended that you have at most two concurrent training sessions that utilize GPUs.

[3] https://colab.research.google.com/

- Follow the recommended settings and hyperparameters provided in the starter code. Our tests show that your training (50 epochs with batch size of 2048) should take roughly around 25 minutes.
- For the ResNet model, if your implementations is correct, it should offer an accuracy of at least **60** percent for 50 epochs of training on CIFAR10 testset. We will grade you based on : (1) the correctness and logic of your implementation (2) reported accuracy.
- It is a fact that performance of deep CNNs is dependant on their initialization. Thus, it is recommended to train and evaluate the model more than once and use the results from the better performing model.
- You may ignore warning messages regarding new changes in Tensorflow 2.0 and deprecated usage of some functionalities.