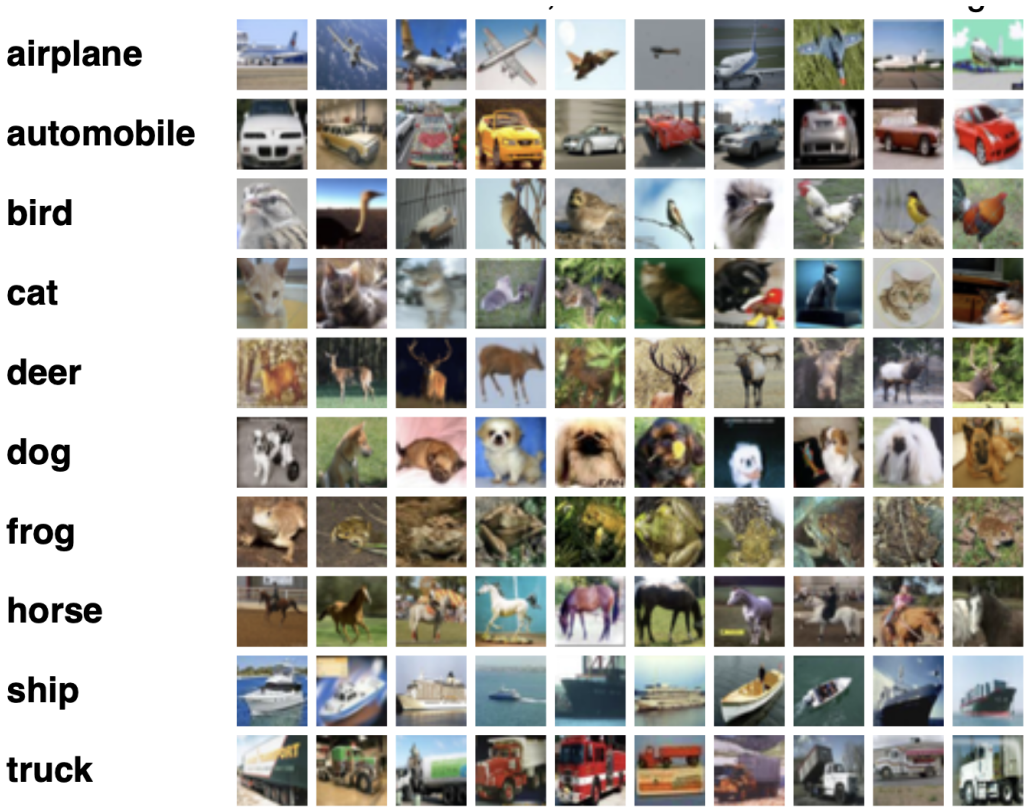
Machine Learning Project [25 points]

June 7, 2021

1 Problem Description

The goal of this project is to implement a convolutional neural network for classifying images in the CIFAR-10 dataset [1]. The CIFAR-10 dataset contains 60,000 color images with pixel dimensions 32×32. There are 50,000 training images and 10,000 test images. Shown below is a snapshot showing a random selection for the 10 different object classes.



Your main task is to implement a simple convolutional neural network that is loosely inspired by the AlexNet architecture that won the ImageNet competition in 2012 [2] Then, you will make several simple modifications to this network architecture to improve its performance. Note that you will NOT be working with ImageNet but CIFAR-10, which is a much smaller dataset, in order for you to be able to train the network in a timely manner. In particular, you

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

[2] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012) "Imagenet classification with deep convolutional neural networks", in Advances in Neural Information Processing Systems 25 (NIPS 2012), (pp. 1097-1105)

will be asked to first implement a convolutional neural network based on AlexNet and then create variants by introducing improvements to the basic architecture that will reduce over-fitting leading to a better performance. These "improvements" include Dropout, BatchNorm and Image Augmentation and are described in detail, as separate tasks, in Section 1.4.

1.1 Coding Tasks

Task-1 : Implementing a Convolutional Neural Network

The first step is to implement an AlexNet-variant that you will be using and modifying throughout this project. The architecture is as follows:

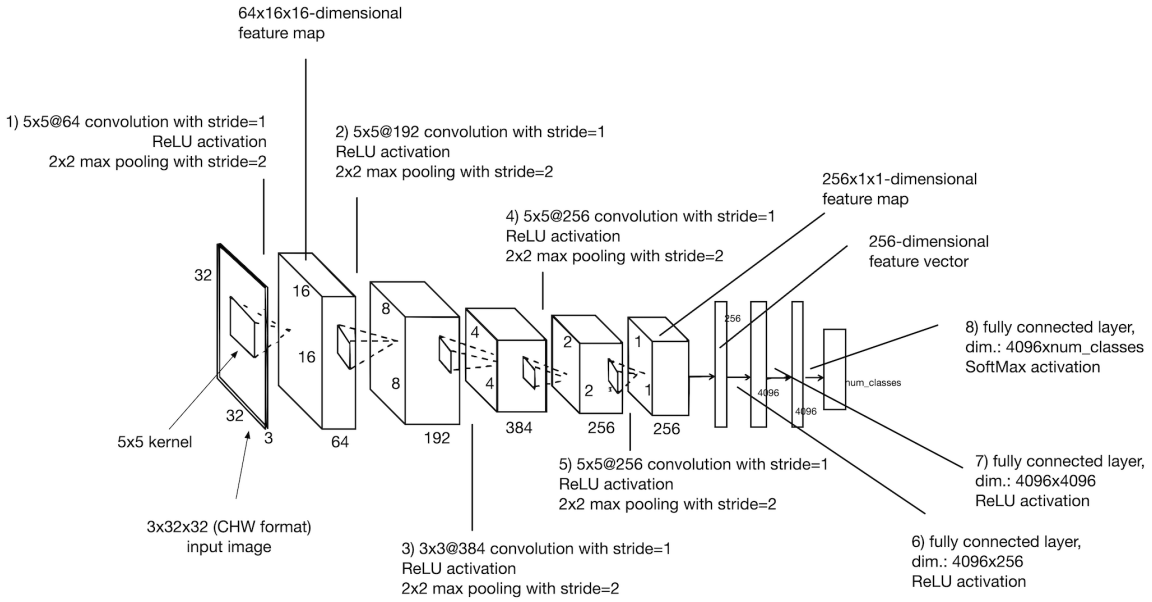


Figure 1: Convolutional Neural Network (AlexNet)

Note that there are some differences compared to the original AlexNet architecture. Overall though, there are 7 hidden layers in total: 5 convolutional layers and 2 fully-connected layers. There is one output layers mapping the last layer's activations to the classes. For this network,

- all hidden layers are connected via ReLU activation functions
- the output layer uses a softmax activation function
- make sure you return the logits and the softmax output; the logits are used for computing the cross-entropy loss (instead of passing the softmax outputs) for numerical stability reasons.

Task-2 : Adding Dropout

In this step, your task is to modify the first architecture by adding add dropout layers to reduce over-fitting. You can copy and paste the model of your architecture from above and

make the appropriate modifications. In particular,

- place a Dropout2d layer (this is also referred to as "spatial dropout"; will be explained in the lecture) before each max-pooling layer. You may give a try with a dropout probability $p = 0.2$, but you are invited to explore different values.
- place a regular dropout after each fully connected layer with probability $p = 0.5$, except for the last (output) layer.

The new architecture is depicted in figure 2 (changes, compared to the previous section, are highlighted in red):

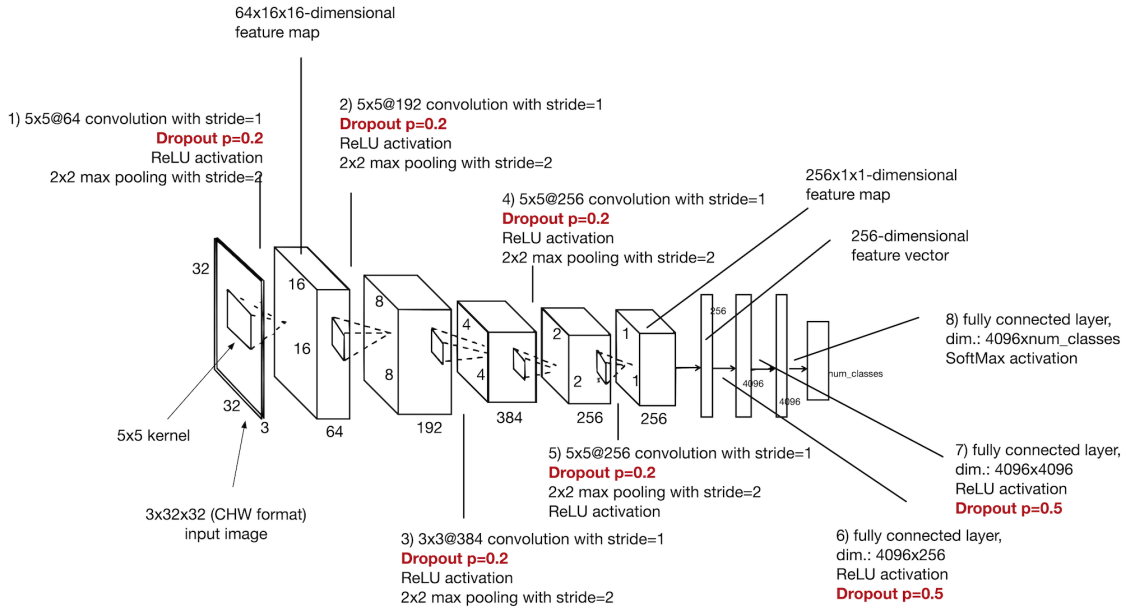


Figure 2: Modified AlexNet architecture by adding Dropout layers (task-2)

Task-3 : Adding BatchNorm

In this 3rd part, you are now going to add BatchNorm layers to further improve the performance of the network. This use BatchNorm2D for the convolutional layers and BatchNorm1D for the fully connected layers. The new architecture is depicted in figure 3 (changes, compared to the previous section, are highlighted in red):

Task-4 : Removing maxpooling layers

In this part, your task is to remove all maxpooling layers and replace the fully-connected layers by convolutional layers. Note that the number of elements of the activation tensors in the hidden layers should not change. I.e., when you remove the max-pooling layers, you need to increase the stride of the convolutional layers from 1 to 2 to achieve the same scaling. Furthermore, you can replace a fully-connected convolutional layer by a convolutional layer using stride=1 and a kernel with height and width equal to 1. The new architecture is depicted in figure 4 (changes, compared to the previous section, are highlighted in red):

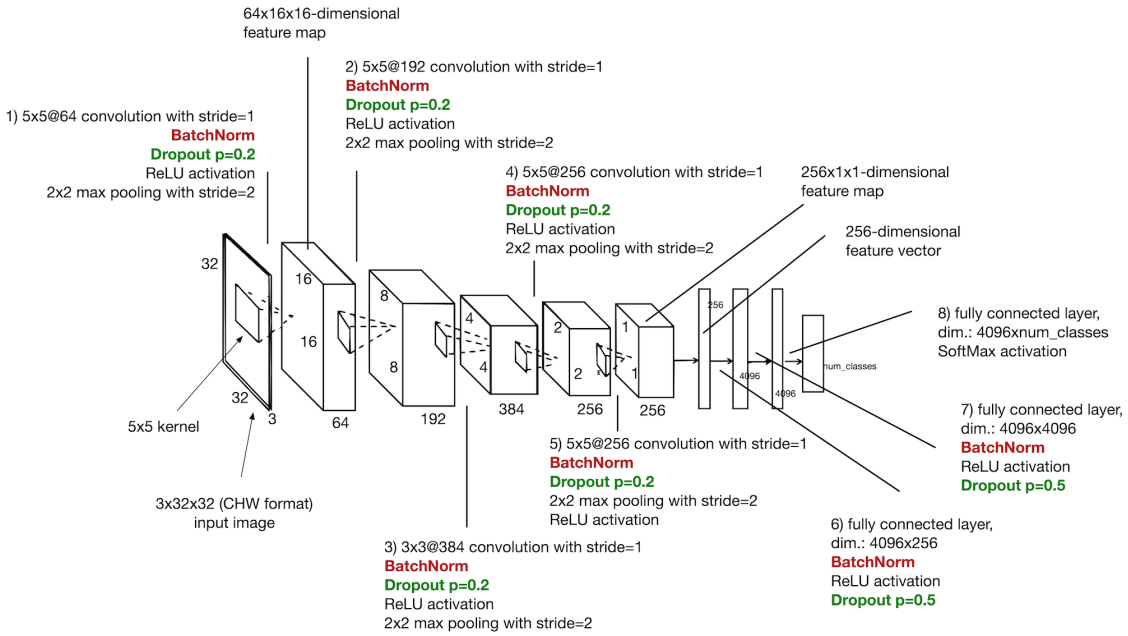


Figure 3: Modified AlexNet architecture by adding BatchNorm layers (task-3)

Task-5 : Add Image Augmentation

In this part, you should use the architecture from the previous task (i.e. task-4), but use additional image augmentation during training to improve the generalization performance.

1.2 Approach

Figure 5 highlights the approach that you should follow to address the problem. Although you can download a Python version of the dataset from the website `https://www.cs.toronto.edu/~kriz/cifar.html`, it is much easier to import it through tensorflow's dataset functions, i.e. :

```
(train_images, train_labels),(test_images, test_labels) =
    tensorflow.keras.datasets.cifar10.load_data()
```

The above training set consists of 50,000 figures. You should extract a subset of 10,000 figures from it create a separate validation set which you can use for hyper-parameter tuning. Note that this subset should not be part of your training set, anymore. In the pre-processing stage you should perform normalization of the data and potentially on-hot-encoding. The CNN architectures described in section 1.4 should be programmed in the model development stage. The most time consuming part of your project is expected to be the training stage, since at this stage you will have to perform extensive hyper-parameter tuning to optimize your clasifier. The main focus should be the dropout parameter p, but feel free to explore the influence of other parameters such as kernel size, learning rate etc, optimizer function etc. Note that the network training in each of the above implementation sections will take 5 min on a GPU, but on a CPU it will be much longer. Therefore, it is highly recommended

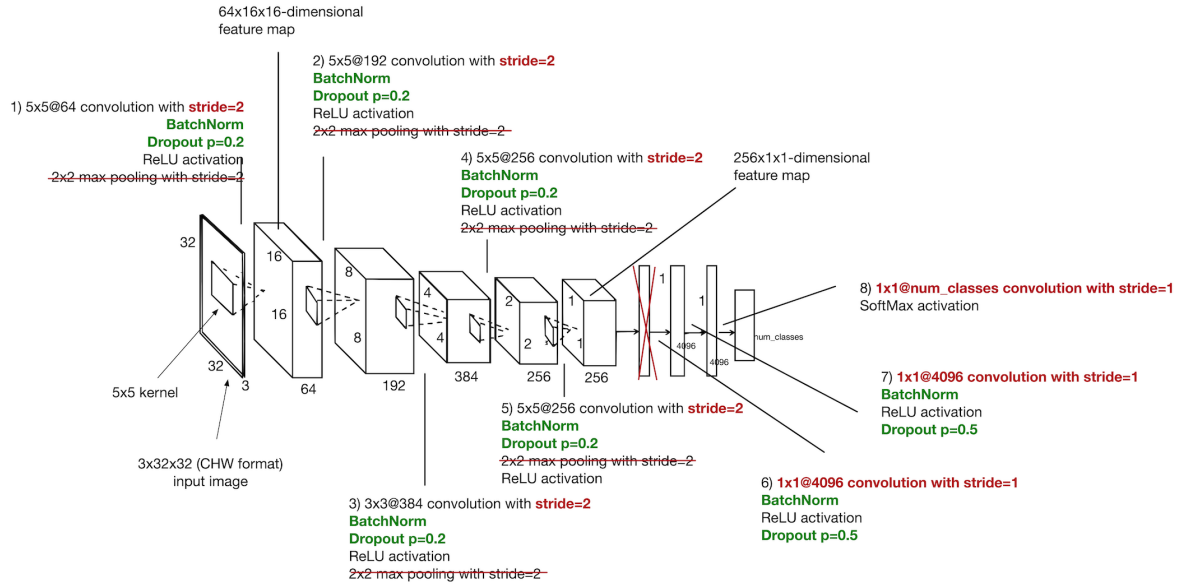


Figure 4: All convolutional approach by removing maxpooling (task-4)

to run this homework on a cloud server, e.g. Google Colab, which will allows free use of GPUs. At the performance evaluation stage you should use the test set to characterized the classification accuracy of the optimized models.

1.3 Milestones and Write Up

There is a number of milestones each project should meet:

- Implement the aforementioned CNN architectures and train them.
- Perform extensive optimization of their performance, through hyper-parameter tunning, to deal with bias-variance tradeoffs.
- Compare the aforementioned variants, after their optimization, and provide reasonable justification of their performance differences. Identify the maximum accuracy you can achieve.
- (optional) Develop any other method that can provide better accuracy results. This can be an addition on the AlexNet, or a totally different method. Feel free to take ideas from recently published literature.
- Write a dense report (i.e. 1000 words -coding parts excluded) that will summarize your experiments and findings.

1.4 Write Up

You are expected to describe and motivate your experiments in a format appropriate for scientific results. To do this, you will want to think carefully about the experiments you conduct: what conjecture are you testing? what is the purpose behind, say, conducting a

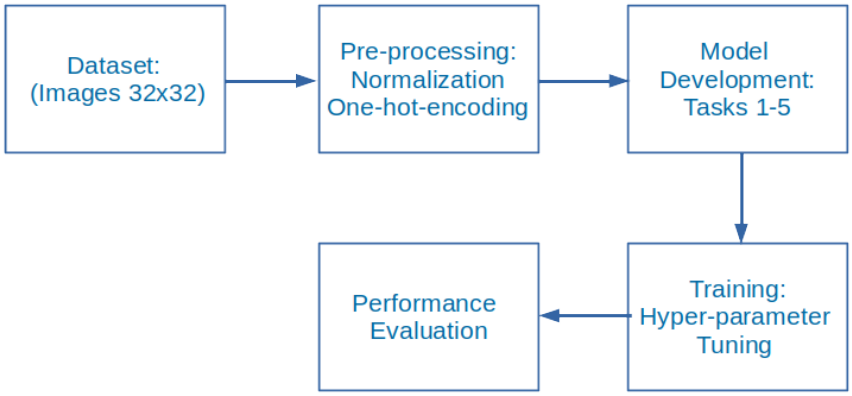


Figure 5: End-to-end machine learning process

parameter sweep, or chossing a particular learning method to explore? You will also need to keep records of your experiments, or be prepared to redo them periodically. You can also conduct additional experiments with data outside the CIFAR-10 collection, if you feel like it will be helpful to test your hypothesis. However, your final CIFAR-10 submission should be based on the distributed training data.

You should submit the code you are using in your experiments, as separate files (.py or jupyter notebook(s)), along with a draft of your final project report. The writeup is important - it will be read and graded. Note that this is not a *complete* report of everything you did - you should be writing a report that will help the reader reproduce your results. It should also give enough information for someone *not* in the class to understand what you did was justified and correct, what the goal was, and what if anything was learned in your attempts to accomplish that goal. The structure of the report should include:

- **Abstract.** Here, you should briefly (i.e. 3-4 lines) summarize your motivation, the problem you are addressing, and the experimental results you obtained.
- **Introduction.** A more extensive version of the abstract where you give more details, in your own words about the motivation of your work, highlight the methodology of your simulations and the most important results.
- **Methods.** This includes a more extensive discription of your approach, the scenario of your simulations with the underlying rational. There is no need to reproduce the CNN descriptions of Section , you can just refer to them. However, any other new method should be described with sufficent detail. In any case, make sure that you provide brief but sufficient technical information for the reader to reproduce your results.
- **Results.** Include a description of your key tuning/comparison experiments you performed in the course of this project. Identify ways to present your results in a brief, clear justified and cohesive way, e.g. by making use of a table, bar graph etc. Do not include unessary information, only the important results that can be justified and lead to solid conclusions.

- **Conclusions.** Summarize briefly your motivation, the technique you introduced and your results. If appropriate, discuss your intutions behind the techniques you implemented/proposed, or make conjectures abou why they performed as they did.
- **Bibliography.** Feel free to include bibliography, especially if you introduce/implement any newly published technique. It can be included in a footnote format, and it will not count towards the word limit.

In the writing, *don't* use a lot of lists and bullet points - use narrative text as much as possible. Do proof your report carefully. The report should read like a machine learning paper, and it should be comprehensive to the future students taking the EE4108. Your submissions will be graded based on their completeness in addressing all points mentioned above, their clarity, their correctness, the writing style and the readability of the results as presented in the tables/graphics.