Default Project 2 - DD2424, 2023

Teachers and TAs of DD2424 2023 and earlier

1 Building and training a modern ConvNet architecture from scratch

The second default project will explore the concept of training a ConvNet from scratch to classify Cifar-10. In the basic project you will retrace the steps of the tutorial How to Develop a CNN From Scratch for CIFAR-10 Photo Classification plus make some simple investigations. Once you have completed the basic network to get a relatively high performance then if you wish to aim for a higher grade you can pursue extensions.

1.1 Basic project to get E

The basic project will involve constructing a baseline network architecture based on the VGG network to solve the Cifar-10 dataset. The baseline network you will construct will have three VGG blocks as described in How to Develop a CNN From Scratch for CIFAR-10 Photo Classification. You can use whichever deep learning software package you like, not specifically Keras, the tutorial is mainly a guideline for the specific architecture to use and the expected performance you should achieve. Initially, train your network with the optimizer SGD + momentum and no regularization. With this network and training you should be able to achieve results $\sim 73\%$.

Next you will explore adding different types of regularization independently to the basic network:

1. Dropout regularly applied throughout the network.
   Target performance: **Baseline + Dropout**: $\sim 83\%$
2. Weight decay a.k.a L2 regularization.
   Target performance: **Baseline + Weight Decay**: $\sim 72\%$
3. Data-augmentation - horizontal flipping + $x-$ and $y-$ translation shifts.
   Target performance: **Baseline + Data Augmentation**: $\sim 84\%$

Then you should train the network with multiple regularization strategies combined to increase performance plus the addition of batch normalization. As you are applying more regularization, you might need to train for longer but the use of batch normalization may allow you to use a higher learning rate. The final training regime should include higher levels of dropout, data augmentation and batch normalization. The target performance of this network but training set-up is $\sim 88\%$.

All these stages are explicitly described in the tutorial. Thus you need also to perform some explorations not so demarcated in the tutorial. You should try and explore each of the following for the final set-up

- The tutorial normalizes the input data by ensuring it is between 0 and 1. See if normalizing the data to have zero mean and standard deviation 1 (as in the programming assignments) has any effect.
- Replace the SGD + momentum optimizer with Adam and then AdamW. Do these optimizers lead to better performance and/or faster convergence?
- Try different learning rate schedulers such as learning rate warm-up + cosine annealing, step decay or cosine annealing with re-starts and see how is helps/affects training.
- It is not clear in which order Dropout and Batch Norm should be performed. In the tutorial it is BatchNorm then dropout. Check if changing the order to Batch Norm then Dropout has an effect on performance. Also check if the Dropout and Batch Norm are complementary ie having both Dropout and Batch Norm in the network is better or worse than having a network that just has one of these regularization techniques.

1.2 Extending the basic project to get a higher grade

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade.

1.2.1 From E $\rightarrow$ D/C

If you are aiming for a D or C then here are some extensions from the basic project you could apply to investigate if it was possible to improve performance. You will probably need to investigate > 1 of these extensions to get up to a C grade:

- Make the VGG network architecture you have constructed in the basic assignment look more like a ResNet with the following steps:
  - Replace the large fully connected layer + final fc output layer with a *Global Average Pooling* layer + final fc output layer
  - Replace each MaxPooling layer with a convolution layer applied with stride two.
  - Add skip connections within the network. Look carefully at the ResNet architecture to see faithful ways to do this.

  Set up this new architecture and see if training becomes easier and/or if better results can be achieved. If there is no difference perhaps make the network deeper with a fourth block and compare the results of the two architectures with this deeper network.
- Try more extensive data-augmentations (more geometric data-augmentations: affine transformations, scaling and rotation and/or photo-metric augmentations.) and see if this gives a performance boost.
- Replace Batch Norm with Layer Norm or Instance Norm or Group Norm and see if you can maintain performance levels with the simpler normalization practice.

1.2.2 From E → B/A

If you are aiming for a B or A then here are some possibles extensions from the basic project you could apply to investigate if it was possible to improve performance.

- Train a ResNet architecture from scratch with more bells and whistles that you can find in the literature to get a final test accuracy ≥ 90 on Cifar-10 (within a reasonable training time) and perform training on Cifar-100 to see if the same training and approach can produce good results on a dataset with more classes.
- Speed up training by *quickly and efficiently finding the core set* of examples actually needed for training. There are many ways to do this, but one relatively straightforward and effective general approach is described in SELECTION VIA PROXY: EFFICIENT DATA SELECTION FOR DEEP LEARNING, Cody Coleman et al., ICLR 2020. The high-level idea is to train a ConvNet, which works well but perhaps not amazingly but can be trained relatively quickly, on all the labelled training data. Use the trained *simple ConvNet* to identify the training examples most useful for training - for example keep the training examples whose outputs have highest entropy. This set is termed the core-set, see section 2.2 of the paper for more details. As usually the training set has a lot of redundancy this core-set may correspond to approximately up to 50% of the original data but obviously this will vary from dataset to dataset etc. If you then train the big complicated network on just the core-set then it is often possible to achieve a final test performance similar to training on all the data but you can train the network trains much more quickly. You can use the large VGG net you trained as your "complicated network" and either a smaller VGG network or the big VGG network not trained for so long as your simple network. Then you can explore finding the core-set with the entropy based measure or other measures (if you have the time or patience to implement them) and then train the large network for a long time on just the core set.
- Contaminate the labels of your training data with noisy labels and investigate how it corrupts training and explore an option in the literature to train a network in the presence of noisy labels. An interesting dataset that has pre-defined label noise but is not too big to perform reasonable computations is ImageNette There is a lot of work in this area. Please check out the Custom Projects to see links to relevant papers in the literature.

You are, of course, very welcome to come up with your own extensions. But do remember to specify them explicitly in the project proposal so you can get them properly vetted by a TA. If you do go for an extension from an E to an A then most of the project report should be devoted to the extension as opposed to the basic project. For the basic assignment you should report the main results and put the more extensive fine-tuning results in the appendix.