EE 569 Homework 6 Write Up

Armin Bazarjani

April 30th, 2021

1 Origin of Green Learning

1.1 Motivation

Green Learning in a nutshell is an attempt to move away from the extremely computationally heavy modern age of deep learning. The "green" in green learning referring to the fact that the computations have a surprisingly large carbon footprint. Thus, with less computation we have a lower or "greener" carbon footprint. Now, if we were to say the only motivation for green learning is to have a lower carbon footprint we would be disingenuous. From what I have seen the main motivation for green learning can be understood from an engineering perspective. If we want to match the computational efficiency of our brain, we need drastically more efficient learning models. On top of that, the simpler and more efficient these models become, the more explainable they might also become in the process as well. This will become an ever more important problem in artificial intelligence seeing as the current trends only favor those with the resources required to train deeper models with millions of more training examples. We just simply cannot continue this trend and must look past the short term gain of getting 0.01% higher accuracy on the same benchmarks.

1.2 Approach and Procedures

This question required reading various research papers so my approach was to simply read and take notes on every single research paper.

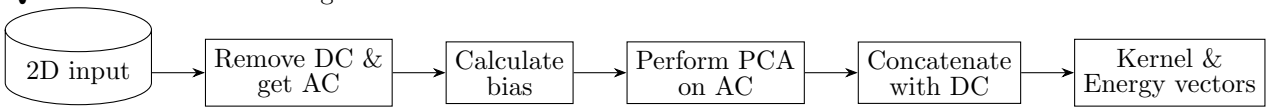
1.3 Experimental Results

There are no experimental results to report for this problem as the entire problem consisted of reading research papers and answering questions about them.

1.4 Discussion

1.4.1 (a) Feedforward-designed Convolutional Neural Networks

Question 1 - The flow diagram for Saab transform can be found below.



Flowchart for SAAB

Question 1:
To give some more detail to the flowchart, I will give a brief explanation of my understanding of the saab transform process. At a very high level, the saab transform is meant to replace a convolutional layer. It is meant to be a feature extractor where the non-linearity that we traditionally get through an activation function is achieved through the addition of a bias vector.

The saab transform begins by taking in a 2D matrix, that being either the input image or the output from a previous hop. Once it recieves this matrix, it removes the DC and calculates the AC components, those being different anchor vectors. The AC terms are used to calculate the bias at the current hop. After, PCA is applied to the AC features to get descriptive features in our kernel as well as an energy vector. This new kernel is concatenated with DC features and the energy vector is concatenated with the DC energy vector.

Question 2:
FF-CNN and BP-CNN are both similar in the sense that they are data driven techniques that work well because they are able to extract descriptive feature representations of our input images. They are also similar in that they are uniquely specialized for image classification problems.

This is essentially where the similarities end. perhaps the biggest difference between the two techniques is that the FF-CNN method does not require backpropagation and it is a one pass design that relies more on linear algebra and statistics than its non-convex gradient descent counterpart. BP-CNN requires multiple iterations of backpropagation and fine-tuning. Additionally the FF-CNN method is fully transparent and interpretable, unlike its BP-CNN counterpart that falls prey to the "blackbox" of deep learning. Another difference is that the FF design does not have any architectural limitations, the biggest limitation for the BP design is that it has to enable end-to-end backpropagation optimization.

1.4.2 (b) PixelHop and PixelHop++

Question 1:
Successive Subspace Learning (SSL) is the process of iteratively learning and

fine-tuning the subspace representation of a set. A set can be anything, for example the MNIST digit dataset or the Fashion-MNIST clothing dataset. The main steps of SSL are near-to-far neighborhood expansion, unsupervised dimension reduction, supervised dimension reduction, and concatenating features for decision making.

Deep Learning (DL) and SSL are quite different ways of doing similar things. They both attempt to find underlying patterns within data (specifically within image data), and they both do this through successively growing neighborhoods. The biggest difference between the two methods is how this is done, which lends itself to why SSL is much more interpretable and flexible than DL. On top of that, SSL is a much more efficient technique that comparatively does not require nearly as many resources as DL. Another difference that I personally found very interesting too was that SSL does not require as much supervision as DL methods, it is what is known as "weakly supervised".

Question 2:
The first module is responsible for successive near-to-far neighborhood expansion. The attributes of pixels that are both near and far can be propagated to the target pixel through communication, attributes being anything from pixel location, to intensity, etc. We start local and then go global, the further away we go from the target pixels, the more hops we need. Hence Pixel Hop.

The second module is responsible for unsupervised dimension reduction, this is an approximate subspace as it is unsupervised. Specifically, we are interested in finding statistical correlations between the attributes of different neighborhoods. PCA is a one stage technique, in the context of SSL, and the paper, they opt for the more sophisticated saab or saak transformation.

The third module is supervised dimension reduction through label assisted regression. In a classification problem the attributes of near and far neighbors follow different distributions for different classes, this is information that we would like to exploit. The label information throughout the different stages is used in conjunction with the label-assisted regression (LAG) unit.

Question 3:
The neighborhood construction step consists of taking the union of a center pixel and it's eight nearest neighborhood pixels. The successive approximation step consists of using the Saab transform that we went over in section "1.4.1 (a) Feedforward-designed Convolutional Neural Networks".

The difference between the basic Saab transform and the cannel-wise Saab transform is exactly as the name suggests. Instead of taking the Saab transform over every single channel at once, we now split the input into each of its respective channels and take the Saab transform of each channel separately.

2 MNIST & Fashion-MNIST Classification

2.1 Motivation

In order to assess the usability of a new method, especially within the realm of deep learning, it is common to apply the new method to well known and well defined existing benchmark dataset. This serves as a proof of concept for the method. If it can perform well (or reasonably well depending on the method itself) on these benchmark datasets, then the creators can argue that it will serve some utility within the community. Or at the very least, there is a new method for solving the same problem with a different set of constraints on how the problem can be solved.

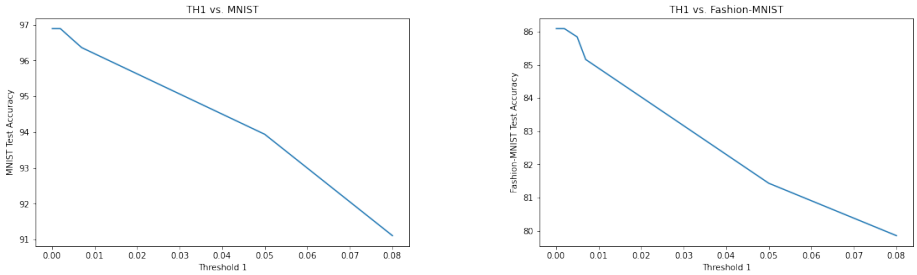
2.2 Approach and Procedures

We were given all of the necessary code for PixelHop and PixelHop++. The only thing left to the students was to put everything together within the "main.py" file. For me personally, I find that when I am working on a machine learning problem, I have a much easier time doing it within a notebook (ie. jupyter). The TA's said that this was okay, so I did all of the problems for this assignment in separate jupyter notebooks for each problem.

I don't think there will be too much variability on how other student's solved these problems as we were guided pretty heavily to all converge on the same solutions. I think the only thing that I did differently from the other students was that I did not use the block_reduce function when doing max-pooling. Instead I took inspiration from Stanford's cs231n course and used a clever reshape technique to do max-pooling. This reshape technique only works if your input is square, kernel size is equal to stride, and input is a multiple of the kernel size.

2.3 Experimental Results

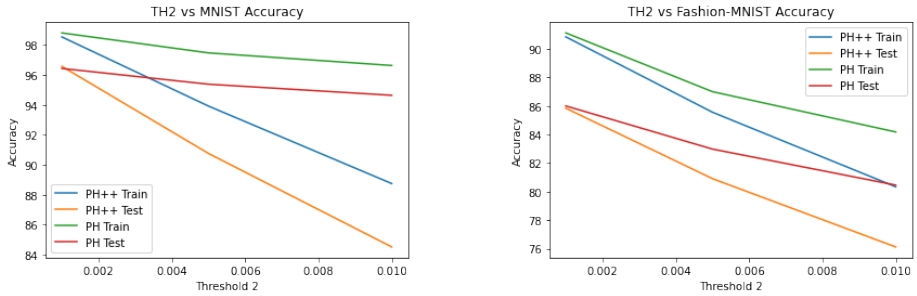
2.3.1 Training PixelHop++ on MNIST and Fashion-MNIST



(a) MNIST (b) Fashion-MNIST

Figure 1: TH1 vs. Test Accuracy on both MNIST and Fashion-MNIST

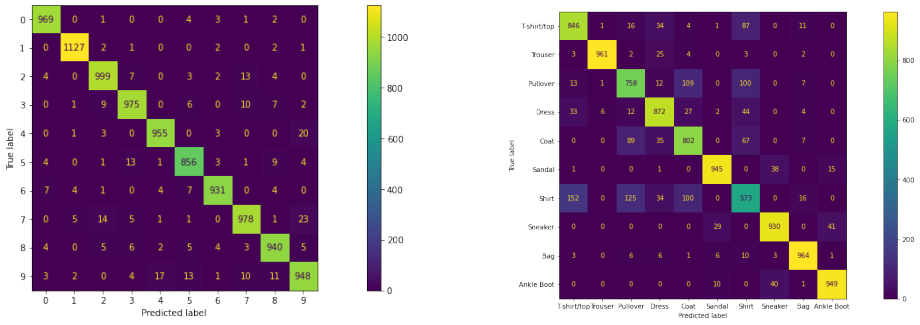
2.3.2 Comparing PixelHop and PixelHop++



(a) MNIST (b) Fashion-MNIST

Figure 2: TH2 vs. Train/Test Accuracy on both MNIST and Fashion-MNIST

2.3.3 Error Analysis



(a) MNIST (b) Fashion-MNIST

Figure 3: Confusion Matrix on both MNIST and Fashion-MNIST

2.4 Discussion

2.4.1 (a) Building the PixelHop++ Model

Question 1:
The training time was around 16 minutes for MNIST and 25 minutes for Fashion-MNIST and the training accuracy was 98.535 on MNIST and 90.835 on Fashion-MNIST. The model size, in terms of number of parameters, was 6,525 for MNIST and 3,775 for Fashion-MNIST.

Question 2:
The test accuracy was 96.57 on MNIST and 85.84 on Fashion-MNIST.

Question 3:
Please find the curves for test accuracy in section "2.3.1 Training PixelHop++ on MNIST and Fashion-MNIST". As for the model size, I noticed that my model size decreased as I increased the value for TH1. This goes along with the fact that the testing accuracy decreases as we increase the value of TH1. The reasoning being that we have less and less features to assist the classifier.

2.4.2 (b) Comparison Between PixelHop and PixelHop++

Question 1:
I ran a few trials using different values for TH2 (as was recommended by the TA on Piazza, but not mentioned in the homework). I found PixelHop to outperform PixelHop++ across the board in both training and testing accuracy. In fact, PixelHop significantly outperformed PixelHop++ at the higher levels for threshold 2. For example, with TH2=0.01 & TH1=0.005, PixelHop++ got a 84.52 testing accuracy on MNIST while PixelHop scored a test accuracy of 94.64.

Obviously, the only thing that is changing here is the fact that PixelHop++ is using channel-wise Saab transform. My intuition tells me that because Saab is being applied channel-wise, the higher the threshold, the higher probability we will lose descriptive features. Now, you do that across a lot of small channels and you lose more features than across a single large channel.

Please refer to section "2.3.2 Comparing PixelHop and PixelHop++" to find graphs I made where I plot both the train and test accuracies of PixelHop and PixelHop++ across different values of TH2 for both MNIST and FashionMNIST.

Question 2:
Because the homework did not mention changing values of TH2, I calculated the model size according to the specifications given to use in Table 1 in the homework. Thus, these model sizes are for a TH1=0.005 and a TH2=0.001 for both PixelHop and PixelHop++.

For MNIST the PixelHop model size was 133,125, and for PixelHop++ the model size was 6,525. For Fashion-MNIST the Pixelhop model size was 74,275 and the PixelHop++ model size was 3,800 Clearly we can see one of the massive benefits of PixelHop++ and thus the channel-wise Saab transform is having fewer features and greatly increasing efficiency for a small hit to accuracy.‘

2.4.3 (c) Error Analysis

Question 1:
Please find the confusion matrix for both MNIST and Fashion-MNIST in section "2.3.3 Error Analysis". Viewing the confusion matrix for MNIST, we can see "1" has the lowest error rate and "5" has the highest error rate. Viewing the confusion matrix for Fashion-MNIST, we can see that "Bag" has the lowest error rate and "Shirt" has the highest error rate.

Question 2:
We can see for MNIST the two most confusing class groups are distinguishing between "4" and "9" and "7" and "9". Intuitively, we can see why this would be the case as both 4 and 7 share a very similar structure to 9. On top of that because these are all hand-written digits, we lose the rigidity that some strokes have, making it even more difficult for certain cases.

We can see for Fashion-MNIST the two most confusing class groups are "Shirt" and "T-Shirt/Top" and "Shirt" and "Pullover". Again, this makes intuitive sense as a Shirt and a T-Shirt/Top are the fact that a shirt in this dataset is a long-sleeve version of t-shirt/top and in certain cases it can be very difficult to make out the sleeves. This is even more evident when comparing shirt and pullover as both have long sleeves and even a human would have some difficulty distinguishing between a pixelated version of the two.

Question 3:
One basic idea that we could implement, that we didn't try in this homework, is to apply some sort of image modification technique to the training images. For example, cropping, rotating, and flipping. To name a few. Another thing that I thought of was that it might be interesting to use another unsupervised dimension reduction technique inside of the Saab transform instead of PCA, maybe something like SIFT or SURF.