EE569 Digital Image Processing

HOMEWORK #6

Successive Subspace Learning

Issued: 4/8/2020 **Due: 4/26/2020**

Xu Kangyan

4876-0109-98

kangyanx@usc.edu

Problem 1: Understanding Successive Subspace Learning (SSL)

(a) Feedforward-designed Convolutional Neural Networks (FF-CNNs)

(b) Successive Subspace Learning (SSL)

I. FF-CNNs

1. Saab transform

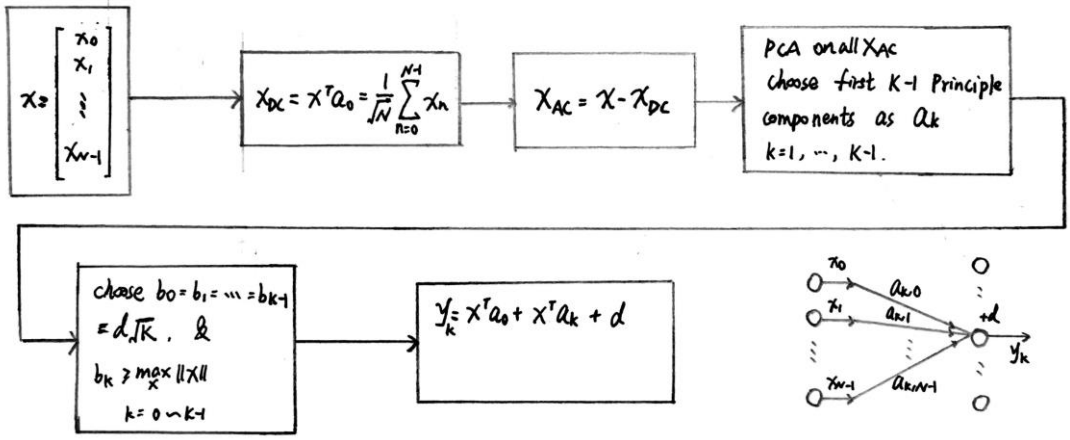

Figure 1. Saab Transform Flow Diagram

Saab Transformation is meant to avoid *sign-confusion* problem without dimension-doubling. This transformation focused on selecting **Anchor vectors** a_k (weights) and **bias** b_k. Besides, there is no need of activation function, given that the bias ensures each output y_k is non-negative. The anchor vectors are divided into two parts (*AC* and *DC*). *DC* part is similar to the mean of vector (which is, if input vector has a zero-mean, the *DC* component will also be zero), *AC* part is the complement of *DC*, to select anchor vectors, **PCA** is done on all possible X_{AC} and $K-1$ principle components are selected as anchor vectors (each has a dimension of N).

2. Similarities and Differences between FF-CNN and BP-CNN

Similarities:

The basic idea is same, which is *dimension reduction*, the input dimension is reduced after operation in each layer until it meets the output dimension.

Differences:

FF-CNN uses *vector space transformation*. To replace filter convolution layer, FF uses *subspace approximation* and *projections*. To replace fully connection layers, FF uses *training sample clustering* and *remapping*. In the former part, **PCA** is used to obtain a set of *pre-selected spatial patterns* for pixels projection (Saab), not like BP uses cost function and chain rule to update the weight and bias parameters. In the latter part, *LSR guided by pseudo-labels* are used to obtain strong discriminability of certain dimensions.

II. Successive Subspace Learning

1. SSL methodology and Comparison

The SSL has four parts. The first is using a series of *PixelHop units* in cascade to realize a successive neighborhood expansion, each unit uses previous unit's output (after max pooling) as its input. Then in each unit *Saab transform* is used to control the dimension. Third, each unit's output goes through *aggression* and *label-assisted regression unit*. Finally, output features from each unit are concatenated and then classified.

Both methods collect attributes from pixel and its neighborhood, then using pooling to reduce redundancy.

There are many differences between two methods. SSL can adjust its model complexity, has no BP, interpretable, lower training complexity, partly using task-independent features, more robust facing attacks, and so on.

2. Functions of Modules 1, 2, 3

Module 1 - A sequence of PixelHop units in cascade

In this module, each unit enlarges neighborhood, so the target pixel with dimension K_{i-1} becomes $K_{i-1} \times (N_i + 1)$, then Saab transform changes this dimension to a new K_i. Besides, pooling is used here to reduce spatial resolution. In general, the $S_{i-1} \times S_{i-1} \times K_{i-1}$ input turns into the $S_{i-1} \times S_{i-1} \times K_i$ output (attribute), which is passed to module 2. Then $S_i \times S_i \times K_i$ output after pooling is passed to next unit.

Module 2 - Aggregation and LAG unit (supervised dimension reduction)

In this module, *aggregation scheme* (maximum, minimum, mean values in small regions) is used to turn $S_{i-1} \times S_{i-1} \times K_i$ input into $P_i \times P_i \times K_i$ output (P_i is hyper-parameter). Then we use *supervised learning* to reduce this dimension to M, gets a feature vector with M features.

Module 3 - Feature concatenation and Classification

Each *PixelHop unit* will obtain a feature vector, these features are concatenated to form a $M \times I$ features (I represents number of *PixelHop units*). Eventually a *multi-class classifier* is trained to do classification.

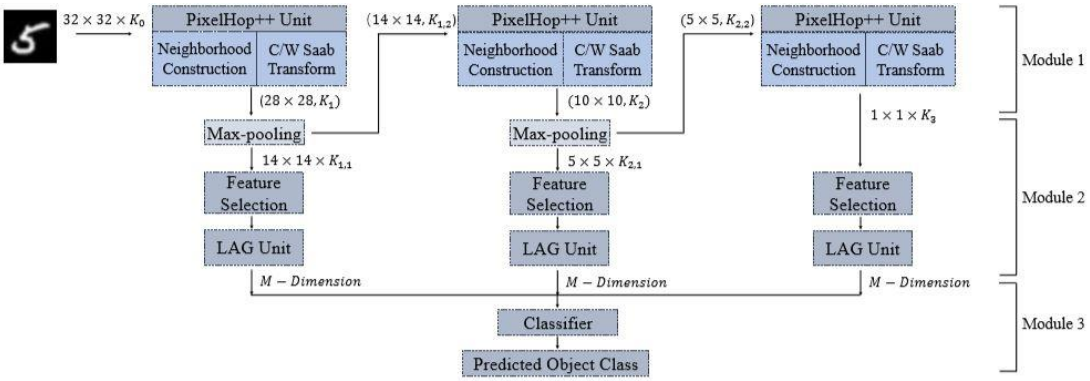

Figure 2. Three Modules in *PixelHop++* method

3. Neighborhood construction and subspace approximation steps

PixelHop unit:

As described in above Module 1 part.

PixelHop++ unit:

The *Saab transform* is replaced by *channel-wise Saab transform* here. That is, the input tensor with size $S_i \times S_i \times K_i$ is **decomposed** into K_i tensors, each has a size $S_i \times S_i$. Then, *c/w Saab transform* takes the K_i channels input and returns K_{i+1} channels output, each also has the former size $S_i \times S_i$.

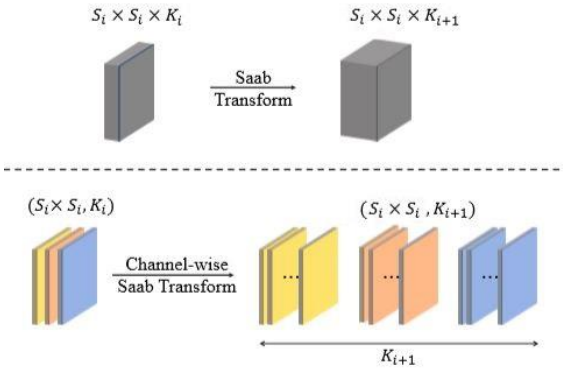

Figure 3. Difference between *Saab* and *channel-wise Saab Transform*

4. Label-Assisted reGression (LAG)

LAG is used to reduce the dimension of the attributes. It is a supervised dimension reduction technique, which uses data label.

First, set J object classes and partition each class into L clusters (Here I think each cluster represents a word in Bag-of-words). Then probability vector of sample is set, which depends on distance between sample and centroid. Eventually, linear *Least Squares Regression* equations are set to link the input vector (attributes) and the probability vector.

The Advantage:

LAG makes the use of label information, reduces the dimension of final feature vector and it is based on *K-means* and *Least Squares Regression*.

Problem 2: CIFAR-10 Classification using SSL

(a) Building a PixelHop++ Model

(b) Error analysis

I. Building PixelHop++ Model using Python

1.

In module 1, max pooling function is added in the shrink function. Due to the performance of laptop, when set the window size to 2 and using 5000 training images the training is quickly done, however the window size 5 with 10000 training images cost nearly 0.5 hour.

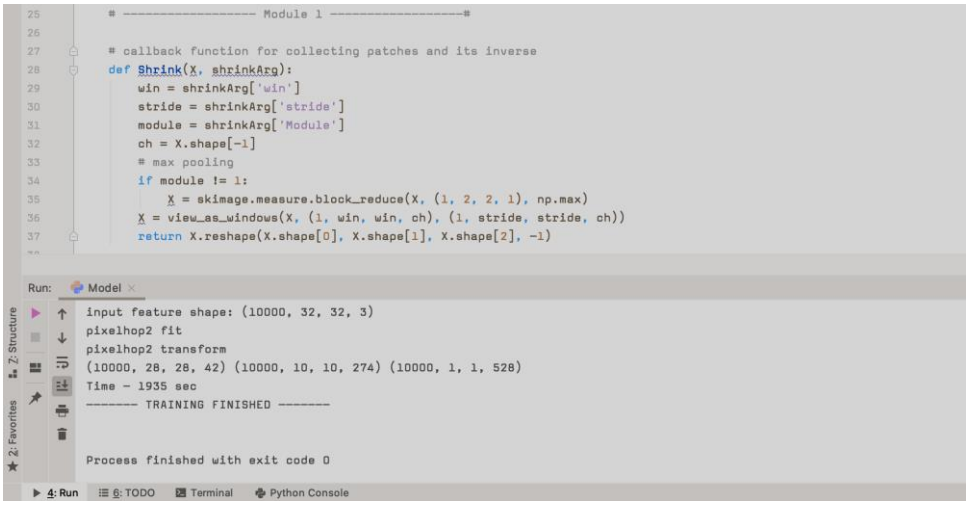

Figure 4. Module 1 Training with 10k images

Due to the low performance of devices, I cannot use 50000 training images (actually no response from laptop). So I only use very low amount of data to go through the process.

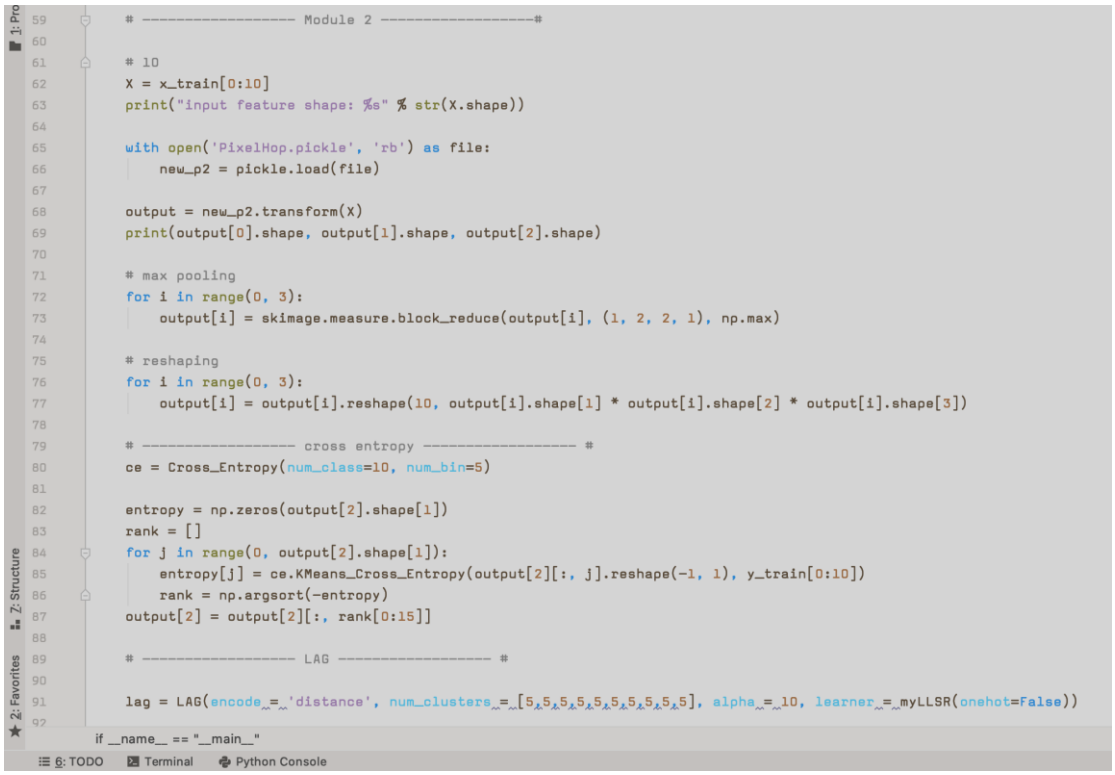

Figure 5. Module 2 Cross Entropy with only 10 images

The ideal output of three *PixelHop++ Unit* should be $50000 \times 28 \times 28 \times 42$, $50000 \times 10 \times 10 \times 274$ and $50000 \times 1 \times 1 \times 528$. After Max-pooling, those outputs are reshaped to be 50000×8232, 50000×6850, 50000×528. Take first output as an example, 8232 times of K-means should be done and after comparison, 1000 dimensions should be kept to realize

dimension reduction. However I only test on 10×528 and reduce its dimension to 10×15.

Ideal model size:

A. parameters of Saab filters: $K_1 = 42, K_2 = 274, K_3 = 528$.

B. Regression matrix in LAG units: $50 \times n$

C. Classifier: $3Hops,\ M = 50, 10classes, Parameter\ number = 1500$.

2. 10000 test images cannot be done.

3. Training sample changes cannot be done.

2. Error analysis

1. Confusion matrix cannot be done.

2. confusing class groups cannot be done.

3. Propose ideas cannot be done. Maybe Max-pooling can have some changes.