COL774 - Machine Learning: Assignment - 4

Param Khakhar - 2018CS10362, Shreyans Nagori - 2018CS10390

28 December 2020

Introduction

The following is a writeup for the Assignment-4, Machine Learning (COL774). There are sections corresponding to each of the two questions and within each section, there are sub-sections for each sub-part.

Task -1 : Non Competitive Experimentation

The test set used here is the public test set. We have hardcoded the no of epoches accordingly as per time limit.

Task - 1.a: Vanilla Neural Network

The optimizer used is SGD and the learning rate used is 0.1. The following are the different Macro-F1 scores obtained during the experimentation with different activation functions.

Activation Function	Macro-F1-Score on Train Set	Macro-F1-Score on Test Set
ReLU	0.638	0.336
Sigmoid	0.725	0.317
TanH	0.827	0.351

The converge criteria is set as when the absolute difference between the losses in the last 5 epochs becomes less than 1e-04. The best training results are obtained for $TanH$ whereas the best test results are obtained for $ReLU$. ReLU activation is less prone to overfitting, as compared to others.

Task - 1.b: Feature Engineering

Once again the optimizer used is SGD, and the learning rate used is 0.5. Following are different macro F1 scores obtained during the experimentation with different activation functions and given feature engineering.

Using Gabor Filters feature learning.

Activation Function	Macro-F1-Score on Train Set	Macro-F1-Score on Test Set
ReLU	0.145	0.141
Sigmoid	0.0990	0.0980
TanH	0.1526	0.1412

Using Histograms of Gradients

Activation Function	Macro-F1-Score on Train Set	Macro-F1-Score on Test Set
ReLU	0.1602	0.1591
Sigmoid	0.0573	0.0573
TanH	0.1601	0.1594

In Gabor Filters I took projection along 8 directions, and then the mean and variance as the 2 features along each direction to get 16 features per feature for model training, but the accuracy was really low. Using Histogram of gradients, accuracy was slightly better here we only focus of sharp figures with high gradient of change to mark out sharp figures.

Task - 1.c: Convolutional Neural Network

On using the convolutional neural network architecture as prescribed in the assignment specification, the following scores were obtained:

- **Macro-F1 Score on Train Set:** 0.925
- **Macro-F1 Score on Test Set:** 0.323

The high accuracy on the train set is indicative of the increased complexity of the learning algorithm. However, the model doesn't generalize well. There was overfitting in the vanilla neural network and since the number of parameters are high for CNN, there would be increased overfitting along with increased time taken for training. Theoretically, CNN is expected to learn more relevant features, as compared to vanilla neural network, for image classification task.

Task -2 : Competitive Experimentation

Unsuccessful Approaches:

The following sections comprises the unsuccessful approaches tried by us, along with their results:

- PCA (400 components, 96.7% explained variance) on the original data followed by using SVM (rbf-kernel) for the classification. The results are as follows:
  - F1-Score on Train Set: 0.553
  - F1-Score on Validation Set: 0.347

  Here, validation set refers to the public test set.

- PCA (400 components, 96.7% explained variance) on the original data followed by using Fully Connected Neural Network [400, 512, 64, 7] for the classification. The results are as follows:
  - F1-Score on Train Set: 0.058
  - F1-Score on Validation Set: 0.058

  Even with varied number of hidden layers and neurons in the layers, we didn't get much improvement.

- (Conv(1,32)-RELU-BatchNorm)
  Maxpool((2,2),stride=2)
  (Conv(32,64)-RELU-BatchNorm)
  Maxpool((2,2),stride=2)
  (Conv(64,128)-RELU-BatchNorm)
  Maxpool((2,2),stride=2)
  FC(512,128)-RELU
  FC(128,64)-RELU
  FC(64,7)-RELU
  Softmax(7).

  The idea was to form a simple CNN model consisting of 3 convolutional layers followed by 3 Fully connected layers and then the softmax layer.
  - F1-score on private data = 0.46.

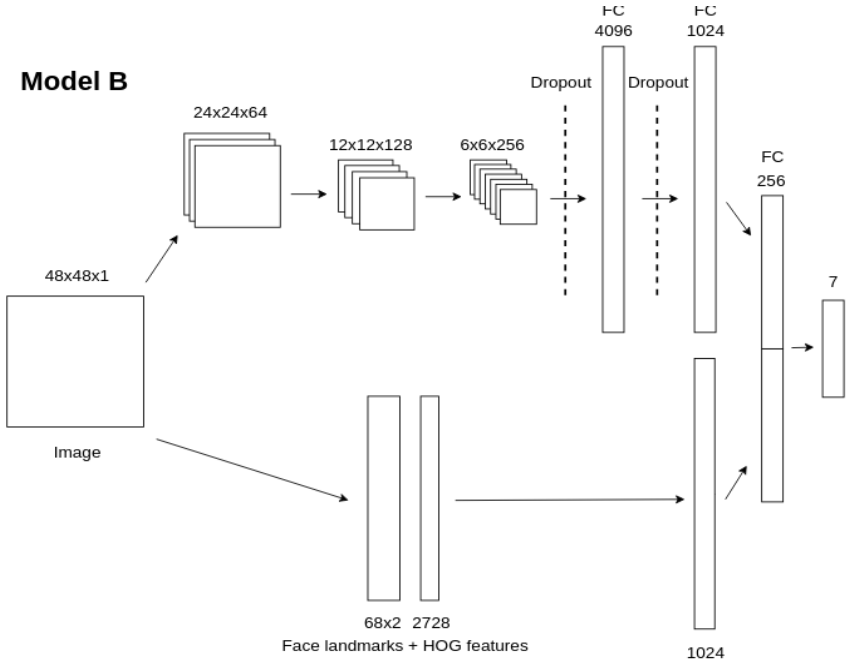

Figure 1: Model used in order to add HOG, F1 score rose to 0.48.

- (Conv(1,64)-RELU-Conv(64,64)-RELU-Conv(64,64)-RELU-BatchNorm)
  Maxpool((2,2),stride=2)
  (Conv(64,128)-RELU-Conv(128,128)-RELU-Conv(128,128)
  RELU-BatchNorm)
  Maxpool((2,2),stride=2)
  (Conv(128,128)-RELU-Conv(128,128)-RELU-Conv(128,128)
  RELU-BatchNorm)
  Maxpool((2,2),stride=2)
  (Conv(128,128)-RELU-Conv(128,128)-RELU-Conv(128,128)-RELU-BatchNorm)
  Maxpool((2,2),stride=2)
  (Conv(128,256)-RELU-Conv(256,256)-RELU-Conv(256,256)-RELU-BatchNorm)
  FC(2304,1024)-ReLU-BatchNorm1d(1024)
  RFC(1024,256)-ReLU-BatchNorm1d(256)
  FC(256,7)ReLU-BatchNorm1d(7)
  Softmax(7).
  - F1-score on train data = 0.513
  - F1-score on private data = 0.493.

Note: The optimizer used is Adam with the learning rate of 0.01, for all the experiments.

- Architecture-1:

  Conv2d(1,32,(3,3),1)-ReLU-BatchNorm2d(32)
  Conv2d(32,64,(2,2),2)-ReLU-BatchNorm2d(64)
  MaxPool2d((2,2),stride = 2)
  Conv2d(64,64,(3,3),1)-ReLU-BatchNorm2d(64)
  Conv2d(64,128,(2,2),2)-ReLU-BatchNorm2d(128)
  MaxPool2d(2,2),stride = 2)
  Linear(512,256)-ReLU-BatchNorm1d(256)
  Linear(256,64)-ReLU-BatchNorm1d(64)
  Linear(64,7)- ReLU-BatchNorm1d(7)
  Softmax(7)
  - F1-Score on Validation Set: 0.483
  - F1-Score on Private Set: 0.475

- Architecture-2:

Conv2d(1,32,(3,3),2)-ReLU-BatchNorm2d(32)
Conv2d(32,64,(3,3),1)-ReLU-BatchNorm2d(64)
MaxPool2d(x,(2,2),stride = 1)
Conv2d(64,64,(3,3),1)-ReLU-BatchNorm2d(64)
Conv2d(64,64,(3,3),1)-ReLU-BatchNorm2d(64)
MaxPool2d(x,(2,2),stride = 2)
Conv2d(64,64,(3,3),1)-ReLU-BatchNorm2d(64)
Conv2d(64,128,(3,3),1)-ReLU-BatchNorm2d(128)
MaxPool2d(x,(2,2),stride = 2)
Linear(512,256)-ReLU-BatchNorm1d(256)
Linear(256,64)-ReLU-BatchNorm1d(64)
Linear(64,7)-ReLU-BatchNorm1d(7)
Softmax(7)

  - F1-Score on Validation Set: 0.517
  - F1-Score on Private Set: 0.515

There are other architectures, which were variations of the submitted approach, in terms of the number of hidden layers used, in-out channels for convolution layers, but we weren't getting any significant improvement.

Submitted Approach:

- Architecture:

Conv2d(1,32,(3,3),1,padding=1)-ReLU-BatchNorm2d(32)
Conv2d(32,64,(3,3),1,padding=1)-ReLU-BatchNorm2d(64)
MaxPool2d(x,(2,2),stride = 2)
Conv2d(64,64,(3,3),1,padding=1)-ReLU-BatchNorm2d(64)
Conv2d(64,64,(3,3),1,padding=1)-ReLU-BatchNorm2d(64)
MaxPool2d(x,(2,2),stride = 2)
Conv2d(64,64,(3,3),1,padding=1)-ReLU-BatchNorm2d(64)
Conv2d(64,64,(3,3),1,padding=1)-ReLU-BatchNorm2d(64)
MaxPool2d(x,(2,2),stride = 2)
Conv2d(64,64,(3,3),1,padding=1)-ReLU-BatchNorm2d(64)
Conv2d(64,128,(3,3),1,padding=1)-ReLU-BatchNorm2d(128)
MaxPool2d(x,(2,2),stride = 2)
Linear(128*3*3,512)-ReLU-BatchNorm1d(512)
Linear(512,128)-ReLU-BatchNorm1d(128)
Linear(128,7)-ReLU-BatchNorm1d(7)
Softmax(7)

  - F1-Score on Validation Set: 0.571
  - F1-Score on Private Set: 0.567

- **Remarks:**
  - The architectures tried were mainly of the form (Conv-ReLU-BN-Conv-ReLU-BN-MaxPool2D)*X-(Linear-ReLU-BN)*Y-Softmax, for different values of X and Y. We tried X = $2, 3, 4$ and Y = $2, 3, 4$. It seems as if the model is underfitting, and therefore, we observed that increasing the values of X resulted in increased accuracy.
  - We also tried adding dropout layers along with L2 regularization of weights, but didn't see any improvements in the final score.
  - With increased complexity of the architecture, the training time also increased.
  - Initially, our plan was to integrate multiple models as ensemble but, we couldn't find more than 1 model with approxixmately same or close F1-Score.