Data Classification with Support Vector Machine and BastNet

Kenneth R. Mills
Department of Electrical and Computer Engineering, Temple University
tud10222@temple.edu

Introduction: A machine learning classification challenge is presented, wherein the goal is to classify data using a deep learning approach and a non-neural network approach. Training, development, and evaluation data sets are provided for two-dimensional and five-dimensional data. The classifiers are, of course, trained with the training data then assessed and adjusted based on their performance on the development data before ultimately being scored with the evaluation data. The 2D data is initially used to prototype the classifiers because the data points can be easily visualized and there are fewer points leading to faster training times. I make extensive use of MATLAB's built-in machine learning development tools to streamline the training process. After choosing a machine learning algorithm and deep learning architecture through training and evaluation on the 2D data, I adjusted the classifiers to take 5D inputs and proceeded to develop the classifiers for the 5D data set. I present below the trained classifiers and results produced.

Support Vector Machine: For the non-neural network approach I decided to use support vector machines (SVMs) because their well-defined mathematical definitions fill me with a sense of security. MATLAB has a Classification Learner App as part of its Statistics and Machine Learning Toolbox. This app allows a user to import a labelled data set and easily train various classifiers including decision trees, k nearest neighbors, and SVMs. For the SVM training, one can select a kernel function as either linear, quadratic, cubic, or Gaussian. The app supports parallel processing for training multiple models simultaneously, which I exploited to choose the kernel with best performance. I trained each SVM with a 10-fold cross validation scheme. After training in the app, the model can be exported to the MATLAB workspace where it can be saved to a .mat file, so the trained model can be called in any MATLAB script. The SVM was trained on my workstation CPU, which is an Intel Xeon E5-1630.

BastNet: For the deep learning approach I decided to start with transfer learning. My goal was to take an existing pre-trained network and adjust the initial layers to take the data sets from this challenge and to adjust the final layers to give an output as one of two classes. The MathWorks provide several popular networks for download through MATLAB's Add-On Explorer, and I decided to work with GoogLeNet because I am a fan of Google's services. MATLAB also has a Deep Network Designer App that allows users to easily visualize their network. The app more importantly allows one to design and analyze a network through a simple GUI. I found that the transfer learning process would not be as easy as expected in part due to the many convolutional and max-pooling layers changing the dimensions of the data as it propagates through the network. I ultimately decided to prune the network to just the first 24 layers and the last 4 layers. Effectively, I truncated the network to the first inception module. I then adjusted the sizes of the first convolutional and max-pooling layer to support the input data dimensions, and similarly I adjusted the output layer to accommodate the output classes. Figures 1 and 2 below show GoogLeNet and my inception network architectures. Furthermore, since GoogLeNet was originally designed for image classification, I transformed all data vectors into images, saved as .png, by the following operation: $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}$. Thus for the 2D data set, I had 2-by-2 images and for the 5D data set, 5-by-5 images. The inception network was trained on my workstation GPU, which is a Nvidia Quadro M2000. Due to an encroaching deadline training for the 5D network was done with 5000 samples randomly chosen from each class. In part due to the struggle of training this network, which feels like I am fighting with it, and because it

	2D Data Set			5D Data Set		
Algorithm	**Train**	**Dev**	**Eval**	**Train**	**Dev**	**Eval**
SVM	7.92%	8.15%	8.45%	36.92%	36.72%	37.13%
BastNet	17.61%	16.75%	17.45%	41.68%	41.58%	41.91%

Table 1. Error Rates for Trained Models

ultimately turned out to be bastardized version of GoogLeNet, I call my inception network Bastard Network, or simply BastNet.

Results: Table 1 presents the error rates of the two approaches. It is clearly seen that the SVM system performs favorably compared to the baseline system. However, BastNet performs considerably worse than the baseline.

Conclusions: I presented my classifiers for this final machine learning challenge. By leveraging MATLAB's built-in tools, I was able to relatively easily train a support vector machine and deep learning system to classify the data sets provided. I trained a SVM with Gaussian kernel for the non-neural network approach and developed my BastNet which uses the first inception module of GoogLeNet for the deep learning approach. The SVM was trained on a CPU, while BastNet was trained on a GPU. The evaluation results showed the SVM model to be comparable to the baseline performance, while BastNet shows considerably poorer results.

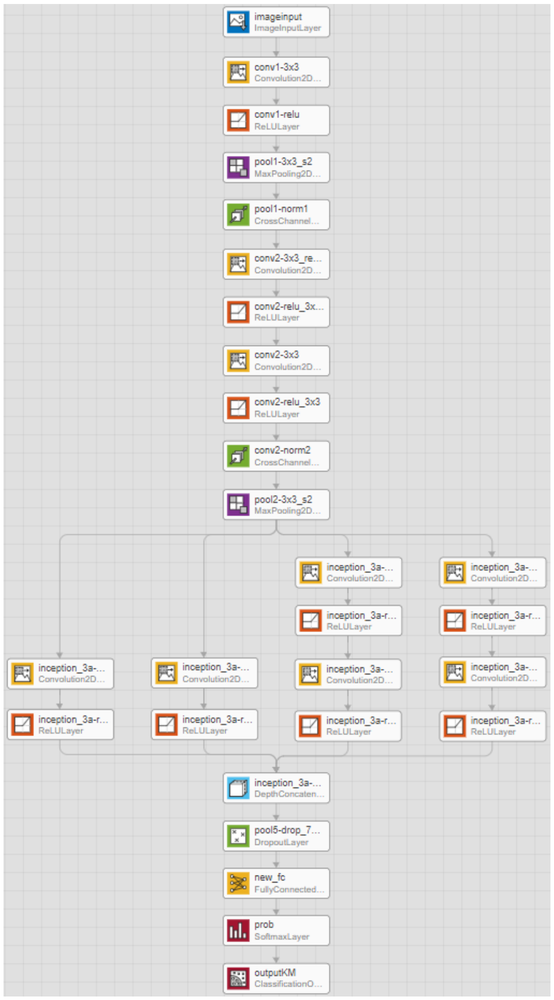

Figure 1. BastNet

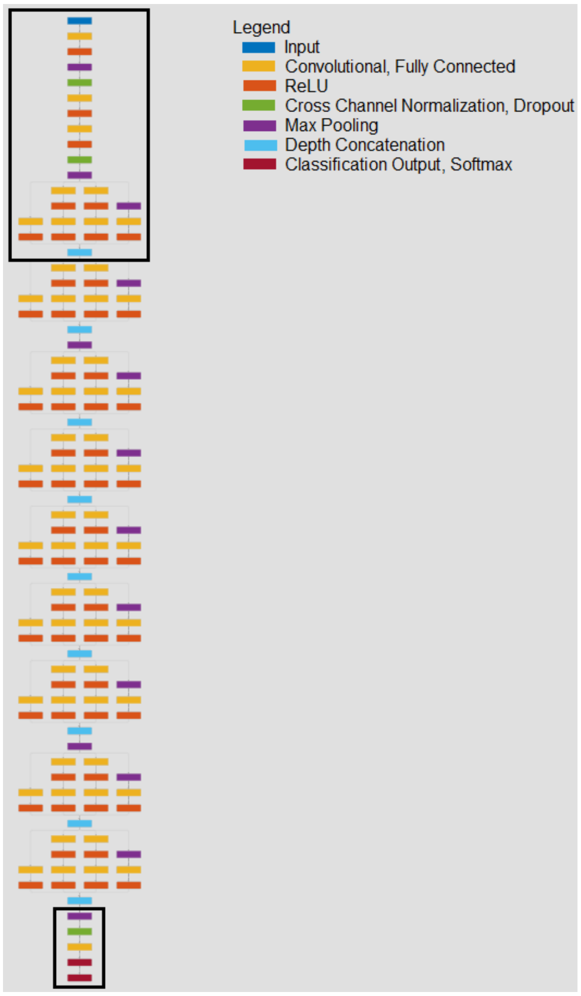
Legend
Input
Convolutional, Fully Connected
ReLU
Cross Channel Normalization, Dropout
Max Pooling
Depth Concatenation
Classification Output, Softmax

Figure 2. GoogLeNet, boxed layers are retained in my network