Smart learners: Choosing What to Learn Using Bimodal Distribution Removal

Arnab Rakshit
Research School of Computer Science, Australian National University

Abstract. Neural networks work really well on large datasets; however real world datasets aren't always plentiful. Moreover, real world data is noisy and includes outliers. Outliers leads to a larger variance in model loss (or error) and thereby increasing the time needed to train the network and also leading to lower accuracies. My focus in this exploration is the use of transfer learning [4][5] technique to extract features using the pre-trained VGG16 [6] CNN model and applying Bimodal Distribution Removal (BDR) technique [1] to eliminate outliers gradually. I attempted to use this technique on two different datasets to determine how well its suited for various type of datasets. In both cases training accuracies sharply increased whereas validation accuracy improved in the first dataset but suffered in the second dataset which was much smaller in number of samples than the first dataset. I discuss some possible reasons for this observation and suggest future improvements.

Keywords: Neural Networks, Outlier detection, Bimodal distribution removal, Transfer learning.

1 Introduction

The first dataset used in this paper is the Letter Recognition dataset[1] from UCI Machine learning Repository. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet and establish effectiveness of Bimodal Distribution Removal technique, henceforth called BDR.

The second dataset used is the American sign language alphabets dataset[2] (ASL Dataset) from Kaggle. The objective with this dataset is to apply BDR on a more realistic setting i.e. on real images first by extracting features using the pre-trained VGG16 CNN model using transfer learning technique and applying BRD during classification.

1.1 Background about the data

The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. A sampling of the data is in figure 1.

```
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	T	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8\n
1	I	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10\n
2	D	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9\n
3	N	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8\n
4	G	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10\n

Fig. 1. These are the first 5 records in the letters dataset. The first column (named 0) contains the target value for each example. The target values are one of the 26 English alphabets.

[1] http://archive.ics.uci.edu/ml/datasets/Letter+Recognition.

[2] https://www.kaggle.com/dansbecker/running-kaggle-kernels-with-a-gpu/data

The ASL dataset contains 87,000 images which are 200x200 pixels. There are 29 classes, of which 26 are for the letters A-Z and 3 classes for SPACE, DELETE and NOTHING. These 3 classes are very helpful in real time applications, and classification. The test data set contains a mere 29 images, to encourage the use of real world test images. A sampling of the dataset is below:

1.2 Characteristics of the data set

This data is fairly evenly distributed (Fig 2.) among the 26 letters of the English alphabet so problems associated with skewness of dataset doesn't arise. This dataset is naturally noisy (*randomly distorted noise was introduced*). Since the focus of this paper is to investigate the learning behaviour of a neural network while applying the technique of Bimodal Distribution Removal the letters dataset is a great fit for its purpose as we don't have to deal with other anomalies. Similarly, the ASL alphabet data is also evenly distributed among all 29 classes as shown in Fig 3 making it a good fit.

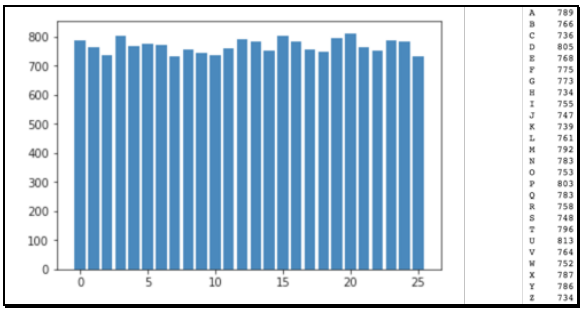


Fig. 2. Evenly disturbed data. x-axis contains numbers 0 – 25 which correspond to each of letter, e.g. 0 is A while 25 is Z. y-axis is the count of each letter.

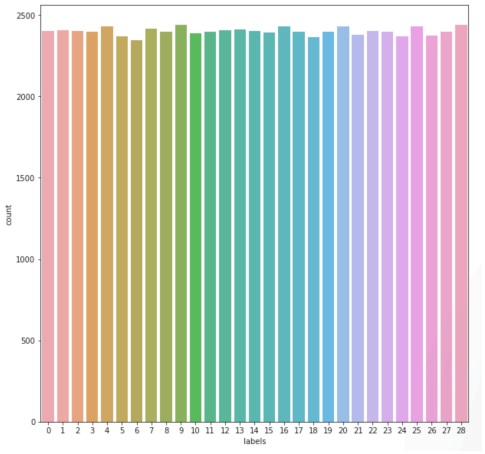


Fig. 3. Evenly disturbed data. x-axis contains numbers 0 – 28 which correspond to 26 letters A-Z and 3 classes SPACE, DELETE and NOTHING. y-axis is the count of each letter. [7]

1.3 Objectives

The goal in this investigation is to apply the method of Bimodal distribution removal (BRD) to determine how well it works on letter classifications. Furthermore, applying some variations to the original BRD method

introduced by P. Slade & T.D. Gedeon [1] is also a side objective. Finally, applying BRD in a more realistic setting on a real life dataset using transfer learning with a pre-trained VGG16 model is the end goal.

2 Method

This paper uses neural networks which have humble beginnings in the 1980's. However, it uses modern tools which makes training neural networks less programing intensive by abstracting back propagation algorithm and parameter updates from the end user. This paper applies a technique from the 1980's-1990's namely Bimodal Distribution removal to remove outliers so that the network trains faster and with lower variance. Outlier removal is a way to introduce bias into a non-parametric model which is necessary because real world data is not always sufficiently large. [1][2]
As mentioned earlier two distinct datasets were used for two distinct experiments in this paper. The first data set did not need much preprocessing as it had hand coded features. However, the ASL alphabet dataset was raw images so a pre-trained VGG16 CNN model was used to extract features from the raw images as an additional preprocessing step before feeding the features to a fully connected neural network classifier.

2.1 Model Design

2.1.1 Model and data pre-processing for Letter Recognition dataset:

The neural network model in this case is based on is a 3-layer neural network with 16 input neurons, 50 hidden neutrons and 26 output neurons. A dropout of 0.3 was introduced before the hidden layer to reduce over-fitting. Experiments were also done on a 200 neurons in the hidden layer both with and without dropout. Commensurate with the goal of this paper each model was run for 50 epochs and results were tabulated. Finally, the best model was trained longer for up to 200 epochs but with an early halting criteria built in to avoid over-fitting and wasteful resource consumption when there weren't any tangible improvements. Pytorch nn module was used to design the model. Cross entropy loss was used as an error or loss measure.

Since input feature were hand engineered by experts as float values within a range of 0-15 no adjustments such as normalisation and scaling were need on the features. However, the categorical values of classes had to be converted to integer values to calculate losses. This was done by simply using the corresponding ASCII values of the letters.

2.1.2 Model and data pre-processing for ASL alphabet dataset:

The neural network model used in this case is the famous VGG16 model with weights pre-trained on the ImageNet dataset. The model has an input size of 64x64 with 3 channels i.e. a resulting size of 3x64x64. The model has two sections namely: "features" and "classifiers". The features section comprises a series of convolution, ReLU and maxpooling layers with the final output being a set of 25088 features. The first convolution layer used a stride of 1 in both x and y direction and had 3 filters initially. A padding of 1 was also needed on both sides of the image. The subsequent classifier section uses these 25088 features to score each image against all 29 classes. Finally, cross entropy or softmax is used to choose the best fitting class as the prediction.

The first section of the VGG16 was used as a feature extractor. The output of 25088 features for each of the images was saved onto a file to be used as the input dataset for the next section of the VGG16 model i.e. the classifier. Since the goal of this paper is to investigate BRD technique the feature extractor (the CNN) not trained at all on the ASL alphabets images. Only the classifier was trained on the images via the extracted features. This technique of using transfer learning is called "ConvNet as fixed feature extractor" [4].

2.2 Model Measures

The datasets used is well balanced with comparable representation by each of the 26 letters and 3 symbols. In such an evenly distributed dataset, test accuracy is a great measure. However, plotting the confusion matrix also allows to see exactly in which cases the model errs. Unfortunately, the letters dataset didn't include the original images of the letters so further visual analysis wasn't possible. As an aside, it will be interesting to get access to the raw images to find out the cases where the model gets it wrong and also to see if the outliers detected by

BDR will also be considered as outlier by humans. For the ASL dataset it was possible to look at some of the actual images where the model got it wrong.

2.3 Model training

2.3.1 Training for Letters dataset

The model was trained using back propagation algorithm implemented by pytorch using autograd technique. Adam optimiser [3] was used to update weights of the network as it maintains per parameter learning rate and is one of the most commonly used in recent papers. The model was trained to identify the target letters encoded as their corresponding ASCII values from the 16 input signals which were also integer values. The target class with the largest probability was chosen as the predicted target. After a configurable number of epochs, a BDR was executed which removed outliers and the training continued. The most interesting aspect of the training was to figure out a halting criteria. Without a stopping criteria eventually all samples would be removed from the training set. Moreover, much before that, the model would overfit due to too few examples being in the training set.

Stopping criteria for training is an interesting topic. It is not straightforward to determine when the model should stop training. In the BDR paper training was halted when variance fell below 0.01. However, in case of the letters dataset the variance never fell to 0.01 so an alternative criterion to halt training was required as discussed in subsequent sections.

The letters dataset had 20,000 examples; out of which 16,000 were used as training set and the next 2000 as validation set. The last 2000 was kept aside as test set only used once to ascertain the final accuracy of the model. In each epoch the 16,000 examples were presented to the neural network randomly and in mini-batches. The size of the mini batches was configurable being yet another hyper parameter.

2.3.2 Training for ASL alphabet dataset using VGG16

Training for this dataset was done in two distinct phases. Firstly, the feature extractor of VGG16 model was used to extract 25088 features for each image. Since the CNN accepted a 64x64 image with RGB channels the raw ASL images we resized and cropped into 64x64 size before being feed into the VGG16 CNN. Since I used the CNN as a fixed feature extractor no back propagation and weight adjustments were done for the feature extractor section of the model. The output vector with size N x 25088 was saved onto a file to be later used as the input to the next section of the model which acts as the classifier.

In this second phase the features are loaded from a file which had 7830 samples. Out of these 80% i.e 6264 were used for training, the next 10% i.e 783 were used for validation while the final 10% was kept aside for the final test only to be used once. Similar to the earlier dataset samples were fed to the network during training randomly in mini-batches and a similar halting criteria was used. Experiments were done with both SGD and Adam optimiser. It was observed that SGD performed significant poorly and typically taking too long to achieve decent accuracies.

2.3.3 Training with BDR technique:

The BDR technique is based on the postulate that losses exhibit a bimodal distribution quite early on during the training process. This axiom was definitely upheld in both the letters and ASL alphabet datasets exhibiting two peaks of loss values. In the case of ASL alphabet dataset this phenomenon was more clearly pronounced as shown in figure 5. The first modal peak can be easily attributed to training samples which were easiest for the network to learn and correctly classify. The second modal comprised of training samples which are potential outliers. The ones in the middle are termed as slow coaches which the network might eventual learn to correctly classify. To allow the network sufficient time to learn the slow coaches BDR is applied every ß epoch; ß being yet another hyper parameter. The essence of BDR technique is to allow the network to identify outliers intrinsically without any external actor determining outliers.

However, implementing BDR might be a challenge because its expensive to identify a bimodal error distribution during training. Hence variance of the losses is used instead to achieve the same results. Firstly, the mean of all losses δ_ts is calculated. Due to the dominance of the low error peak (Figure 5, epoch 58) δ_ts will be very low, but greater than nearly all errors in the low error peak (due to the presence of the high error peak). Those patterns in the low error peak are not outlier candidates. In order to isolate potential outliers, all those patterns with error greater than δ_ts are taken from the training set. The dominance of the outliers in the subset will skew the distribution towards the outliers. The mean δ_ss and standard deviation ó_ss of this skewed

these two statistics is possible to decides which patterns to permanently remove from the training set. Those patterns with error ≥ δ_ss + α ó_ss where 0 ≤ α ≤ 1 are removed.

2.3.4 Training with adaptive BDR technique

Adaptive BDR introduced in this paper proposes that α be made to dynamically adapt depending on the epoch number. This paper uses a simple way to make α adaptive by dividing α by the epoch number.

αeffective = α static ⁄ epoch number

The reasoning is that as the training progresses the outlier peak will shrink, i.e. have a lower variance so α also needs to decrease. Extensive exploration with various settings of adaptive BDR could not be performed due to lack of time, hence this paper proposes that this be investigated in future work.

3 Results and Discussion

As mentioned in the BDR paper[1] the distribution of loss takes a normal form initially. However, the network quickly learns most of the example and the mode of the loss distribution shifts heavily towards zero. At that point a longer tail of the distribution is observed with a slight bulge at the right end. This is characteristic of the bimodal distribution as mentioned in the BDR paper. In case of the letter dataset the bimodal distribution isn't very evident because of the very low magnitude of the second peak. Fig 3 below shows the loss distribution at the first epoch and after a few epochs. It is clearly visible how the loss distribution shifted towards zero and the formation of slow coaches[2] and outliers are also evident.

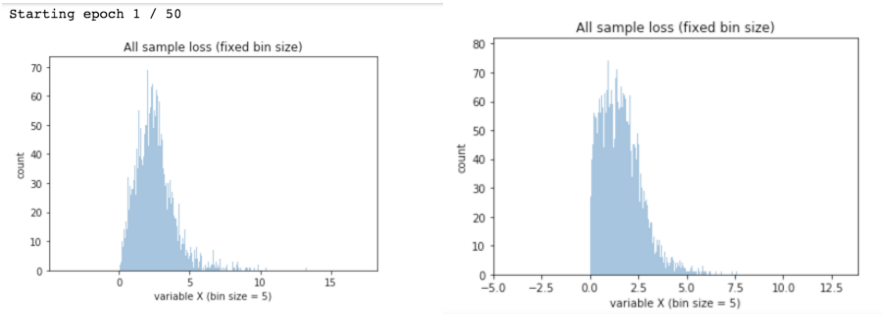


Fig. 4. Plot of the distribution of loss at first and second epoch.

However, the ASL alphabet dataset exhibited typical bimodal distribution quite early in the training epochs as shown in figures below. These results intentionally are without applying BDR so that the bimodal distribution could be observed first before removing outliers.

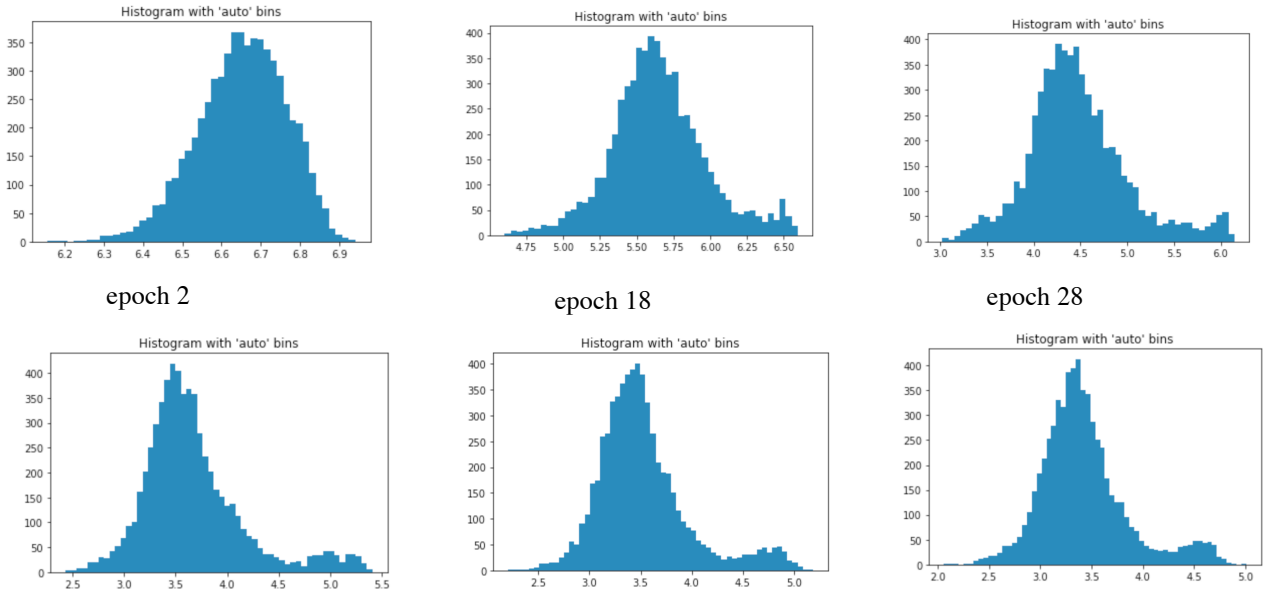
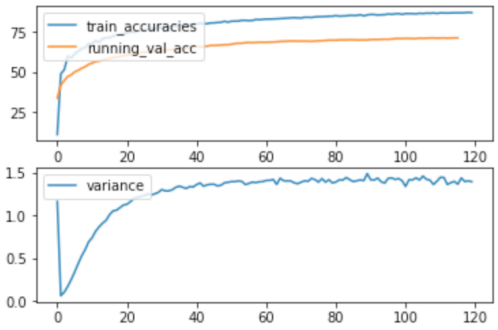
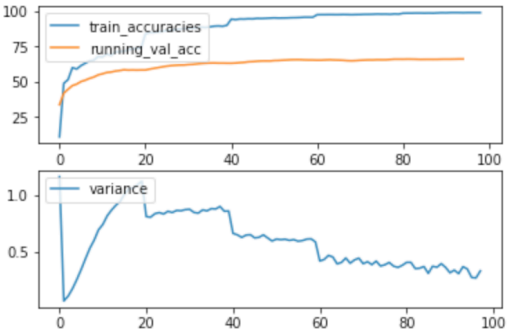


Fig. 5. Plot of the distribution of loss at first and second epoch.

Several inferences can be made from these loss distributions. Its clearly evident that the mean and mode decreases as the epochs progress: this is a sign that the network is making more correct predictions. Moreover, the variance decreasing progressively especially during the later epochs is a sign that the network is becoming more confident in its predictions. Both of the above observations are also confirmed by the plot of accuracies and variances as shown below.



Accuracies and variance without BDR



Accuracies and variance with BDR

Fig. 6. Top subplot: training accuracy and validation accuracy on y-axis with epoch number on x-axis. Bottom subplot: loss variance on y-axis with epoch number of x-axis

After the initial trainings without BDR a fresh experiment while applying BRD was conducted. BDR was carried out every 20 epochs so that the network had enough time to learn slow-coach patterns. A more aggressive removal might remove examples too soon so 20 epochs was chosen. This is another hyper parameter which can be tuned and ideal be determined by the network itself. This could be a further area of study.

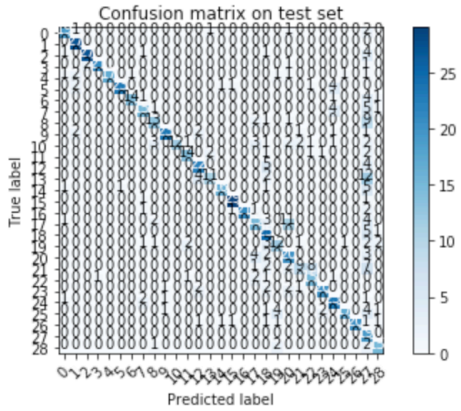
As shown in fig 6 (above) after the first outlier removal at the 20th epoch there was a significant jump in training accuracy and a marginal but clearly evident increase in validation accuracy shortly following the removal. Correspondingly, there was a sharp drop in variance of loss after the first episode outlier removal. This clearly indicates that outlier removal has been effective. However, the impact of outlier removal is not profound on subsequent episodes and in fact the impact progressively diminishes. Also, notable improvement in validation accuracy is almost non-existent in subsequent removals. Training accuracies have almost hit 100% (99.06%) when applying BDR as compared to 87.26% without BDR. Another significant difference is that variance has dropped with every BDR but it was constant in case of training without BDR removal.

As shown in fig 6 the accuracies (both training and validation) have not significantly improved after the first outlier removal. A reasonable halting criteria is necessary to determine when to stop training. Usual techniques are to stop when validation accuracy decreases or loss increases however this technique is not robust enough since it might be the case that the model got stuck in a local minimum which it will eventually emerge from to find a lower minimum. Another approach would be to continue training for as long as we have time and resources available while monitoring for over-fitting.

In this experiment a slight variation of drop in validation accuracy to monitor over-fitting has been used. A running mean of last 5 validation accuracies was maintained. If the running average didn't improve by at least 0.10 for 10 consecutive epochs, both numbers being hyper parameters, training was halted. This criterion also suffers from the same problem of being stuck in a local minimum but is better than looking at only the last validation accuracy. During my experiments such a 10 consecutive non-improvement occurred during the 178th epoch.

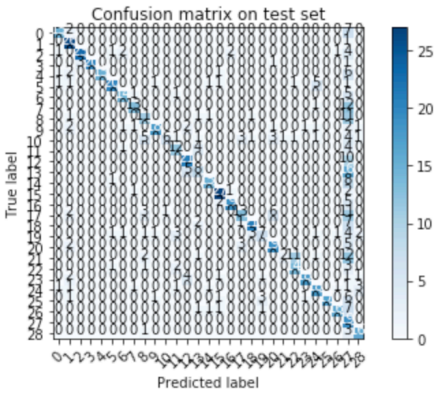
Finally, the confusion matrix and accuracies on the test set for both the variants: with and without BDR were plotted as shown in the figures below. Superficially it appears that BDR seems to have adversely affected generalization of the model; i.e. the model has over fitted to the training set. This is one of the dangers of BDR which was noted by Slade and Gedeon in the original BDR paper [1].

As shown in figure 7 below, the diagonals of the confusion matrices are dark while the other areas are light: this is a characteristic of a model doing quite good with the predictions. However, there are some notable areas where the models have not performed well enough. Such dark regions outside the diagonal are more prominent in the model with BDR. This is most likely due to over fitting which resulted due to removal of non outliers. This can happen if the outliers are not representative of the noise in the entire dataset. Due to lack of resources (GPU and time) training was restricted only to 7830 samples out of the entire set of 87000 images. Surely this small proportion of the dataset was not representative of the typical real outliers (or noise); as a result, non-outliers were removed and the network failed to generalize to the test set. This could explain why BDR didn't improve test set accuracy for the ASL alphabet dataset while it did for the letters dataset as described below.



Confusion matrix without BDR

```
****Final check on test set*****
Got 534 / 783 correct (68.20)
```



Confusion matrix with BDR

```
****Final check on test set*****
Got 463 / 783 correct (59.13)
```

Fig. 7. Confusion matrices and final test accuracy results

As per as accuracy on test set goes the highest accuracy of 68.20% was achieved with no BDR. The next best accuracy of 65.26% was achieved with BDR every 40 epochs. Finally, with BDR every 20 epochs the test set accuracy was 59.13%. These results show that BDR is removing training example which are not really outliers because such samples are present in the test set as well, hence the test set accuracy is decreasing.

Most recent results [7] on this dataset is at about 91% validation accuracy. However, such results have been achieved by training on the entire dataset using GPUs. Given that in this paper only 10% of the entire dataset was used results of around 68% is not too poor.

Results on letters dataset:

The model was trained for a maximum of 200 epochs with adaptive BDR and a halting criterion of no more than 0.1 increase in validation accuracy for 10 consecutive epochs. In this model BDR was applied every 50 epochs instead of 10 as was used earlier for the tabular results. The model achieved testing set accuracy of 78.59% which is about 1% points more than the model without BDR. Moreover, the confusion matrix in this case is significant better than the ones for ASL alphabet dataset with few dark areas lying outside the diagonal.

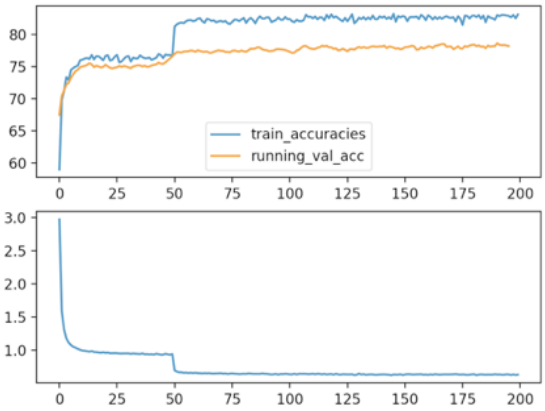


Fig. 8. Top subplot: training accuracy and validation accuracy on y-axis with epoch number on x-axis. The validation accuracy curve is smoother because it uses running average of last 5 accuracies.
Bottom subplot: loss variance on y-axis with epoch number of x-axis

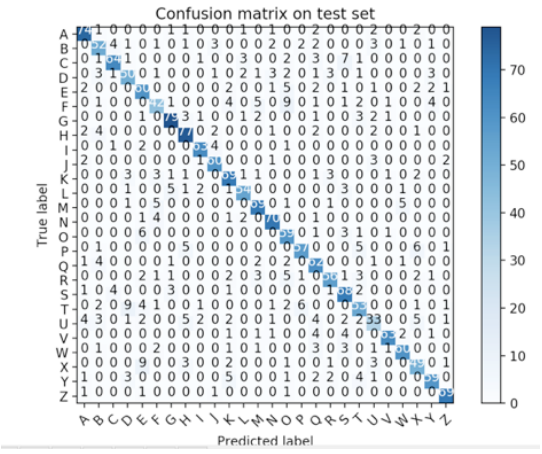


Fig. 9. Confusion matrix Y-axis has true labels while x-axis has the labels predicted by the model.

4 Conclusion and Future Work

Having explored the datasets and trained the neural network with various hyper parameters with and without BDR we can confidently conclude that BDR absolutely helps in improving train, validation and test set accuracies if the outliers training set is representative of the test set. This indicates that BDR is very sensitive and care needs to be exercised when applying BDR on a relatively small dataset because the small dataset may not represent outliers appropriately. Adaptive BDR which was introduced in this paper needs to be explored further as it showed promise in improving accuracies and reducing over-fitting under BDR. Another area of further exploration should be the hyper parameter which determines how frequently outlier removal episodes are run.

Due to limitations of time and GPU recourses it wasn't feasible to run feature extraction and BDR on the entire dataset of 87000 images. It is highly recommended that this is done in a future study because BDR has shown great benefits when applied to the entire dataset like the letters dataset. And it can be hypothesised that BDR will prove to be beneficial when applied to all 87000 images. This paper proves that modern neural network techniques like CNN and transfer learning can be applied on the raw images to extract features while excellent techniques from the 90's like BDR can be used to train the following classifier.

References

1. Slade, P and Gedeon, TD "Bimodal Distribution Removal," Proceedings IWANN International Conference on Neural Networks, Barcelona, 1993. also is Mira, J, Cabestany, J and Prieto, A, New Trends in Neural Computation, pp. 249-254, Springer Verlag, Lecture Notes in Computer Science, vol. 686, 1993.
2. Gedeon, TD and Harris, D, "Network Reduction Techniques," Proc. Int. Conf. on Neural Networks Methodologies and Applications, San Diego, vol. 2, pp. 25-34, 1991.
3. Diederik P. Kingma, Jimmy Lei Ba ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION
4. Andrej Karpathy, http://cs231n.github.io/transfer-learning/
5. Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson : How transferable are features in deep neural networks? https://arxiv.org/abs/1411.1792
6. Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition, https://arxiv.org/abs/1409.1556
7. https://www.kaggle.com/paultimothymooney/interpret-sign-language-with-deep-learning