T1-03 Recreating and Improving Baseline of Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Shah Tausif Iqbal
260515282
shah.iqbal@mail.mcgill.ca

Dongjoon Lim
260587899
dong.lim@mail.mcgill.ca

Wei Zheng
260829004
wei.zheng2@@mail.mcgill.ca

I. INTRODUCTION

Despite its high dependence on GPU and ambiguity of the Black Box hidden layers, most of the recently published papers are based on how good deep neural networks are performing. However, less complex machine learning algorithms such as SVM, Random Forest are overlooked as their performances are considered to be inferior(LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton 2015). In our paper, we will apply simple but still powerful Machine Learning algorithm baselines for performing the same tasks performed in the paper Dropout: a simple way to prevent neural networks from overfitting.(Srivastava, Nitish, et al 2014) and aim to attain similar or better performance by extensive hyperparameter tuning of these algorithms.

II. TASKS

In Srivastava's paper (Srivastava, Nitish, et al 2014), dropout was introduced to reduce overfitting of neural networks. Our first task was to reconstruct this paper using baselines and hyperparameters they selected. Out of 7 datasets that were worked with, we have chosen to work with MNIST, SVHN, and Reuters-RCV1.

Our second task was to implement several simple machine learning algorithms on the same dataset and to perform extensive hyperparameter tuning of these algorithms to beat the performance results of the reconstruction. After getting our results, we discussed the differences between the performance of the paper's baseline and our simple machine learning baselines. Also, discussions on how some simple machine learning baselines performed better than others and what made these algorithms perform better were done.

III. METHODOLOGY

A. MNIST

1) Reconstruction: To reconstruct the baseline implemented in the paper, tensorflow framework was used. Preprocessing was not done to the MNIST images in the paper so raw MNIST images were used for the reconstruction as well. Built-in dropout function in tensorflow was used to implement dropout rate of each layer in neural networks and all the experiments' hyperparameters were identically set as the paper. However, for some nonclear descriptions of models in the paper, hyperparameters were selected to maximize the similarity of our reconstruction to the paper.

2) Improving the Baseline: Simple machine learning baselines were implemented using scikit-learn's built-in algorithms. Preprocessing techniques such as PCA dimensionality reduction and HoG feature tracker were done to the dataset to see if they help to improve the performance of learning. Hyperparameter tuning of these baselines was done through built-in gridSearchCV function. Only the best performing hyperparameters are reported in the result section.

B. SVHN

The data matrix was given in the form (N, w, h, d) where N was the number of samples, h is the height, w being the width and d being the depth. The total number of training images were 73257 and the total number of test images were 26032.

As for the training, validation, test split - 63733 were given to the training set, 9524 to the validation set and 26032 to the test set.

1) Preprocessing: Two methods of preprocessing were used for fitting the model as shown in the paper. The first method was dividing all the data by a certain number(255). Another method used was applying normalization to all the training data in the training set for a zero mean and unit variance.

Preprocessing Simple Machine learning Algorithms The images had to be reshaped to a 1-dimensional array for simple machine learning languages such as SVM, KMeans, Random Forest to fit. HoG feature tracker was another preprocessing technique used to change the training data to see an increase in performance. Auto encoding techniques were used to encode images in this process without much improvement in performance.

2) Reconstruction: Reconstruction of the baselines and the algorithms used in the paper was done through the help of many external machine learning python libraries namely Keras and SciKit Learn. The Dropout method, what the paper is about, is already implemented in Keras. Extensive hyperparameter tuning was done to get the best results. Although the paper does not mention a few hyperparameters that were needed for the results, a convolution neural network with batch normalization, max pooling after every layer dropout gave us results very close to the paper as shown in the results section below.

3) Improving the Baseline: Simple machine learning algorithms such as Support Vector Machines, KMeans, Random Forest, K Nearest Neighbours, XGB Classifiers and Gradient Boosting Classifiers were used for our simple machine

learning algorithms. Extensive hyperparameter tuning was executed in which the best-performing ones are mentioned.

C. Reuters-RCV1

1) Preprocessing: The feature matrix is a scipy CSR sparse matrix, with 804414 samples and 47236 features. Non-zero values contains cosine-normalized, log TF-IDF vectors. It is categorized into 103 classes. These classes are arranged in a tree hierarchy.

As the original paper mentioned, the subset of Reuters was created by 402,738 articles and a vocabulary of 2000 words comprising of 50 categories in which each document belongs to exactly one class. So we have to convert the multi-label classification to the multi-class classification. As the 50 classes were not specified, we manually picked 50 classes with 402946 documents that are the closest selection from our preprocessing process. And 2000 words were selected by the most frequent features(non-zero in the documents).

2) Training/Validation/Test split: 3-fold cross-validation is used to determine the best model parameters. Split the 400K data into 50% of the training set, 50% of the test set. Then split a validation set taking up 20% data from the training set. The validation set is used to tune the hyper-parameter by GridSerch, record the accuracy to determine the best model hyperparameters.Fit the trained model by the combination of training and validation set. The remaining 50% test set is used to compute accuracy.

3) Reconstruction and improving the Baseline: Both neural network and simple machine learning baselines used the subset after preprocessing. As the structure of neural networks was not clear in the paper, by using Tensorflow framework we tried the similar structures that the author used in the other dataset with the best range of dropout rate recommended. Simple machine learning baselines were created through Scikit learn library. Hyperparameter tuning of these baselines was done through built-in gridSearchCV function.

IV. RESULTS

A. MNIST

1) Recreating Results: 6 different structures of simple fully connected neural networks were trained to classify the MNIST dataset. The dropout rate of 0.8 was implemented to the input layer and 0.5 was implemented to all the other hidden layers. The best performing structure of neural network was 2 layered with 4096 units activated by ReLU and regularized with max-norm constraint with the classification accuracy of 98.75%. In general, we can observe that all the neural network structures had classification accuracy above 98% so they performed well on the MNIST dataset.

Accuracies of 6 different structures of neural network architecture are reported in below table 1.

TABLE I
RECONSTRUCTION OF NEURAL NETWORK

Model	Activation	Accuracy%
Dropout NN (Logistic)	3 layers, 1024 units	98.20
Dropout NN (ReLU)	3 layers, 1024 units	98.01
Dropout NN + max-norm constraint	3 layers, 1024 units	98.53
Dropout NN + max-norm constraint	3 layers, 2048 units	98.60
Dropout NN + max-norm constraint	2 layers, 4096 units	98.75
Dropout NN + max-norm constraint	2 layers, 8192 units	98.25

2) Improving baseline: Accuracy report of different kinds of simple machine learning algorithms. Dimensionality reduction using PCA and feature tracking HoG was also performed for preprocessing.

TABLE II
SIMPLE ALGORITHMS WITH RAW DATA

Model	Hyperparam	Accuracy%
LinearSVM	C=5, gamma=0.05	98.47
kernelizedSVM	degree=7, gamma=0.05	97.9
KNN	n-neighbors=3	97.14
Random Forest	n-est=200, max-depth=7	96.03
SimpleVoting of best3 above	None	97.89

TABLE III
REDUCED FEATURE TO 100 USING PCA

Model	Hyperparam	Accuracy%
LinearSVM	C=5, gamma=0.05	92.37
kernelizedSVM	degree=7, gamma=0.05	91.57
KNN	n-neighbors=3	90.10
Random Forest	n-est=200, max-depth=2	79.69
SimpleVoting of best3 above	None	93.27

TABLE IV
DATA PREPROCESSED USING HOG

Model	Hyperparam	Accuracy%
LinearSVM	C=5, gamma=0.05	96.16
KNN	n-neighbors=3	92.53
Random Forest	n-est=200, max-depth=7	77.89
SimpleVoting of best3 above	None	92.95

The best performance of simple machine learning algorithms was achieved when training was done using raw data. Apparently, preprocessing of dimensionality reduction using PCA or HoG feature tracker worsened the performance of simple algorithms. Among simple machine learning algorithms, linear SVM classifier with C=5, and gamma=0.05 performed the best with the classification accuracy of 98.47%.

B. SVHN

1) Recreation: As for recreating the results found and described in the paper, we came very close to what was achieved. The results are shown below

TABLE V

RECONSTRUCTION OF NEURAL NETWORK SVHN

$Model$	Accuracy%
Conv NN + Max Pooling	94.10
Conv NN + Dropout	94.51
Conv NN + Dropout in All Layers	95.71

A 3 layer convolution neural network was built with max pooling, max pooling and dropout in fully connected layers and max-pooling and dropout in all layers.
As for all convolution neural networks, we did hyperparameter tuning of stride, padding, activation functions. Batch size(128) and epochs(120) were kept the same when running these algorithms. An optimizer of stochastic gradient descent was used for every CNN compared to adam which gave slightly less accuracy.

As for the first result, max norm regularization was used on the CNN. Then, after adding batch normalization before the activation function of ReLu and then MaxPooling, an accuracy of 94.10% was achieved.

As for the second result, max norm regularization was used on the CNN. Then, after adding batch normalization before the activation function of ReLu and then MaxPooling, an accuracy of 94.51% was achieved. Dropout was applied to this layer on every fully connected layer where dropout of 50% was applied to the first layer, 30% to the second and 30% to the third.

The only difference between the third and the second layer is that dropout was applied to all the layers. This increased the accuracy to 95.71%

2) Simple Machine Learning Algorithms: Here we have applied simple Machine learning algorithms. Preprocessing techniques, as explained before includes HoG feature tracking and autoencoders. When using normal training set, the results were significantly low such that preprocessing techniques had to be put in place. First, the training sets were normalized to have zero mean and unit variance. Secondly, we applied HoG features tracking on all the validation, training and test sets. For this we set the pixels per cell to be (16, 16), added 8 orientations, cells per block to be (2,2) and block norm to be L2 loss. The results after applying this preprocessing technique to different machine learning models are shown below in this table:

TABLE VI

SIMPLE MACHINE LEARNING MODEL HYPERPARAMS

$Model$	HyperParameters%
kernalized SVM	kernel=poly, degree=7, C=0.1
Random Forest	estimators = 200
kNN	n = 40
KMeans	n_clusters = 10
XGB Classifiers	
Gradient Boosting Classifiers	estimators=100, learning rate=1.0

TABLE VII

SIMPLE MACHINE LEARNING MODEL HYPERPARAMS

$Model$	Accuracy%
kernalized SVM	55.20
Random Forest	59.51
kNN	56.48
KMeans	15
XGB Classifiers	58.55
Gradient Boosting Classifiers	55.1

The best performance achieved on the test set is thus the random forest where the hyperparameter set was the $numberofestimators = 200$

C. Reuters RCV1

1) Recreating Results: 3 pairs of simple fully connected neural networks (implementing different dropout rates or no dropout) were constructed for Reuters RCV1 dataset. When the dropout rate was set to 0.5 to the input and hidden layers with Relu activation, the neural network with every 4096 units in the 2 hidden layers achieved the best performance in 90.46%. By contrast, the same NN structure without implementing dropout method achieved similar result slightly decreasing down to 90%. Accuracy of different kinds of neural network architecture are listed in Table 5 below

TABLE VIII

RECONSTRUCTION OF NEURAL NETWORK

$Model$	Activation	Accuracy%
Dropout(p=0.2) NN (ReLU)	3 layers, 3328 units	90.12
NN (ReLU)	3 layers, 3328 units	89.67
Dropout(p=0.3) NN (Relu)	2 layers, 2048 units	90.07
NN (Relu)	2 layers, 2048 units	89.72
Dropout(p=0.5) NN (Relu)	2 layers, 4096 units	90.46
NN(Relu)	2 layers, 4096 units	90.00

2) Improving baseline: Accuracy report of different kinds of simple machine learning algorithms.

TABLE IX

SIMPLE ALGORITHMS WITH RAW DATA

$Model$	Hyperparam	Accuracy%
LinearSVM	C=1	87.77
kernelized SVM	C=10,gamma=1,kernel='rbf'	90.769
KNN	n-neighbors=5	70.714
Random Forest	n-est=200, max-depth=150	86.98
Logistic regression	Solver='saga'	89.27

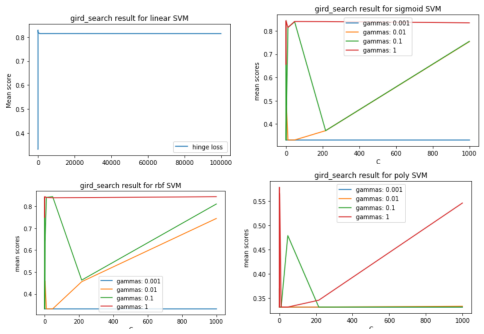

Fig. 1. Hyperparameter tuning result in SVM

The Fig.4 shows the gridSearch result collected from SVM. By finetuning the hyperparameter of different simple machine learning algorithms, the best performance we got within all the algorithms was from the kernelized SVM (C=10,gamma=1,kernel: rbf) with 90.769% classification accuracy.

V. CONCLUSION

A. MNIST

The best performing simple machine learning algorithm was linear SVM classifier with a 98.47% classification accuracy. This algorithm is comparable to the neural networks since this classification accuracy is better than 3 of the neural network structures we reconstructed. The best performing hyperparameter was C=5 and gamma=0.05 for this algorithm.

The hyperparameter C is the parameter for the soft margin cost function, which controls the influence of each individual support vector. In a nutshell, if we have a high value of C, the stricter the classification becomes, then the margin of the SVM will become smaller. Gamma is the free parameter of the Gaussian radial basis function. Thus when gamma is large, then the variance is small implying the support vector does not have wide-spread influence resulting in high bias but low variance.

For the case of MNIST, the selected C value was relatively large and gamma was relatively small for better performance. Unlike other image classification problems like CIFAR-10 or SVHN where they have 3 RGB channels, MNIST only has a single intensity channel where the values are ranging from 0 to 255. Also, as MNIST is a set of handwritten digits constrained in 28X28 pixels meaning that we have the desired form of how each digit should look like. In other words, MNIST dataset classification should focus more on reducing variance than reducing bias. As a result, we got these hyperparameters that somewhat strictly penalizes misclassification while having a small variance. In other words, with a relatively small dataset like MNIST, it is crucial to choose classifiers that can regulate overfitting flexibly. This is the reason SVM performed the best while random forest and KNN were prone to overfitting.

However, preprocessing of the dataset worsened the performance of the classification. Dimensionality reduction using PCA was performed with many different numbers of features, but all of them did not perform as well as training algorithms using raw dataset. One reason for this is because PCA works well on features that are dependent or correlated to each other(Cao, L. J., et al 2003). MNIST is an image set so each feature of this dataset is an integer intensity value of a certain pixel which we cannot assume they are dependent on each other. Moreover, MNIST dataset is already simple that most of the features in each image are 0 representing white pixels and few of the features have a positive value to constitute the handwritten number. Applying PCA on this already simple dataset would not help the performance, instead one can end up losing too much information. Applying HoG to the dataset also did not help with the classification performance. HoG uses histogram based approach to find the weighted sum of the gradients of the image(Lu, Wei-Lwun, and James J. Little 2006). This can help complex images to simplify salient features, but again MNIST is already a simple enough dataset that applying HoG will only result in information loss.

In conclusion, if classifying MNIST with neural networks is not available due to some hardware limits, we could consider SVM classifiers for classifying raw MNIST as a good substitute given that right hyperparameters are chosen.

B. SVHN

a The limitations of the accuracy we had was not enough information on how to preprocess to run KMeans on the SVHN dataset. KMeans is a very simple clustering machine learning model but it requires clusters of data, which need to be preprocessed. If more information was received on the KMeans, then accuracies of up to 90% could have been achieved.

Apart from the training set and the testing set, there were more than 500K examples in an extra set which could be used for training. Due to time constraints, this extra set was not used but it would have increased the accuracy of the models.

C. Reuters RCV1

(1)The best performing simple machine learning algorithm was kernelized SVM classifier with 90.769%. This algorithm is better than the best neural network structures we reconstructed with 90.46% accuracy. The best performing hyperparameter was C=10,gamma=1 and with Gaussian radial basis function for this algorithm.

(2)SVM is one of the Non-parametric methods which have a potentially infinite number of parameters, where the number of support vectors depends on the dataset complexity. SVM is flexible and powerful which require fewer assumptions. C and Gamma are the parameters for a nonlinear support vector machine (SVM) with a Gaussian radial basis function kernel. The parameter C trades off misclassification of training examples against the simplicity of the decision surface. For a large value of C, the optimization will choose a smaller-margin hyperplane. The hyperplane does a better job

of getting all the training classified correctly, a large C gives you low bias and high variance, vice versa. Gamma is the free parameter of the Gaussian radial basis function controlling the tradeoff between error due to bias and variance in our model. small gamma will give you low bias and high variance while a large gamma will give you higher bias and low variance.

In the Reuter RCV1 dataset, we have more than 200K training examples with 50 classes, which is a huge dataset providing help on avoiding overfitting. So we are able to handle the model with relatively high variance and low biases. By the chosen relative large C, the cost of misclassification was penalized a lot, which means the bias is reduced. And with the proper magnitude of gamma, the variance will not be too high resulting overfitting.

(3)We tried many network architectures and found that dropout gave improvements in classification accuracy over all of them. However, the improvement was not as significant as that for the image data sets. This might be explained by the fact that this data set is quite big (more than 200,000 training examples) and overfitting is not a very serious problem.

VI. STATEMENT OF CONTRIBUTIONS

Shah worked on the SVHN dataset. Provided the template for the final report, and wrote the sections of the final report corresponding to these tasks

Dongjoon worked on the MNIST dataset and shared the workload of the Reuters RCV1 dataset. Wrote the sections of the final report corresponding to these tasks

Wei worked on the Reuters RCV1 dataset and wrote the sections of the final report corresponding to these tasks

We hereby state that all the work presented in this report is that of the authors.

REFERENCES

[1] Srivastava, Nitish, et al. *Dropout: a simple way to prevent neural networks from overfitting.* The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[2] Lewis, D. D., et al. *RCV1: A New Benchmark Collection for Text Categorization Research.*The Journal of Machine Learning Research : JMLR. (2005).5(1): 361-398.

[3] Lu, Wei-Lwun, and James J. Little. *"Simultaneous tracking and action recognition using the pca-hog descriptor."* null. IEEE, 2006.

[4] Cao, L. J., et al. *"A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine."* Neurocomputing 55.1-2 (2003): 321-336.

[5] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436.