Improvements to Feed-Forward Neural Networks Used to Classify Forest Type Covers Based on Cartographic Features

Ronald Picard
Vanderbilt University
Nashville, TN, USA

***Abstract*—As computational power increases, deep neural networks provide an empirically effective way to classify large data sets based on a small labeled subset. Forest type covers are an example application of were these networks can be used. It is imperative for private and public land managements agency's to have accurate records of forest type covers for the regions under their management. However, due to legal and manpower limitations, this is not always possible. Neural networks provide a mechanism by which this data may be obtained. We present an improvement on feed-forward neural network architectures used to classify forest type covers based on cartographic features.**

***Index Terms*—Machine Learning, Neural Networks, Tesorflow, Keras, Cartography, Forest Type Covers**

I. Introduction

As the computation power of processors and accelerated GPUs have increased, deep neural networks have seen a major increase in popularity. Deep neural networks provide increased capacity which allows more complex functions to a approximated, and as a result, large data sets to be handled ease. We explore improvements to past work [1] on the prediction of forest type cover based on cartographic variables using neural networks. The networks we explore are deeper than previously studied for this application. We empirically show the utility of deep networks and the increased classification accuracy associated with them.

II. Problem Background

In this section we introduce some fundamental topics required to understand the different sections of this work.

A. Basics

1) Forest Type Cover: The type cover of a forest refers to the type of forest cover that grows on a particular geographic are. It is important for both private and public land management agency's to have accurate records of the forest type covers that are present on particular areas under their governance. Generally, these agencies must collect this cartographic feature by estimating remotely sensed data or via field agents who directly record the data. This can lead to high cost for some agency's, and in some cases, the data cannot be recorded due to the legalities surrounding the land. This results in a need for accurate predictive models to obtain the data. [1]

2) Machine Learning: Machine learning algorithms involve three primary elements; an experience E, tasks T, and performance measure P. According to Michell (1997) A computer program is said to learn from experience E with respect to some tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with E." [2]

3) Feed-Forward Neural Networks: Feed-Forward neural networks are complex mathematical functions made up of structured layers of neurons (units). Each neural network involves a linear transformation followed by a non-linear activation function. Every network has an input layer, output layer, and hidden layers.

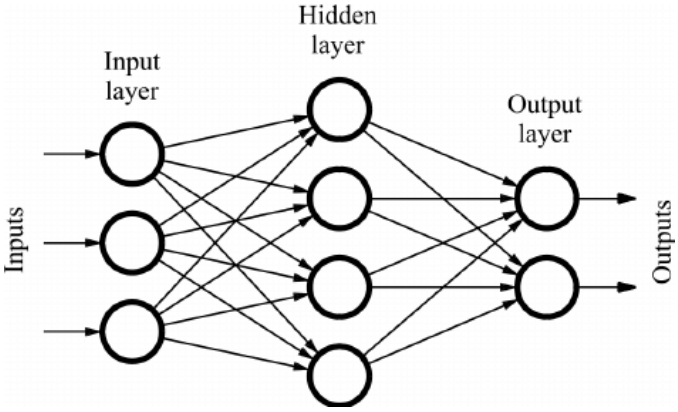

Fig. 1: Diagram of a Feed-Forward Neural Network [3]

4) Training Neural Networks: In order to train neural networks you need three things; a task T, a performance measure P, and an experience E. The task is for the network to predict the truth label based on the input (classification). This is known as forward propagation. The performance measure is known as the loss function. The loss function provides a cost value that neural networks attempt to minimize while training. This function is dependent on the task of the neural network and represents how different the predicted results of the network are from the truth labels. The experience is the technique that is utilized to update the weights and bias's based on the loss function. This is known as backpropagation. The most common backpropagation technique is known as gradient descent. Gradient descent involves that calculation of the gradient (rates of change) of the lost function with

respect to the weights and bias's of the network. This gradient is then multiplied by the learning rate and subtracted from the weight's and bias's. The result is that during each iteration, the network moves closer and closer to the minima of the loss function. 2 illustrates the forward and backward propagation of a feed forward neural network: where W represents the weights of each layer, b represents the bias's of each layer, a is the activation function of each layer, L is the loss function, da is the gradient of the loss function with respect to the activation function, dz is the gradient of the loss function with respect to the linear function of each layer, dW is gradient of the loss function with respect to the weights of each layer, and db is the gradient of the lost function with respect the bias's of each layer.

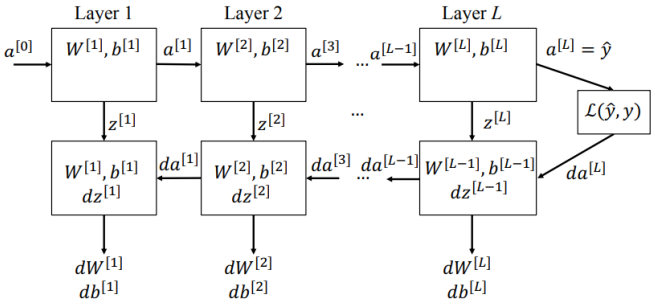

Fig. 2: Forward and Backward Propagation [4]

5) Classification: A common task among supervised neural networks is to classify data samples into categories. This task is known as classification. The neural networks do this by learning the weights and bias's needed to accurately predict which sample belongs to which category based on the samples features. This technique proves useful in the classification of a forest type cover based on cartographic data sample features. The loss function (performance measure) associated with multi-category classification is known as Softmax (1).

$$softmax(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)} \tag{1}$$

6) Data Sets: Validation, Training, Testing: Supervised neural networks involve the use of labeled data; that is, data that has a true label associated with it. In order find out if our neural network is accurately approximating the function of classifying forest type covers, we need to split the into three data sets; validation, training, and testing. The validation set is a small subset of the training data that is used to tune the hyper-parameters of the neural networks. After the hyper-parameters have been tuned, the training data is used train the network. After the network has been trained, the testing data is run through the network to see how accurately the network predicts unforeseen data.

7) Hyper-Parameters : The hyper-parameters of a neural network are parameters relating to the structure and algorithmic techniques of the neural network; as opposed to the weights and bias's of the neural network. Adjusting these parameters has a significant effect on how well the neural network performs. Listed below are some of the most important hyper-parameters.

- Learning Rate
- Number of Hidden Layers
- Number of Units Per Layer
- Type of Activation Function in the Hidden Layers

8) Regularization: Regularization techniques are any technique applied to a neural network in an attempt to prevent overfitting. Overfitting occurs when the network results in a low training cost (and high training accuracy), but a low test accuracy. Effective regularization techniques allow the network to generalize well; meaning it preforms well on the unforeseen data. Listed below are some common regularization techniques.

- Adaptive Moments (ADAM)
- Dropout
- L1 and L2 Regularization
- Early Stopping
- Noise Injection
- Data Set Augmentation
- Ensemble Methods
- Bootstrapping

9) Tensor Flow: Tensor Flow is a machine learning library provided by Google, inc. This library abstracts away low-level details of neural network implementations and supports many of the techniques and algorithms that are commonly used when training neural networks. In addition, Tensor Flow has Python language support.

10) Keras: Keras is a high-level API for Tensor Flow that further abstracts the details of the neural network implementations and provides an intuitive interface for constructing neural networks. In addition, Keras has Python language support.

III. Data Set

A. History

This data set represents a collection of 581,012 cartographic samples (30x30-m raster cells, encompassing approximately 129,213 acres) from four regions of Roosevelt National Forest in norther Colorado. [1] The regions are located approximately 70 miles northwest of Denver Colorado (see figure 3). These locations areas have had little direct human management. This means the type covers have resulted primarily from ecological processes. [1]

B. Features

The features of the samples in this data set were obtained from the US Geological Survey (USGS) and the US Forest Service (USFS). [1] The samples of this data set contains 54 features that are listed below. [5] The first 10 features are variables relating to measures of the samples, the next 4 features are binary variables relating to the absence or presence of a specific wilderness ares, and the final 40 features are binary variables relating to the absence or presence of specific

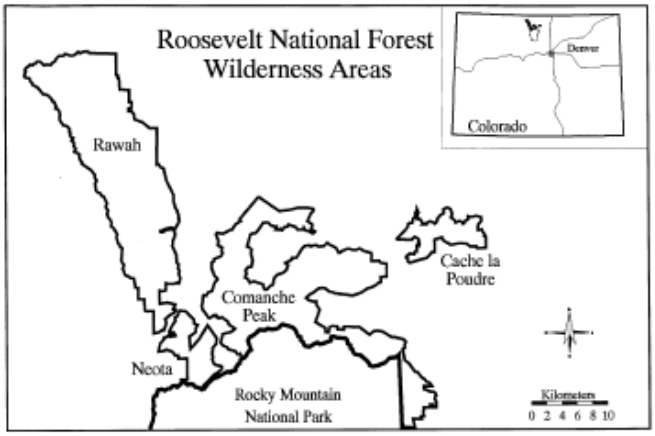

Fig. 3: Study Area Location Map [1]

soil types based on a combination climatic and geologic zones. [5]

1) Elevation (m)
2) Aspect (azimuth from true north)
3) Slope ()
4) Horizontal distance to nearest surface water feature (m)
5) Vertical distance to nearest surface water feature (m)
6) Horizontal distance to nearest roadway (m)
7) A relative measure of incident sunlight at 09:00 h on the summer solstice (index)
8) A relative measure of incident sunlight at noon on the summer solstice (index)
9) relative measure of incident sunlight at 15:00 h on the summer solstice (index)
10) Horizontal distance to nearest historic wildfire ignition point (m)
11) Wilderness area designation (four binary values, one for each wilderness area), and
12) Soil type designation (40 binary values, one for each soil type)
13) (11-14) Binary Wilderness Area: A
14) (15-54) Binary Soil Types: A

C. Labels

The data set is labeled by forest type cover; which is numerical integer from the set: 1-7. These numerical digits correspond the 7 type covers listed below. [1] These type cover classification represents the dominant tree species found in the wilderness areas. [1] There are other cover types in these areas; however, they only exist in small patches and have been excluded from this data set. [1]

1) Lodgepole Pine (Pinus Contorta)
2) Spruce Spruce: Fir (Picea Engelmannii and Abies Lasiocarpa)
3) Ponderosa Pine (Pinus Ponderosa)
4) Douglas-fir (Pseudotsuga Menziesii)
5) Aspen (Populustremuloides)
6) Cottonwood: Willow (Populus Angustifolia, Populus Deltoides, Salix Bebbiana, Salix Amygdaloides)
7) Krummholz: Primarily Engelmann Spruce (Picea Engelmannii), Subalpine Fir (Abies Lasiocarpa), and Rocky Mountain Bristlecone Pine (Pinus Aristata)

IV. Preprocessing

The data set requires preprocessing before the training of a neural network may occur. IV-A describes how we normalize the data, IV-B describes how we encode the truth labels for this multi-classification problem, and IV-C describes how we separate our data into the validation, training, and testing sets.

A. Data Normalization

The first 10 data features comes in various forms of magnitude. We need to normalize this data so that no single feature is favored in neural network because of magnitude. The last 44 features are already in binary form. We will leave them this way in order to directly compare our results to [1]. In order to normalize the first 10 features we utilize 2; where x is the sample feature, $\bar{\mu}$ is mean of that feature for all samples, σ is the variance of that feature for all samples, and x' is the normalized feature. This normalizes the data around 0.

$$x' = \frac{x - \bar{\mu}}{\sigma} \tag{2}$$

B. Hot-Vector Encoding

The labels for the data set come in the form of an integer values from the set 1-7. In order to utilized software max and directly compare to the output of our neural network we encoded the values into a hot-vector of size 6 (zero indexed) in which for each sample has one entry as 1 and the other 6 entries as 0, as illustrated by 3.

$$\begin{bmatrix} 1 = [1,0,0,0,0,0,0] \\ 2 = [0,1,0,0,0,0,0] \\ 3 = [0,0,1,0,0,0,0] \\ 4 = [0,0,0,1,0,0,0] \\ 5 = [0,0,0,0,1,0,0] \\ 6 = [0,0,0,0,0,1,0] \\ 7 = [0,0,0,0,0,0,1] \end{bmatrix} \tag{3}$$

C. Data Sets: Validation, Training, Testing

In order to compare our neural networks to [1], we must use the similar methodologies when splitting the data into validation, training, and testing sets. [1] only used 11,340 total samples for the training data set (1,620 from each cover type), 3,780 total samples from the training data for the validation set (540 from each cover type), and rest of the 565,892 samples were used for testing. Though this is only a small subset, it allows the training data to be independently identically distributed (I.I.D.) from each of the classification types. Therefore, in order to do a direct comparison we utilize the same number of samples and sample methodology as [1] in our neural networks. Though using such small set of samples from the labeled data set is not standard practice today, it will allows us to compare the effectiveness of our network architectures to past work [1].

Training	Validation	Testing
11,340	3,780	569,672

V. Neural Networks

A. Architectures

The architectures explored in [1] are listed in 4. Test results for these architectures are illustrated in Figure 5. As illustrated, they only explored networks with a single hidden layer, the best of which involved 120 units in the hidden layer. With the computation power of today's computers we will deepen that architectures in our tests by adding more hidden layers and more hidden units into to increase the capacity and decrease the training cost. In addition, [1] utilized Sigmoid (4) as the hidden layer activation function. Because Sigmoid has an upper saturation limit (see Figure 6), which can result in vanishing gradients, we utilize ReLU (5) in our architecture, which does not have an upper gradient saturation limit (see Figure 7). We discuss the learning rate in the V-B. Since the best results were achieved by utilizing all 54 features in [1], we utilize all 54 features in every architecture we analyze.

Number of independent variables	Network architecture	Learning rate	Momentum rate	Validation data set MSE
54	54-120-7	0.05	0.5	0.2747
53	53-120-7	0.05	0.5	0.2908
21	21-60-7	0.05	0.5	0.3061
20	20-60-7	0.05	0.5	0.3363
10	10-90-7	0.10	0.5	0.3312
9	9-60-7	0.05	0.6	0.3699

[a] Network architecture values represent the number of input nodes, number of hidden nodes, and the number of output nodes (respectively) present in the network.

Fig. 4: Accuracy explored by Blackard, et al. [1]

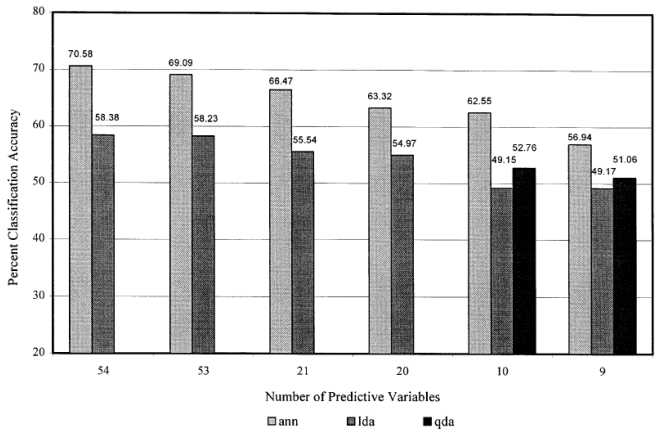

Fig. 5: Test Accuracy Vs. Number of Hidden Layer Units. [1]

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

$$ReLU(x) = max(0, x) \tag{5}$$

B. Adaptive Moments (ADAM)

We utilize ADAM as our learning rate algorithm which adapts the learning range to each dimension based on the exponentially weight averages of both the gradients (also known as momentum) and the square of the gradient (also known as RMSProp). This helps prevent the networks from getting stuck on local minima, while allowing for rapid convergence. Figure 8 compactly illustrates the basics of ADAM. We use recommended hyper-parameters provided by [2]. Testing with larger learning rates, smaller momentum, and smaller RMSProp hyper-parameters results in less desirable results.

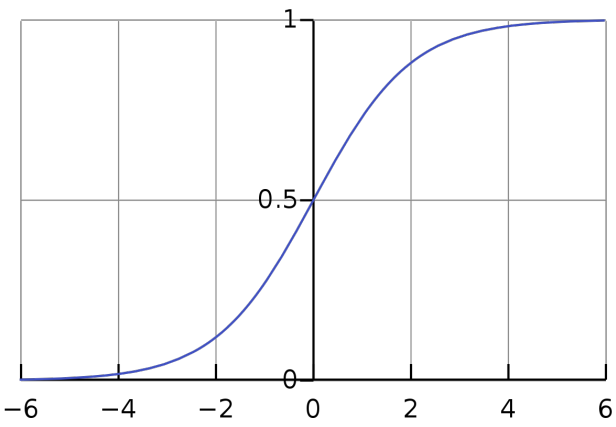

Fig. 6: Sigmoid Curve [6]

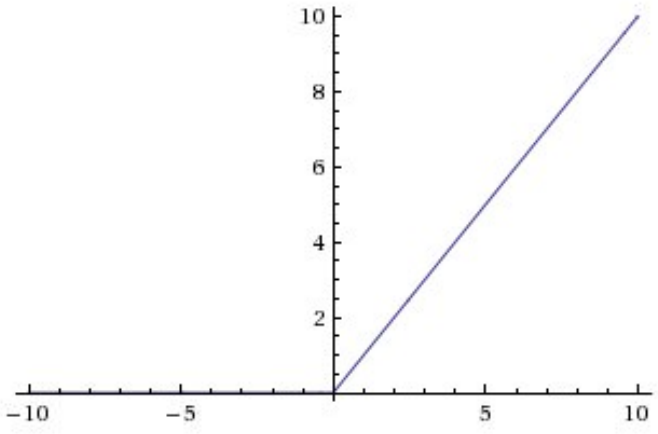

Fig. 7: ReLU Curve [7]

Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
$t \leftarrow t + 1$
Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$
Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$
Compute update: $\Delta\boldsymbol{\theta} = -\epsilon\frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}}+\delta}$ (operations applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

Fig. 8: Adaptive Moments (ADAM) Algorithm [4]

C. Dropout

We will utilize a small amount of dropout (0.25 hyper-parameter) in each hidden layer of our architectures. This will help prevent specification units from receiving too much

weight and biasing the entire network; therefore, it will help prevent overfitting our training data.

D. L1 and L2 Regularization

The intuition behind the use of L1 regularization for this problem, is that it creates sparse connections which drive some of the weight values to zero. Since there are 45 binary soil types and wilderness regions that have less importance than the first 10 features, a network should end up focusing primarily on the first 10 features and less on the binary soil types. L2 Regularization is presumed to have less of a benefit in this problem, since features do not have uniform importance; however, it is tested for comparison. Since L2 regularization has a tendency to push all weights towards zero and shift the focus of the network to all features (smooth the decision boundary), this will likely cause worse results due to the high number of less relevant binary features. We experimentally determine that the most effect hyper-parameters for L1 and L2 to be 0.00001 and 0.00001, respectively. If they have a higher order of magnitude, then the networks do not fit the training data, if they have a lower order magnitude, then they will have little effect on the results.

VI. Results

The results from 5 trial neural networks are shown in table IV. The architecture for each trials is shown in table I, the adam hyper-paramters are shown in II, and regularization techniques used in each trial are shown in III. Every hidden layer in every trial utilizes ReLU as the activation function, minibatch stochastic gradient descent with a minibatch size of 512 for GPU acceleration (see A), and dropout with a hyper-parameter of 0.25. Trial 1 is a baseline trial in which we utilize a similar to the network that produced the best results in [1]; that is 54-120-7. We trained for 10,000 epochs and reached a cost of 0.7509, a training accuracy of 0.9302 and a test accuracy of 0.7470. Even at baseline, with the adjustment of a few noted items, we achieve a higher testing accuracy than [1]. In order to explore the impact of a deeper architecture (and increased capacity) on the results, Trial 2 utilizes a similar architecture with the addition of a second hidden layer with 20 units. The results improved slightly with a decreased cost of 0.1392, a increased training accuracy of 0.9497, and a similar test accuracy of 0.7404. In Trial 2 the cost decreased as the network depth increased. This indicates that the original network capacity was too low to accurate fit the training data. [8] indicates success using an architecture with 3 hidden layers of 54-1024-512-256-7. This architecture has significantly more capacity than the architectures tested in Trials 1 and 2. Trial 3 utilizes this architecture to achieves significantly improved results with a low cost of 0.0004, a high training accuracy of 0.9908, and an improved test accuracy of 0.7657. The high capacity of the architecture from Trial 3 fits the training data much better than the architectures from Trials 1 and 2. There seems to be little that can be done to further improve the testing accuracy with this small subset of data beyond this point due to the already low cost. Additional training data would need to be used to improve the results. Trials 4 and 5 explore the impact of L1 and L2 regularization respectively with this successful architecture. Trial 4 utilizes L1 regularization with a hyper-parameter 0.00001, and results in a similar cost of 0.0046, a similar training accuracy of 0.9915, and a similar test accuracy of 0.7547. As illustrated the impact of L1 regularization is minimal. Trial 5 utilizes L2 regularization and results in a high cost of 1.6036, a low training accuracy of 0.9329, and a low test accuracy of 0.7430. As predicted in V-D, the results are worse with L2 regularization. Trials 1 and 2 with low capacity architectures have higher variation in cost (see figures 9 and 10) throughout the training cycle than Trials 3,4, and 5 (see figures 11, 12, and 13) that have high capacity architectures. In general, the higher capacity networks perform better, and all the networks improve upon the results obtained in [1].

TABLE I: Architectures

Trial	Architecture	Epochs	Minibatch Size
1	54-120-7	10000	512
2	54-120-20-7	10000	512
3	54-1024-512-216-7	2000	512
4	54-1024-512-216-7	2000	512
5	54-1024-512-216-7	2000	512

TABLE II: Adam

Trial	Initial Learning Rate	Momentum	RMSProp
1	0.001	0.9	0.999
2	0.001	0.9	0.999
3	0.001	0.9	0.999
4	0.001	0.9	0.999
5	0.001	0.9	0.999

VII. Related Work

As discussed throughout this work, we improve upon the network architectures used in [1]. All 5 trial architectures achieve higher test accuracy than [1]. The intuition behind this is twofold. First, our network architectures utilize modern regularization techniques and optimization algorithms that improve convergence rates and prevent overfitting. Second, Trials 2, 3, 4, and 5 present architectures with increased capacity allowing the networks to better fit the training data.

TABLE III: Regularization (Per Hidden Layer)

Trial	Dropout	L2 Norm	L1 Norm
1	0.25	0.0	0.0
2	0.25	0.0	0.0
3	0.25	0.0	0.0
4	0.25	0.0	0.00001
5	0.25	0.00001	0.0

TABLE IV: Results

Trial	Cost	Training Accuracy	Test Accuracy
1	0.7509	0.9302	0.7470
2	0.1392	0.9497	0.7404
3	0.0004	0.9908	0.7657
4	0.0046	0.9915	0.7547
5	1.6036	0.9329	0.7430

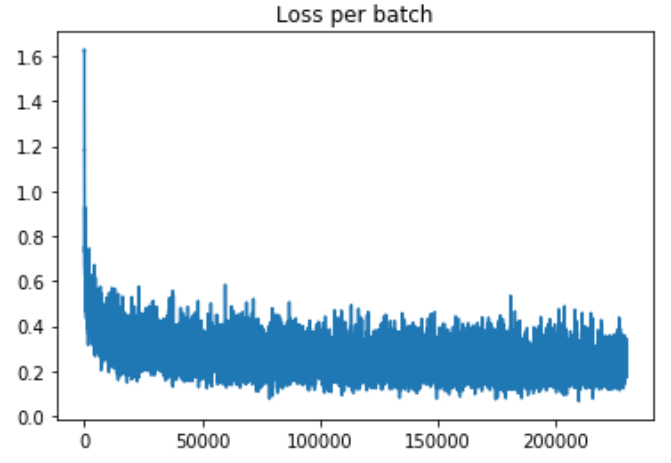

Fig. 9: Trial 1

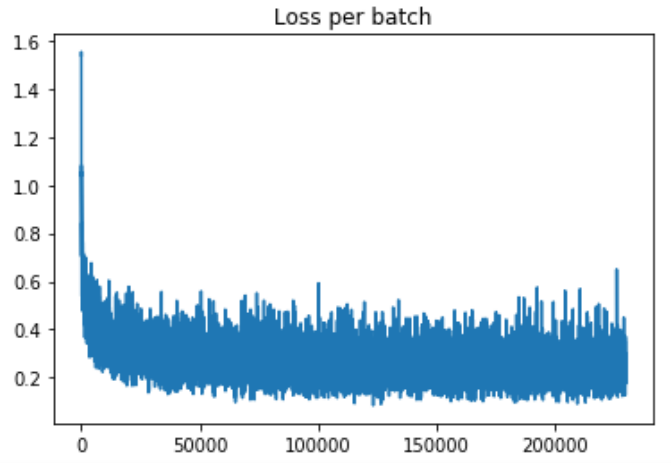

Fig. 10: Trial 2

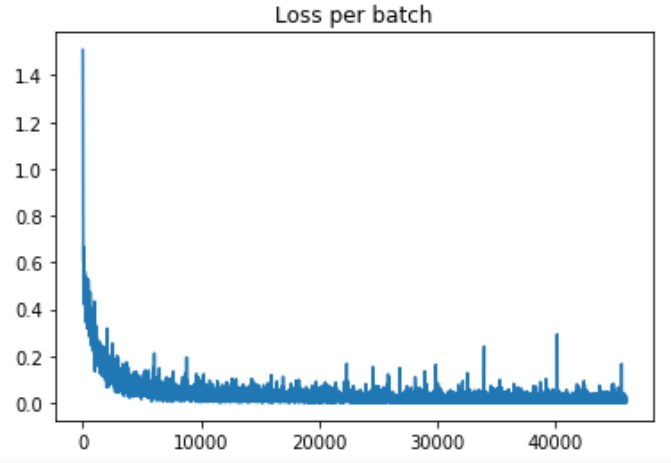

Fig. 11: Trial 3

We illustrate how deeper and wider networks provide a higher capacity that is better capable of fitting the training data.

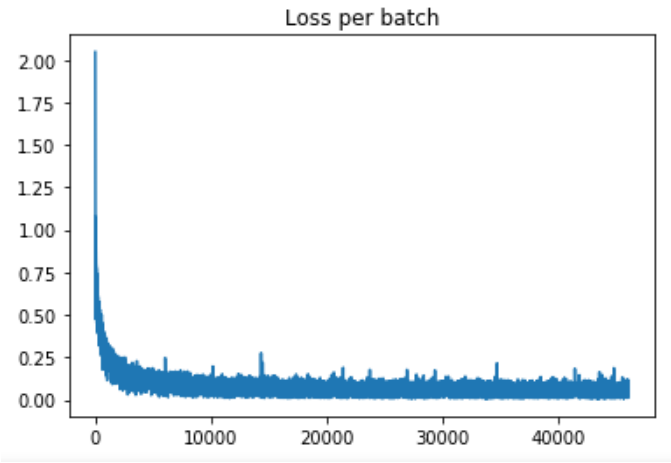

Fig. 12: Trial 4

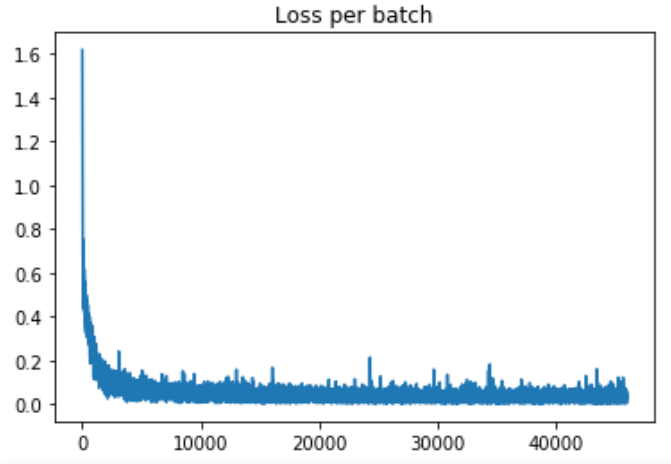

Fig. 13: Trial 5

VIII. Conclusion

A. Summary

We have explored improvements to neural networks used to classify forest type covers based on cartographic features. The data set we used contained 581,012 samples with 7 classifications based on 54 features. We utilized 11,340 independently identically distributed samples as the training data and 569,672 samples as the testing data for comparison to [1]. We presented 5 trial network architectures that achieved higher test accuracy in each case than [1]. Trial 3 achieved the best results with an architecture with 54-1024-512-216-7, 2000 epochs, a minibatch size of 512, 0.25 dropout per hidden layer, and no L1 or L2 regularization. This network achieved a cost of 0.0004, training accuracy of 0.9915, and test accuracy of 0.7547. [1] only used networks with a single hidden layer. We showed that be deepening and widening the network, we achieve a capacity better capable of fitting the training data.

B. Future Work

We would like to see a hypotheses testing approach applied for tuning the network architecture capacity. It is important

to find the appropriate network capacity that results in as low of a training cost as possible while achieving as high of a testing accuracy as possible. In addition, we would like to see the effect of utilizing more of the data set as training data. In order to compare to [1] we were limited to a smaller sample size; however, with the computational power available today it would be interesting to increase the training data set.

APPENDIX

GPU Acceleration: A GeForce GTX 1050Ti GPU was utilized for training these networks. The required supporting libraries of CUDA and cuDNN that were installed.

Wilderness Areas:

1) Rawah Wilderness Area
2) Neota Wilderness Area
3) Comanche Peak Wilderness Area
4) Cache la Poudre Wilderness Area

Soil Types:

1) Cathedral family - Rock outcrop complex, extremely stony.
2) Vanet - Ratake families complex, very stony.
3) Haploborolis - Rock outcrop complex, rubbly.
4) Ratake family - Rock outcrop complex, rubbly.
5) Vanet family - Rock outcrop complex complex, rubbly.
6) Vanet - Wetmore families - Rock outcrop complex, stony.
7) Gothic family.
8) Supervisor - Limber families complex.
9) Troutville family, very stony.
10) Bullwark - Catamount families - Rock outcrop complex, rubbly.
11) Bullwark - Catamount families - Rock land complex, rubbly.
12) Legault family - Rock land complex, stony.
13) Catamount family - Rock land - Bullwark family complex, rubbly.
14) Pachic Argiborolis - Aquolis complex.
15) unspecified in the USFS Soil and ELU Survey.
16) Cryaquolis - Cryoborolis complex.
17) Gateview family - Cryaquolis complex.
18) Rogert family, very stony.
19) Typic Cryaquolis - Borohemists complex.
20) Typic Cryaquepts - Typic Cryaquolls complex.
21) Typic Cryaquolls - Leighcan family, till substratum complex.
22) Leighcan family, till substratum, extremely bouldery.
23) Leighcan family, till substratum - Typic Cryaquolls complex.
24) Leighcan family, extremely stony.
25) Leighcan family, warm, extremely stony.
26) Granile - Catamount families complex, very stony.
27) Leighcan family, warm - Rock outcrop complex, extremely stony.
28) Leighcan family - Rock outcrop complex, extremely stony.
29) Como - Legault families complex, extremely stony.
30) Como family - Rock land - Legault family complex, extremely stony.
31) Leighcan - Catamount families complex, extremely stony.
32) Catamount family - Rock outcrop - Leighcan family complex, extremely stony.
33) Leighcan - Catamount families - Rock outcrop complex, extremely stony.
34) Cryorthents - Rock land complex, extremely stony.
35) Cryumbrepts - Rock outcrop - Cryaquepts complex.
36) Bross family - Rock land - Cryumbrepts complex, extremely stony.
37) Rock outcrop - Cryumbrepts - Cryorthents complex, extremely stony.
38) Leighcan - Moran families - Cryaquolls complex, extremely stony.
39) Moran family - Cryorthents - Leighcan family complex, extremely stony.
40) Moran family - Cryorthents - Rock land complex, extremely stony

REFERENCES

[1] F. M. Michel Goossens and A. Samarin, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables." [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169999000460

[2] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[3] ResearchGate, "1 sample of a feed-forward neural network." [Online]. Available: https://www.researchgate.net/figure/Sample-of-a-feed-forward-neural-network_fig1_234055177

[4] X. Koutsoukos, "Back-propagation for deep feedforward networks." [Online]. Available: https://brightspace.vanderbilt.edu/d2l/le/content/119549/viewContent/966708/View

[5] U. M. L. Repository, "Covertype data set." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Covertype

[6] "Logistic-curve." [Online]. Available: https://en.wikipedia.org/wiki/Sigmoid_function#/media/File:Logistic-curve.svg

[7] "Rectified linear units (relu) in deep learning." [Online]. Available: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning

[8] "Dnn classifier for forest type." [Online]. Available: https://www.kaggle.com/jeonghunyoon/dnn-classifier-for-forest-type/notebook