Towards Establishing a Predictive Machine Learning Model in Agriculture applying Convolutional Neural Networks

Bohdan Junior
bohdanjr@stanford.edu

Joshua Adams
adamsdsc@stanford.edu

Abstract

Introduction/Objective: *Machine learning is playing an increasingly important role in precision agriculture. We conducted a study using a collection of 3-channel images with 500 x 500 (910 total images) to predict the most possible classification of carbon levels in soil samples from its cellphone images.* ***Methods:*** *Building upon previous experiments using regression and classification models, we conducted new experiments with deep learning CNN models (AlexNet, VGG, and Resnet).* ***Results:*** *There was improvement of accuracy from 50% to 75% in more recent models.* ***Conclusions:*** *There is currently inconclusive evidence of mapping soil properties to images, provided that the images were very similar in textures / colors. Finally, we confirmed the possibility of enhancing predictions using ML. In addition, we present a path of a predictive ML model in future research with the use of semi supervised learning model such as meta pseudo labels.*

1. Introduction

The use of technology in agriculture has been expanding over the last couple decades so as to provide financial profit benefits to farmers and environmental benefits to both farmers and the global consumer community [22]. One of the accelerated reasons for use of technology in the global farming or agriculture industry is because of the increased stress due to climate change influence on the soil moisture content [7]. Therefore, one significant objective of research is to identify methods which can control and improve productivity of crops by evaluating the soil conditions [4]. Soil organic carbon (SOC) plays a key part in the moisture levels of soil and is a main component of soil organic matter (SOM) [1,8]. With the advent of advanced technologies that assist with planting, tillage, and the overall growth of plants and the improvement in AI/ML method approaches, such as deep learning, identifying changes in soil could greatly assist farmers in the overall planting process [4].

The training/testing input will include include soil images obtained from samples with associated predetermined carbon and nitrogen levels. We started with Convolutional Neural Networks (CNN) models based on classical architectures, AlexNet, VGG, and Resnet. We also planned to apply a more sophisticated model, meta pseudo labels – which is a semi-supervised method [17] time permitting.

1.1. Problem Statement

The specific problem that we attempted to solve is prediction of the carbon level for soil samples based cell phone images taken, using the three channel data of the correspondent image.

1.2. Input

The intention is to work with the dataset mentioned in Sec. 4. Using a collection of 3-channel images with 500 x 500 (910 total images). These cropped images were generated from image samples with a spatial resolution of 3096 x 4128 (newer). We applied multiple (mentioned in Sec. 3) CNN classification algorithms.

1.3. Output

Our algorithmic output goal is to predict the most accurate classification of soil to given C-levels. In this study, we relied on CNN to extract features from soil images and then classified the correspondent soil samples and aligned to given carbon levels. We applied multiple classification algorithms first to the training model, then to the validating labeled test data.

2. Previous and Related Work

2.1. Previous work

In a previous course project, CS229 Fall 2021, we applied classic machine learning (ML) algorithms resulting in minimal accuracy, ranging from 20 percent to 36 percent in classification models - see Tab. 1 [10].

As a suggestion, convolutional neural networks (CNN networks) was applied implementing a LeNet5 based on a Coursera Course [16]. We first obtained 100 percent accuracy on training set but only 20 percent on validation set, clear evidence of overfitting. Applying data augmentation

and exponential decay schedule we were able to increase test accuracy to 48 percent - see Fig. 1.

2.2. Related work

The use of ML is gaining increased attention for precision agriculture. As an example, Yang, Zhang, Guo and Zhuo obtained relevant results of carbon content in soil training CNNs with satellite images [21]. Similarly, challenges have been identified in obtaining the soil total nitrogen levels as part of soil health indicators and therefore attempts to develop and train ML models using drones, multispectural images, sensors, augmented realty, etc., as possible solutions in precision agriculture [12, 13]. Additionally, studies have attempted to obtain rapid and nondestructive characterization of soil properties by taking pictures from a phone for evaluating soil purposes [19]. Use of cell phone image analysis would then negate the need for laboratory analysis and allow for a better understanding of crop yield to obtain accurate yield predictions. This is significant where use of laboratory grade analyzers may not be available due to costs and geo-location constraints. Therefore, development of a machine learning image-based module trained by using different soil samples from different agricultural fields with highly variable C and N-levels at different soil moisture levels is necessary. Identifying SOC in a real-time manner may then assist in predicting potential soil crop yields.

Table 1. Classification models used in CS229

Model	Accuracy	Precision	Recall
Multiclass Softmax	0.20732	0.18537	0.20732
Multiclass GDA	0.36585	0.37615	0.36585
Neural Network	0.10909	0.10478	0.10909
LaNet-5 CNN	0.48300	0.47082	0.48300

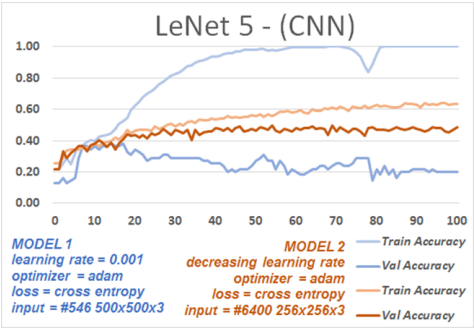


Figure 1. LeNet-5 architecture

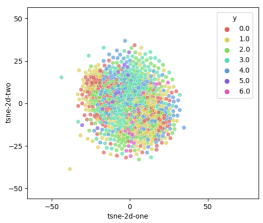
3. Method Approach

3.1. Classification

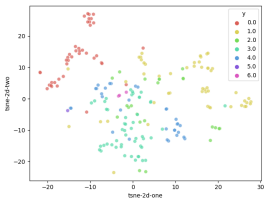
One essential aspect of supervised learning is the categorization of data-classification, specifically [18]. Classification skills are developed and repeatedly used through out life. In deep learning techniques, classification, is distinguished from traditional machine learning in the use of automatic data feature extraction and the design and implementation of artificial neural networks [4]. Classification is applied typically in phases - first, a classification algorithm is applied on the training data set and then the extracted model is validated against a labeled test data set to measure the model performance and accuracy [18].

3.2. The Challenge

Our soil images have 500 x 500 RGB pixels each, totalling $500 \cdot 500 \cdot 3 = 750,000$ pixels per image. Thus the challenge was to devise a way to identify some patterns presented in 750 K pixels/features of around 1,000 images, and associate it to the soil characteristics. If there were 2 or 3 features, we could visualize it on a 2D or 3D graph. With that amount of features (750 K), it was necessary to apply a technique to visualize high-dimensional data, locating the data point in a two dimensional map. In our study, we applied a t-Distributed Stochastic Neighbor Embedding (t-SNE), a variation of SNE that alleviates the problems of difficulties in optimizing due to cost function by using a symmetrized version of SNE, using simplified gradients and Student-t distribution [20]. It is essentially a mathematical approach in which high/low-dimensional Euclidean distances between datapoints can be converted into represented similarly pairwise conditional probabilities. For example, in our study we see the presence of clusters at several scales. From Fig. 2a, it was not possible to identify clusters associated to the classes. Since t-SNE was applied to the original entire dataset (thousands of features), it seems logical that relationships were not observed.



(a) t-SNE applied on the 910 original images with 750K features



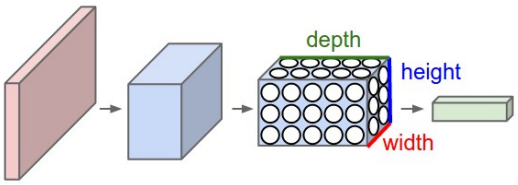
(b) t-SNE applied on randomly selected 200 images of the last dense layer of CNN-Alexnet

Figure 2. t-SNE

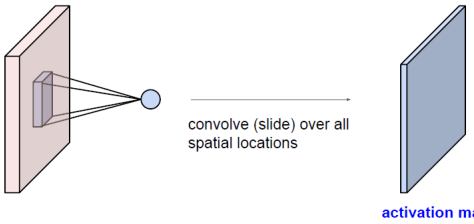
If however we look at Fig. 2b, it is possible to observe clusters associated with classes. This image represents a sample of 200 images from the original dataset that went through a trained CNN (Alexnet). Instead of 750K features per image, the trained CNN returned only 4,096 features per image, the most relevant ones to properly generate clusters and consequently classify the images. And it showed that it was feasible to classify the soil images using CNNs.

3.3. CNN Selection

We defined CNN initially here to map to all our trained models described later in Sec. 3.5. A CNN can be viewed as a sequence of layers. For this study, each method approach used received an volume input, *i.e.*, an image with height, width and depth (channels), and was transformed to a 3D *output* by means of a differentiable function - see Fig. 3. An *output*, called activation, in turn becomes the input for the next layer.



(a) The red represents the input layer, the blue represents convolutional layers and the green the output layer [5].



(b) A filter is convolved with the image, producing an activation map [5].

Figure 3. Used with permission from CS231n https://cs231n.github.io/convolutional-networks/. Copyright CS231n 2022.

Each layer of a CNN is composed of neurons with learnable weights and biases. It is common to refer to the sets of weights as a filter (or a kernel). Every filter is small spatially, but extends through the depth of the input volume. From the Fig. 3b, it is possible to see that during the forward process, each filter is slid across the image, computing the dot product between the entries of the filter and the input at any position. Ultimately giving the response of that filter at every spatial position. In our Fig. 4 example, there was a collection of 64 filters trained over a CNN network that activated a region in the input image whenever a pattern like it was seen.

Intuitively, the network will learn filters that activate when they see a pattern in the image, passing the activation map as an input to the next layer, until the forward pass is complete.

After its completion, the whole network should express a single differentiable score function, presenting a score. In our case the score function is the softmax:

$$P(y_i|x_i;W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \tag{1}$$

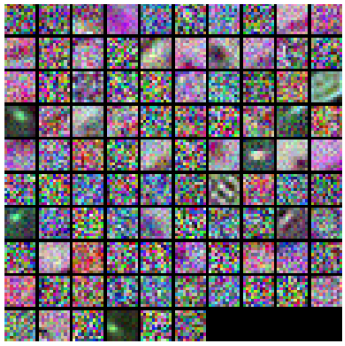


Figure 4. Visualization of (64 x 11 x 11 x 3) learned filters in the first conv layer of a CNN (Alexnet)

which can be intepreted as the normalized probability assigned to the correct label y_i given the image x_i and parametrized by W, usually done in the last layer.

3.4. Optimization

In Sec. 3.3, we obtained the score function for our problem - Eq. (1). Next, loss function was defined. Because we are computing for the hot class and it measures the quality of a particular set of weights based on how well the induced scores agreed with the ground truth labels. In our case, this was a categorical cross-entropy loss, seen in Eq. (2).

$$L = \sum_i -log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}) \tag{2}$$

Next, using the gradient descent algorithm over a number of iterations we minimized the loss - Eq. (3).

$$W_{t+1} = W_t - \alpha \frac{\partial L}{\partial W} \tag{3}$$

The loss was defined for both the forward pass and backward pass to obtain $\frac{\partial L}{\partial W}$. It is important to note that α is a hyper-parameter that must be defined. We trained our models using SGD (stochastic gradient descent) with mini-batch, due to better performance, rather than using vanilla gradient descent.

3.5. Training and Testing

Using the input data described in Sec. 4, we utilized Keras/Tensorflow to actually train and test the models, using Google Colab GPUs. During training, we obtained curving for both the trained and validation accuracies over all epochs. After the finalized training, we then obtained predicted y values from a test set. Using the obtained y values, an accuracy calculation and confusion matrix for each model was completed.

For our training and testing we implemented versions of LeNet-5, AlexNet, VGG, and ResNet models to the dataset,

and then attempted different optimizers, *e.g.* SGD, SGD with momentum, and ADAM, along with hyper-parameter value adjustments to improve the results. Softmax - Eq. (1), with 7 classes, to predict final result (cross-entropy loss - Eq. (2)) and other techniques such as learning rate decay was also attempted.

From Eq. (4) and Eq. (5), it is possible to see that ADAM takes advantage of first momentum by using moving average of gradient, i.e., keep following a direction (m_1); and it uses the squared gradients to scale the learning rate, to avoid vanishing gradients, by means of second momentum (m_2).

$$m_{1t} = \beta_1 m_{1t-1} + (1-\beta_1)\nabla_t \tag{4}$$

$$m_{2t} = \beta_2 m_{2t-1} + (1-\beta_2)\nabla_t^2 \tag{5}$$

$$W_t = W_{t-1} - \alpha \frac{\frac{m_{1t}}{1-\beta_1^t}}{\sqrt{\frac{m_{2t}}{1-\beta_2^t}}} \tag{6}$$

The details of the models can be seen in further from Sec. 3.5.1.

3.5.1 LeNet-5 architecture

Our first training attempted architecture and baseline model was the LeNet5- see Fig. 5 [15]. Starting, there was an initial *conv layer*, working as described in Sec. 3.3. Then a pool layer reduced the output to half without extra parameters to learn. Another combination of *Conv-Pool layers*, followed by two fully connected layers and softmax at the end of the network.

Even though LeNet-5 outperforms traditional ML methods. In previous work achieving 48% accuracy - see Tab. 1. We were not expecting excellent results from LeNet-5, since it recognizes characters very well, but does not perform well with more challenging domains, such as soil images.

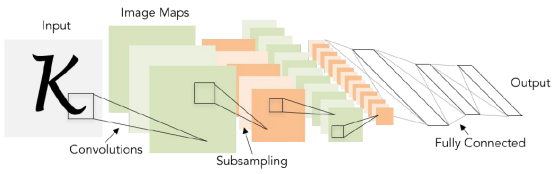


Figure 5. LeNet-5 architecture. Used with permission from CS231n Spring 2022. Copyright CS231n, 2022. [6]

3.5.2 AlexNet

We next trained using AlexNet. The seminal research on AlexNet, was relying on heavy data augmentation, the optimizer used (SGD + momentum) and the use of ReLU as the non-linear activation functions - see Eq. (7) [14]. In our experimentation, our data also relied heavily on data augmentation. Therefore, AlexNet was used as a potential solution. In addition, it is important to mention batch normalization was necessary after every *conv* layer for each training mini-batch which allowed higher learning rates and reduced training time.

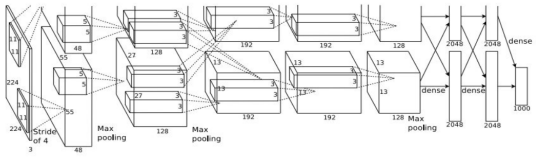


Figure 6. AlexNet architecture. Used with permission from CS231n Spring 2022. Copyright CS231n, 2022. [6]

$$ReLU = max(0, x) \tag{7}$$

3.5.3 VGG

Figure 7. VGG-16 architecture. Used with permission from CS231n Spring 2022. Copyright CS231n, 2022. [6]

The idea behind VGG (see Fig. 7) is to use smaller filters that had the same effective receptive field as greater filter having the benefit of learning more non-linearly with less parameters compared to AlexNet. However, according to Alom et al. (2018), there is a need for more layers within the CNN network [3] using VGG. We did expect to obtain better results using VGG (see Sec. 5.3).

3.5.4 ResNet

ResNet was developed for the purpose of developing very deep networks using residual connections to avoid the vanishing gradient problem - see Fig. 8 [3]. Although very computational expensive, we do expect this architecture to be the optimal model, due to the ability to train nets with more than 150 layers.

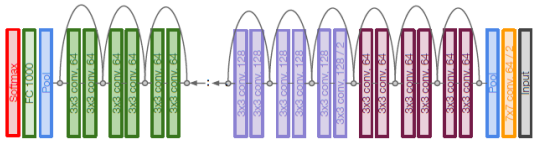
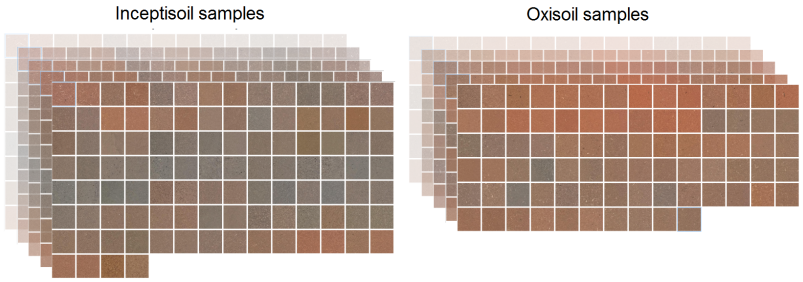


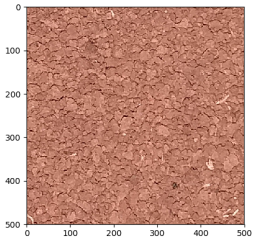
Figure 8. ResNet architecture. Used with permission from CS231n Spring 2022. Copyright CS231n, 2022. [6]

4. Dataset and Features

We used the same image samples from our CS229 project, however new photos were retaken from those samples [9]. Our current code used can be seen here [2]. These



(a) 5 x 2 x 51 = 510 Inceptisoil and 5 x 2 x 40 = 400 Oxisoil 500 x 500 pixels images



(b) Inceptisoil sample example.

Figure 9. Image dataset

images from the samples had spatial resolution of 3096 x 4128, replacing the 2448 x 3264 original ones. And with this new resolution it was possible to crop five (500 x 500) images from each original image.

4.1. Original Dataset Images

The soil samples were collected using the following approach: air dried, ground and then sieved through a 2 mm sieve for image capturing and processing from five sites in Brazil. The laboratory processing comprised of combustion utilizing a FlashEA 1112 Analyzer at a high temperature to obtain the carbon and nitrogen converted gas levels.

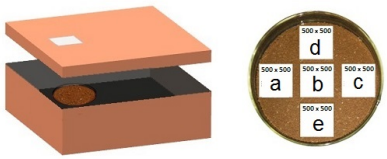


Figure 10. System used for the image acquisition

In total, 91 soil samples were collected, 40 from Oxisol and 51 from Inceptisol. The soil images were collected with a smartphone with 12.78 megapixels camera (Model Samsung J4) with fixed flash for constant lightning from a fixed distance (12 cm). The method used for the image acquisition is shown in Fig. 10, and consists of a cardboard box with dimensions of 0.30 m x 0.22 m x 0.12 m with an opening for image acquisition. For each sample 2 images we registered and stored in JPEG extension with 24 bits and a spatial resolution of 3096 x 4128 pixels. For each sample image, five cropped 500 x 500 pixels images were produced - see Fig. 9b, totaling 910 images Fig. 9a.

5. Experiment Results and Discussion

In this section we outlined our results for each of the defined CNN approaches from sections Sec. 3.5.1 to Sec. 3.5.4. For context support to our experiment, each image corresponds to a sample with a previously measured C-level within a range of $[0.31, 6.62]$ percentage of soil composition - see Fig. 11.

Based on these C-values, we adapted a regression input into a classification one with 7 classes corresponding

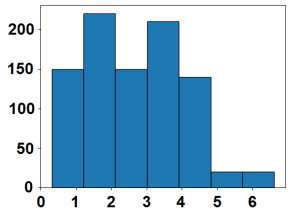


Figure 11. C-Levels distribution among the samples

to seven buckets, as shown in Fig. 11. The use of this approach required implementing a one-hot representation, where $y \in \mathbb{R}^K$ is the ground-truth vector for the training example x such that $y = [0, ..., 0, 1, 0, ..., 0]^T$ contains a single 1 at the position of the correct class.

Due to the current limited number of images (910), it was necessary to use data augmentation, specifically random cropping different scales, *e.g.* 224 and 256. The sample image data were then split into train, validation and test sets.

5.1. LeNet-5

From previous work, we knew that it was possible to observe a reduction of the cross entropy loss by means of using an ADAM optimizer with a constant learning rate ($\alpha_0 = 001$), and that was possible to overfit a relatively small sample of 546 images, obtaining a 100% accuracy on the training set with approximately 20% on the val set - see Fig. 1.

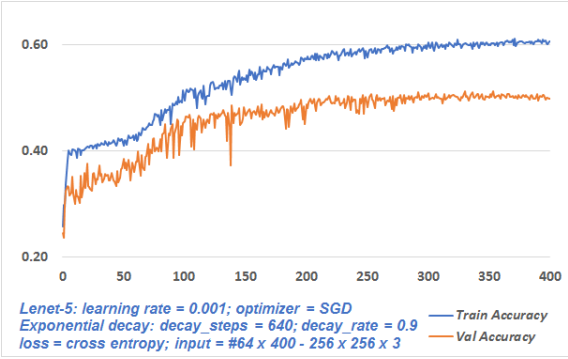


Figure 12. Lenet-5 learning curve

To improve our baseline results, we duplicated the data augmentation technique using more images, since we knew that heavy data augmentation was used in Alexnet. Although the original images were reduced from 500 x 500 x 3 to 256 x 256 x 3 by means of random cropping the original images, relatively uniform in terms of color and textures, it was possible to obtain 64 different samples per epoch with our renewed dataset. The train-dev-test split used was 70-20-10, based on the small quantity of samples: 910.

With respect to the optimizer, we decided to use SGD with momentum equal to 0.9 instead of Adam, as we knew that this combination could outperform Adam. Using this approach, we trained for 100 epochs and observed lots of oscillation at the beginning of training curve (dev accuracy). Thus we applied an exponential decay of 90% at 640 step interval, increased the amount of epochs to 400 and then observed a smoothed learning curve after epoch 300. Although the accuracy on training set was reduced to 61%, it was possible to obtain 51% on validation and 50% on the test set, raising the bar of our baseline model - see Fig. 12.

5.2. Alexnet

We next conducted an experiment with AlexNet. It was necessary to re-scale the dimensions of input data from 256 x 256 to 227 x 227, keeping 3 channels. We relied on heavy data augmentation such as the seminal work, using 64 images per epoch, randomly cropping the images from the original 910 images. The train-dev-test split used was 70-18-12, trying to keep a considerable amount to both dev and test sets.

Initially we decided to increase the initial learning rate to $\alpha_0 = 0.04$, trying to reach high values of accuracy earlier. However we noticed that loss was not diminishing as expected. Then we used the same initial value previously used with Lenet: $\alpha_0 = 0.001$.

In order to improve val accuracy, we changed the optimizer to ADAM, keeping the values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We also changed the decay rate of learning rate to 75%, expecting an early amortization. By tweaking the hyper parameters, we gained almost 10 percentage points, and obtained 70.7% accuracy at test set after 400 epochs - see Fig. 13.

5.3. VGG 19

Based on the original VGG-19 architecture, we started with training the model at an initial learning rate $\alpha_0 = 0.05$, altering the size of input to 224 x 224 x 3, using ADAM optimizer with α exponential decay with decay rate equal to 90%. The train-dev-test split used was 70-18-12. In the original work, a batch size equal to 256 was used. Due to our hardware limitation, we used 64. Although training accuracy reached almost 100%, val accuracy value was around 50%.

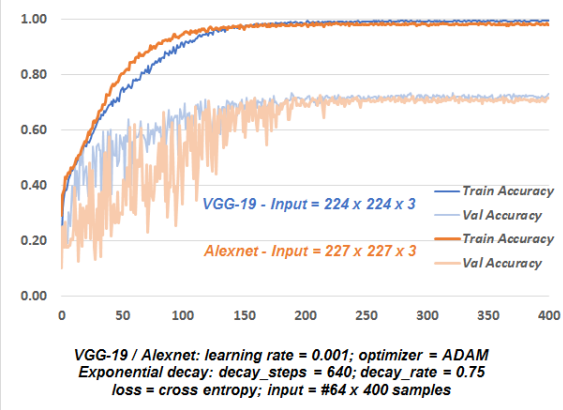


Figure 13. AlexNet and VGG-19 learning curves

We changed the original architecture, adding batch normalization after every convolutional layer in order to reduce overfitting. We also subtracted all pixel values from the mean value, mimicking the original paper; altered the initial learning rate to $\alpha_0 = 0.001$ and the decay rate to 75%, since it worked well with LeNet and AlexNet architectures; and finally added *He* uniform initialization to weights and zero initialization to bias, also following the configuration of original article. After 300 epochs, it was possible to obtain 73.3% accuracy on dev set, and 73.4% accuracy on test set - see Fig. 13.

5.4. Resnet-50 and Resnet-152

For clarification, prior to discussing the results of ResNet here. GoogLeNet was an option in our experiment. However, due to time constraints and that we were implementing ResNet, we decided not to implement this network under the assumption Resnet would give better results. Specifically, implementing a deep network ResNet of 152 layers. Instead of implementing the original GoogLeNet, it would be more advantageous to implement improved nets based on the original, e.g. Inception V2, V3 or V4.

We tried two architectures, ResNet-50 and ResNet-152. It was possible to train the two nets simultaneously, one with local GPU (GTX-970M/3Gb) and the other with Google Colabs (Tesla-T4/16Gb).

We relied on data augmentation, generating 32 / 64 images per epoch for Resnet 50 / 152, respectively. The batch sizes were 32 / 64 likewise, due to hardware constraints. The input dimensions were 160 x 160 to Resnet-50, due to hardware limitations too, and 224 x 224 for the Resnet-152. The train-dev-test split used was 70-20-10.

Starting with the initial learning rate equal to $\alpha_0 = 0.001$, with exponential decay using decay step equal to 640 and decay rate equal to 0.75. At the same time the optimizer was SGD with momentum = 90%, and using Nesterov. The weights were initialized with Gorot uniform. In both architectures the results were unsatisfactory, reaching around

62%.

We then changed the optimizer to ADAM, the initial learning rate to $\alpha_0 = 0.0005$ and decay rate to 0.9. With these changes we observed 10 percentage points gain after 400 epochs, though it was undesirable yet. We were expecting results above 75%. To achieve the expected results, we decreased the learning rate to $\alpha_0 = 0.00025$ and decided to train longer, for 1,000 epochs (Resnet-152) and 2,000 epochs (Resnet-50) - see Fig. 14. The achieved accuracy, for dev set, was 77.13 % for Resnet-50, and 74.29 % for Resnet152.

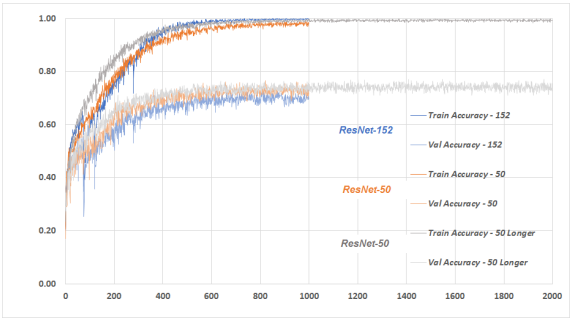


Figure 14. Resnet-50 and Resnet-152 learning curves

Although Resnet-50 outperformed the deeper network, we concluded that it is a question of training time to achieve higher accuracies and minimum adjustments in learning rate. In the original research, the authors performed 60 thousand iterations [11]. In our study we trained for one thousand iterations, needing approximately 18 hours. Applying Resnet-152, using the original parameters and setup from the original research would require longer training times based on our available processing power: 18 h x 60 = 1,080 h = 45 days.

5.5. Comparing results

A confusion matrix was used to determine the performance of the classification obtained from the proposed models. The confusion matrix is an N × N square matrix, where N denotes the number of categories and the elements on the main diameter represent the correct samples. Thus to obtain the efficiency of a classifier, it is just enough to divide the sum of elements in the principal diagonal of the matrix by the sum of all its elements; this is often called overall accuracy - Eq. (8).

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

where TP, TN, FP and FN are true positive, true negative, false positive, and false negative, respectively. Accuracy metrics measured the percentage of correctly classified patterns to the total number of patterns. It is possible to see confusion matrices of the models at Fig. 15, and the calculated overall accuracy at Tab. 3. In terms of overall accuracy, Resnet-50 obtained the best value in dev set and in test set. It is interesting to notice that Resnet-152 was not the top model, based on the explanation given on Sec. 5.4.

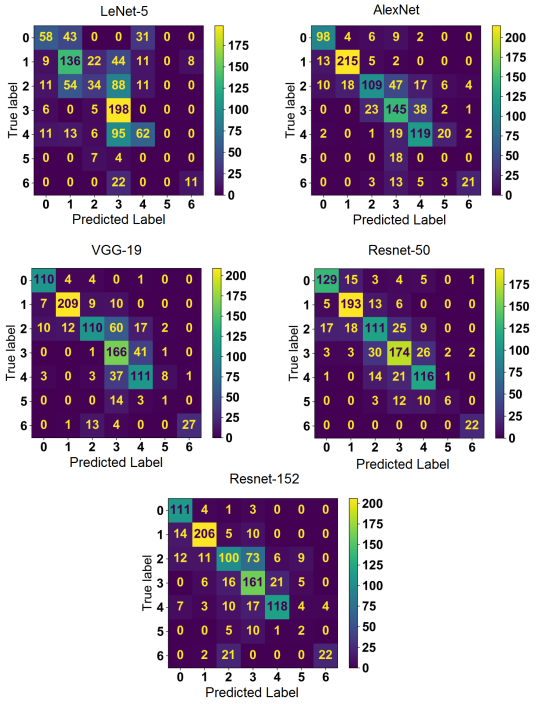


Figure 15. Confusion matrices for the models

In addition to accuracy which evaluates the overall effectiveness of a classifier model, computing individual accuracy gives us other metrics to better judge the performance of the classifier. Therefore, user's accuracy (precision) and producer's accuracy (recall) were computed based on the values of the confusion matrix - Eq. (9) and Eq. (10).

$$precision = \frac{TP}{TP + FP} \tag{9}$$

$$recall = \frac{TP}{TP + FN} \tag{10}$$

where precision represents how well reference images of ground truth are classified and recall shows how well a certain class has been classified.

The difference from precision and recall are better illustrated seeing both values in Tab. 2 for class 5 in Resnet-50 model. Precision is around 70%, meaning that around 7 times out of 10 the model classifies the reference images according to the ground truth. On the other hand recall is around 20%, meaning that the score for classification of this particular class is accurate only 1 in 5 attempts. The low score for recall, in our view, can be related to the low

Table 2. Precision and Recall of Models

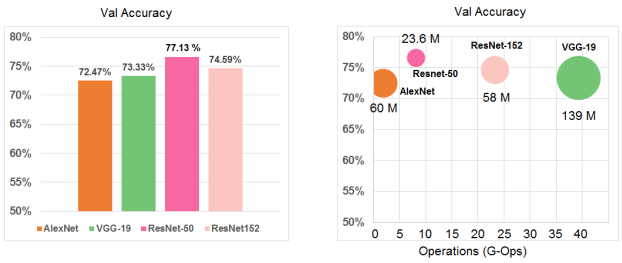
Class	LeNet-5		AlexNet		VGG-19		Resnet-50		Resnet-152	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Class 0	0.61	0.44	0.80	0.82	0.85	0.92	0.83	0.82	0.77	0.93
Class 1	0.55	0.59	0.91	0.91	0.92	0.89	0.84	0.89	0.89	0.88
Class 2	0.46	0.17	0.74	0.52	0.79	0.52	0.64	0.62	0.63	0.47
Class 3	0.44	0.95	0.57	0.69	0.57	0.79	0.72	0.72	0.59	0.67
Class 4	0.54	0.33	0.73	0.69	0.64	0.68	0.70	0.76	0.81	0.72
Class 5	0.00	0.00	0.00	0.00	0.08	0.06	0.67	0.19	0.10	0.11
Class 6	0.58	0.33	0.47	0.58	0.96	0.60	0.88	1.00	0.85	0.49

Table 3. Classification models

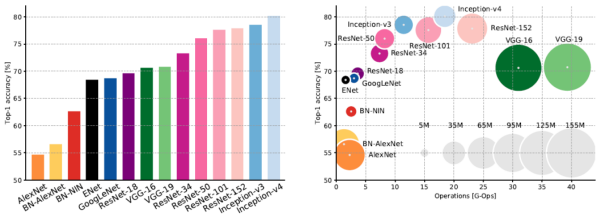
Model	#params	train acc	val acc	test acc
LeNet-5	7.4 M	0.6113	0.5136	0.4990
AlexNet	60.0 M	0.9880	0.7247	0.7070
VGG-19	139.0 M	0.9963	0.7333	0.7340
Resnet-50	23.6 M	1.0000	0.7713	0.7510
Resnet-152	58.0 M	0.9990	0.7429	0.7200

number of samples for this particular class - see Fig. 11. It is possible to observe that both Lenet-5 and Alexnet did not classified well the class 5 (recall equal to zero). More samples can help the model to identify relevant features associated with such class.

Another relevant observation is related to class 2: the recall is approximately 50% for the Alexnet, VGG and Resnet models. Since the number of samples is equivalent to classes 0, 1, 3 and 4, we can infer that can be related to our initial hypothesis of buckets division. Should our divide be different, the results could be improved.



(a) Val accuracy, number of parameters and operations of present work



(b) Top-1 accuracy, number of parameters and operations of the models performing on ImageNet. Used with permission from CS231n Spring 2022. Copyright CS231n, 2022.

Figure 16. Performance of the models.

6. Conclusions and Future Work

Based on the results we obtained previously in CS229 with accuracy values around 48%, and on the improvements obtained in this work- 76%, we are motivated to continue this research.

Reviewing Fig. 16a it is possible to see the accuracies obtained from AlexNet, VGG-19, ResNet-50 and ResNEt-152 in our study. For comparison, reviewing Fig. 16b shows accuracy values, computation cost and number of parameters of the most relevant entries submitted to the ImageNet challenge. Although the context is different from ours, it is noteworthy that the relations of Top-1 accuracies are similar to the relations of val accuracies we obtained in our work. Whether or not this is coincidental remains to be discerned.

Finally, according to [17], the Pseudo Labels methods work by having a pair of networks, one as a teacher and one as a student. The teacher generates pseudo labels on unlabeled images. These pseudo labeled images are then combined with labeled images to train the student. Thanks to the abundance of pseudo labeled data and the use of regularization methods such as data augmentation, the student learns to become better than the teacher. Seeing the results of ImageNet tests, it maybe possible to achieve 90% accuracy in our research. However, this approach only works within a TPU environment, due to the number of a params: 0.5 Billion. Additionally, we will need more uniformly distributed soil sample dataset - see Fig. 11. Next steps will then be to implement a semi-supervised method, as meta pesudo labels, to achieve accuracies around 90%.

In conclusion, once our accuracies and approaches are validated for better accuracy levels, it would be possible develop an application solution for precise agriculture benefiting globally.

7. Contributions and Acknowledgments

7.1. Contributions

Bohdan Junior developed and implemented the models. In addition, he participated in drafting the final report-all sections. Joshua Adams participated in drafting and editing the final paper, discussing and validating the models.

7.2. Acknowledgments

We would like to thank Larissa Macedo dos Santos-Tonial, PhD professor at UTFPR, as part of on-going research for the soil sample images. The soil samples were obtained from Larissa Macedo dos Santos-Tonial, PhD professor at UTFPR, as part of ongoing research. The sample images were collected at five sites in the south of Brazil.

References

[1] FAO - news article: World's most comprehensive map showing the amount of carbon stocks in the soil launched. 1

[2] Joshua Adams and Bohdan Junior. cs231n-sp2022. https://github.com/adamsdsit/cs231N-SP2022, 2022. 4

[3] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018. 4

[4] Afshin Azizi, Yousef Abbaspour Gilandeh, Tarahom Mesri-Gundoshmian, Ali Akbar Saleh-Bigdeli, and Hamid Abrishami Moghaddam. Classification of soil aggregates: A novel approach based on deep learning. *Soil and Tillage Research*, 199:104586, 2020. 1, 2

[5] Staff CS231n. Convolutional neural networks (cnns / convnets), 2022. 3

[6] Staff CS231n. Lecture 6, 2022. 4

[7] European Environment Agency. Soil, land and climate change. Sept. 2019. 1

[8] Food and Agriculture Organization of the United Nations. Global soil organic carbon map. 1

[9] Garg, Rishu, and Junior, Bohdan and Adams, Joshua. CS229-AgricultureML. 4

[10] Garg, Rishu, and Junior, Bohdan and Adams, Joshua. Toward Predicting a Machine Learning Algorithm in Agroculture. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[12] Md Abir Hossen, Prasoon K Diwakar, and Shankarachary Ragi. Total nitrogen estimation in agricultural soils via aerial multispectral imaging and libs. *Scientific Reports*, 11(1):1–11, 2021. 2

[13] Janna Huuskonen and Timo Oksanen. Soil sampling with drones and augmented reality in precision agriculture. *Computers and electronics in agriculture*, 154:25–35, 2018. 2

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 4

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[16] Andrew Ng, Kian Katanforoosh, and Younes Bensouda Mourri. Convolutional neural networks. 1

[17] Hieu Pham, Qizhe Xie, Zihang Dai, and Quoc V. Le. Meta pseudo labels. *CoRR*, abs/2003.10580, 2020. 1, 8

[18] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315, 2016. 2

[19] R.K. Swetha, Prajwal Bende, Kabeer Singh, Srikanth Gorthi, Asim Biswas, Bin Li, David C. Weindorf, and Somsubhra Chakraborty. Predicting soil texture from smartphone-captured digital images and an application. *Geoderma*, 376:114562, 2020. 2

[20] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 2

[21] Lin Yang, Yanyan Cai, Lei Zhang, Mao Guo, Anqi Li, and Chenghu Zhou. A deep learning method to predict soil organic carbon content at a regional scale using satellite-based phenology variables. *International Journal of Applied Earth Observation and Geoinformation*, 102:1–10, 2021. 2

[22] Naiqian Zhang, Maohua Wang, and Ning Wang. Precision agriculture—a worldwide overview. *Computers and electronics in agriculture*, 36(2-3):113–132, 2002. 1