

Optimization of models for recognizing diseased plants of tomato leaves using deep learning technologies

*Baratbek Sabitov*¹, *Nazgul Seitkazieva*^{2*}, *Esenbai uulu Suiunbek*³, *Sharshenbek Zhusupkeldiev*¹, and *Nurzat Asanbekova*²

¹Kyrgyz National University named after J. Balasagyn, Bishkek, Kyrgyzstan

²Kyrgyz State University named after I. Arabaev, Bishkek, Kyrgyzstan

³Osh State University, Osh, Kyrgyzstan

Abstract. In this work, using deep learning technologies as an element of artificial intelligence, various neural network architectures were built that model diseases of strategically important agricultural plants. Modern deep learning optimizers are proposed, and optimization problems for the proposed neural network architectures are studied. The main tasks of agriculture are identified, which today is a key sector for determining economic stability and solving food security. In this work, models for predicting yields based on deep learning are built. The main tasks of agriculture, which today are relevant in the context of climate change, are substantiated. The consequences and causes of the influence of climate change on many large categories of agriculture have been identified. The reasons for the influence of temperature anomalies in summer and winter on plant diseases have been established. The work examines voluminous concepts such as the yield of agricultural crops that lose stability with the rapid spread of various plant diseases, causing millions of losses to farmers and agricultural producers. Methods based on artificial intelligence have been proposed that can prevent the causes and factors of plant diseases. Models created on the basis of neural networks and optimizers are proposed to guarantee the accuracy of models for identifying and early diagnosing plant diseases. In the work, based on models built on the basis of deep learning, plant diseases are predicted using a database of sick and healthy plant images. The accuracy of the models has been established, which can guarantee high-quality prediction of the task. Based on the problem of optimizing neural networks for recognizing plant diseases, a mechanism for applying the selection of optimal parameters and selecting neural network architectures has been developed. The accuracy of the created models was analyzed, which is an assessment of their quality. When optimizing a neural network, the main attention in the work is paid to the task of choosing optimizers. Based on modern optimizers, various models have been obtained for predicting plant diseases based on a database collected in the field.

* Corresponding author: s.nazgul.s@mail.ru

1 Introduction

Determination of plant diseases and its early diagnosis is the main factor for preventing the loss of yields of many agricultural crops. It is one of the complex tasks of agriculture, interconnected with many external factors, requiring various new approaches to farming. The traditional approach involves an industry specialist who makes a diagnosis through careful analysis of the foliage surface. However, not all farmers in underdeveloped countries can afford consultations as they are expensive. Deep learning methods are one of the main elements of artificial intelligence, which can automate predictive analysis and its application in digital agriculture with great accuracy. An important task is its implementation to solve various problems associated with forecasting crop diseases. The work examines one of the most important problems of food security today. To this end, the use of modern approaches to obtain a stable crop yield is to support farmers to implement sustainable agricultural practices from environmental, social and economic points of view.

The research in this work is devoted to methods for processing images of diseased plants of various crops and machine learning methods to identify diseases of plant leaves. The problem of identifying signs of the appearance of diseases, i.e., is being solved. determining feature vectors using modern machine learning methods. The importance of determining the influencing factors on the spread of plant diseases is the key to a stable harvest.

The introduction of agriculture based on modern approaches to modeling agricultural problems for detecting plant diseases and pests is one of the important problems, the main task of which is predicting diseases and pests from collected plant images [1]. Currently, machine vision-based technologies for detecting plant diseases and pests are widely used in agriculture and have to some extent replaced traditional naked eye identification.

Traditional computer vision based plant disease and pest detection often uses conventional image processing algorithms or manual feature engineering plus classifiers [2]. Several classical algorithms for recognizing crop diseases are considered in [3].

In recent years, deep learning models represented by convolutional neural networks (CNNs) have been successfully applied, which have found applications in many areas of computer vision such as plant disease detection [4], medical image recognition [5], script text recognition [6], facial expression recognition [7] and eye pupil recognition [8].

Deep learning overcomes the disadvantage that traditional algorithms rely on artificially created features and is attracting more and more attention from researchers. It is currently successfully applied in computer vision, pattern recognition, speech recognition, natural language processing, and recommendation systems [9-10]. Good state-of-the-art results have been obtained using deep convolutional neural network (CNN) [11-12]. Third-party open-source tools for deep learning are now widely used. In the works of [13-14] and [15], the authors studied some agricultural problems in predicting yields, recognizing diseases of agricultural plants and horticultural crops.

2 Materials and methods

Various neural network optimizers are widely used to build plant disease recognition models. Optimizers in deep learning are algorithms used to tune model parameters to minimize a quality functional or loss function. The choice of optimizer can greatly affect the performance and training speed of the model. Now let's look at some of the most commonly used optimizers in deep learning, including: Gradient Descent, Stochastic Gradient Descent (SGD), Adam, and RMSprop.

Each optimizer has its own unique characteristics and advantages, and the choice of optimizer depends on the problem and model architecture. We discuss the numerical

implementation of the convergence of each optimizer and their comparative result. Optimizers in deep learning play an essential role in determining the accuracy of the built model. At the same time, choosing the appropriate parameters for a deep learning model is important, since this can significantly affect its performance. Optimization algorithms have different strengths and weaknesses and are better suited for certain problems and architectures.

Gradient descent iteratively reduces the loss function by moving in the opposite direction of the gradient. This depends on the derivatives of the loss function when searching for minima. Uses data from the entire training set to calculate the gradient of the cost function to the parameters, which requires a large amount of memory and slows down the process.

We can write the basic form of the algorithm as follows:

$$\theta^k = \theta^{k-1} - \alpha \nabla_{\theta} L(\theta^{k-1}) \quad (1)$$

Where θ is a model parameter, $L(\theta)$ is the loss function, and α is the learning rate. At the initial moment of time, the choice of parameter α can be carried out according to the following rules

$$\alpha = 0.1; 0.01; 0.001; \quad (2)$$

$$\alpha = \alpha_n = \frac{1}{\min(n+1, mn)}, n = 0, 1, 2 \quad (3)$$

$$\alpha_{min} = \frac{1}{mn} \quad (4)$$

The advantages of gradient descent are the simplicity of its implementation. With optimal speed settings, learning can achieve good results. It can converge slowly, especially for complex models or large data sets. This algorithm is sensitive to the choice of learning rate.

Stochastic gradient descent (SGD) is a variant of gradient descent that involves updating parameters based on a small, randomly selected subset of data (i.e., a "mini-batch") rather than the entire data set. We can write the basic form of the algorithm as follows:

$$\theta^k = \theta^{k-1} - \alpha \nabla_{\theta} L(\theta^{k-1}; x^{(i)}, y^{(i)}) \quad (5)$$

Where $x^{(i)}, y^{(i)}$ represents a mini-packet of data.

This can be faster than standard gradient descent, especially for large data sets. It may be easier to avoid local minima.

It may contain noisy data, which reduces the accuracy of the model.

In this case, we must make additional parameter settings.

Stochastic gradient descent with momentum (Nesterov optimizer). The Nesterov algorithm, or with momentum, is a variant of stochastic gradient descent that adds "momentum" to the iteration when updating. This technology helps the optimizer continue moving in the same direction, even if the local gradient is small and stuck at local minima. The momentum parameter is typically considered to be a value between 0 and 1. Our iterative parameter updating process for this method can be written as follows:

$$v = \beta v + (1 - \beta) \nabla_{\theta} L(\theta^{k-1}; x^{(i)}, y^{(i)}) \quad (6)$$

$$\theta^k = \theta^{k-1} - \alpha \cdot v \quad (7)$$

Where v is the momentum vector and β is the momentum hyperparameter.

SGD-Nesterov can help the optimizer navigate more efficiently through the “flat” regions of the loss function, which will help dramatically reduce fluctuations and improve convergence. If the impulse is too high, it can change the trajectory avoiding the minimum point. In this case, some adjustments to the momentum hyperparameter are required.

RMSProp. RMSProp is an optimization algorithm similar to Adagrad, but it uses the exponentially decreasing average of the squares of the gradients rather than the sum. This helps reduce the monotonic decay of Adagrad's learning rate and improve convergence. We can write the update rule like this:

$$g = \nabla_{\theta} L(\theta; x^{(i)}; y^{(i)}) \quad (8)$$

$$G = \beta G + (1 - \beta) \cdot g \odot g \quad (9)$$

$$\theta^k = \theta - \frac{\alpha}{\sqrt{G + \varepsilon}} \odot g \quad (10)$$

Where g is a matrix accumulating the squares of the gradients, ε is a small constant added to avoid division by zero, and β is the decay rate hyperparameter. This algorithm works well with sparse data. Automatically adjusts learning rate based on parameter updates. Can converge faster than Adagrad. Due to some problems, it may still converge too slowly. Requires adjustments to the decay rate hyperparameter.

Adam (short for adaptive torque estimation) is an optimization algorithm that combines the ideas of momentum stochastic gradient descent and RMSProp. This algorithm is the main algorithm and is used in many deep learning architectures. Similar to RMSProp, it uses the exponentially decaying average of the gradients and the squares of the gradients to determine the updated scale. It also uses momentum to help the optimizer navigate the loss function more efficiently. The algorithm for updating iteration over variables can be written as follows:

$$g = \nabla_{\theta} L(\theta^{k-1}; x^{(i)}; y^{(i)}) \quad (11)$$

$$m = \beta_1 m + (1 - \beta_1) \cdot g \quad (12)$$

$$v = \beta_2 v + (1 - \beta_2) \cdot g \odot g \quad (13)$$

$$\hat{m} = \frac{m}{1 - \beta_1^t}, \hat{v} = \frac{v}{1 - \beta_2^t} \quad (14)$$

$$\theta^k = \theta^{k-1} - \frac{\alpha}{\sqrt{\hat{v} + \varepsilon}} \odot \hat{m} \quad (15)$$

Where m and v are the impulse and velocity vectors, respectively, β_1^t and β_2^t are the decay rates of the impulse and velocity.

This is a method that calculates the adaptive learning rate for each parameter. It stores both the decaying average of past gradients, similar to momentum, and the decaying average of past squared gradients, similar to RMSProp.

Thus, it combines the advantages of both methods.

This algorithm converges faster than other optimization algorithms and can perform well on noisy data.

3 Results and Discussion

Specific practical examples of the implementation of tomato disease recognition have been studied. For each of the optimizers, we will consider technologies for constructing models for identifying plant diseases.

We will solve the binary problem of classification, determining tomato disease - a healthy or diseased tomato leaf. A random scan of the data from the database is shown in Figure 1.

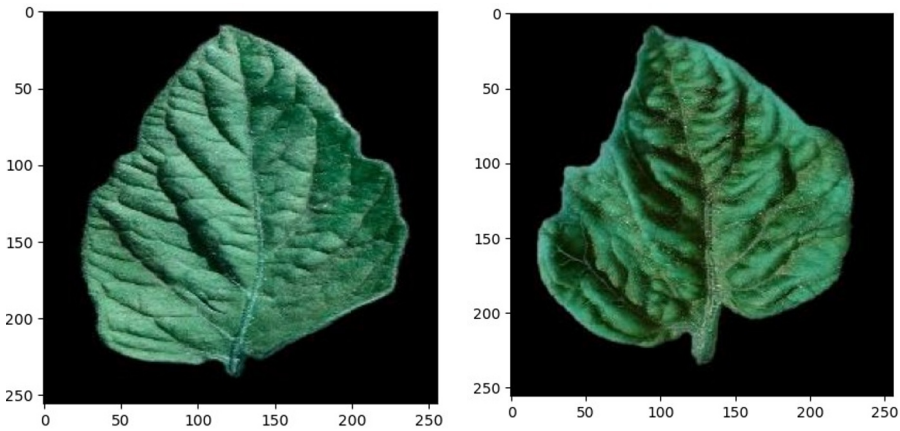


Fig. 1. Random selection of healthy and diseased tomato images.

Four models are built using various optimizers and the depth of neural networks, determined by formulas (5)-(15). In the data set there are a total of 3665 tomato leaves, of which 2074 are diseased plants and 1594 are healthy, i.e., divided into two classes. Let's set up the architecture of the first neural network CNN, model with the following parameters:

Conv2D -> MaxPool -> Conv2D -> MaxPool -> Dense -> Dense -> and as activation-nonlinearity functions, we take LeakyReLU for the internal layers, and the Sigmoid function for the output layer.

The optimizer used, stochastic gradient descent parameters, which will be updated according to formulas (6)-(7). We will feed images into the network in portions according to the rule batch size = 32 images in total with the training epoch epoch = 20. In this case, we will evaluate the model error as binary_crossentropy, and use accuracy as a metric. The obtained results of accuracy and error of the trained model by epoch:

```
Epoch 20/20
106/106 [=====] - 19s 183ms/step - loss: 0.0730 -
accuracy: 0.9750 - val_loss: 0.0912 - val_accuracy: 0.9533
```

In this case, we achieved a model accuracy of 95.3%. Let's look at the graphs of the accuracy and error of the model during training and testing.

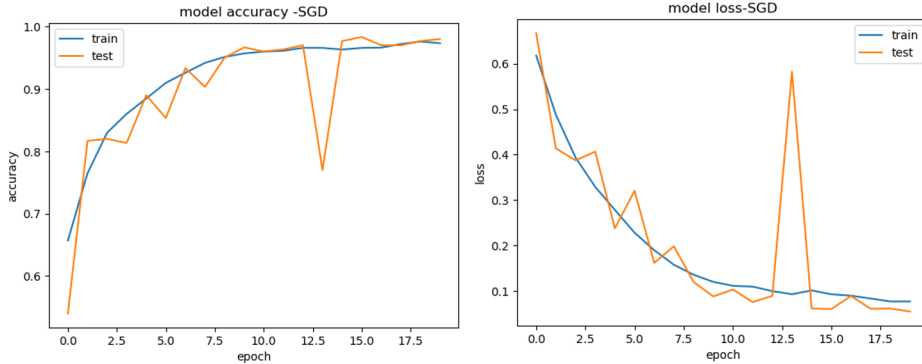


Fig. 2. Results of implementing a model with a gradient descent optimizer.

Now for the model we will use variations of gradient descent losses, namely, we will use the Adam optimizer with the following neural network architecture Conv2D -> MaxPool -> Conv2D -> MaxPool -> Dense -> Dense -> Sigmoid, and as activation-nonlinearity functions, we will take for internal layers LeakyReLU, and for the output layer the Sigmoid function. We use the Adam optimizer, an input package of 32 elements and a training epoch = 20. The accuracy and error of the model are as follows.

Epoch 20/20
 106/106 [=====] - 20s 185ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0635 - val_accuracy: 0.9833

And in graphical representation it looks like this:

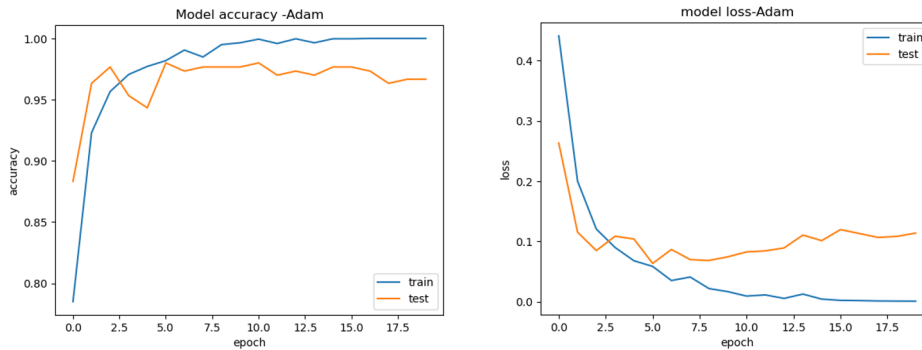


Fig. 3. Results using the Adam and Dropout optimizer.

From the implementation with this optimizer, in percentage terms, we moved forward 98.33% with a model error of 0.0013.

Let's consider the third model. Recently, using regularization methods for retrained models or Dropout technology, good model accuracy results have been obtained. In this case, we use the following model architecture

Conv2D -> MaxPool -> Conv2D -> MaxPool -> Dense -> Dropout -> Dense -> Dropout -> Dense -> Sigmoid

Here are the results of training the model with the above training parameters:

106/106 [=====] - 20s 184ms/step - loss: 0.0392 - accuracy: 0.9878 - val_loss: 0.0516 - val_accuracy: 0.9867

Let us now present the graphs of the trained model

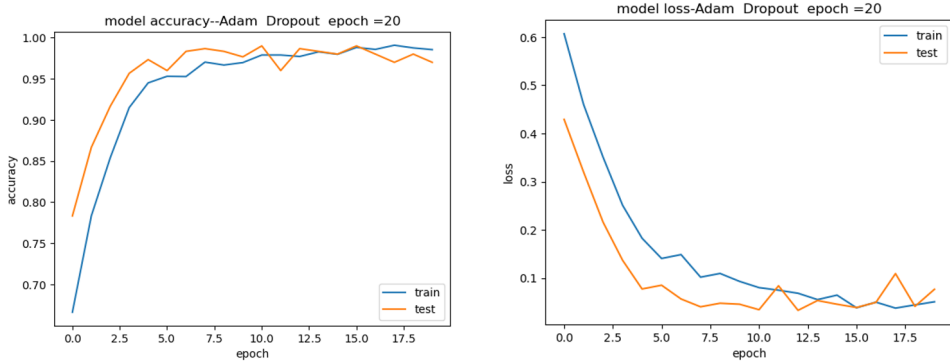


Fig. 4. Results of accuracy and loss during training and testing of models using the Adam optimizer and the regularization method.

In this case, we obtained an accuracy of 98.67%.

Fourth model. In the model considered below, to improve accuracy, we use the regularization method with a deeper neural network with the Adam optimizer and with a network training epoch equal to epoch = 30. Here is the result of the implementation:

Epoch 30/30
 106/106 [=====] - 21s 200ms/step - loss: 0.0536 -
 accuracy: 0.9872 - val_loss: 0.0158 - val_accuracy: 0.9933

At the same time, we obtain the highest model accuracy of 99.33%, which exceeds all models trained up to this point. Here is a graphical representation of the trained model.

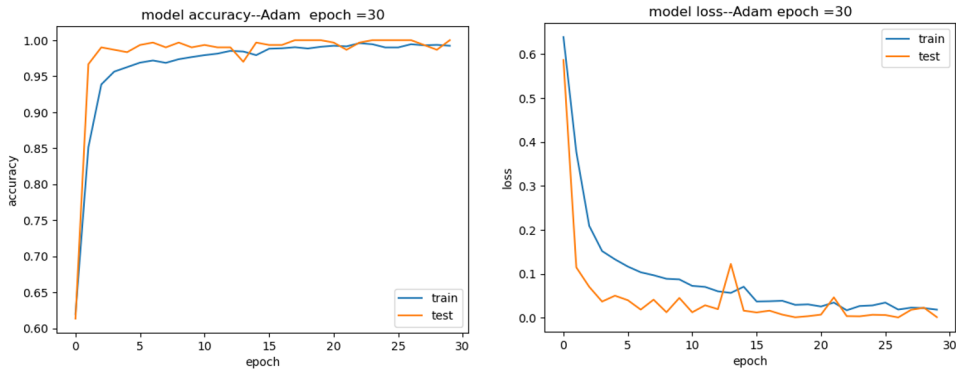


Fig. 5. Deep neural network results (30 epochs) accuracy and loss during model training and testing using Adam optimizer and regularization method.

4 Conclusion

The results of using stochastic gradient descent during implementation gave a result of 95.3%. However, for big data, this technology can show the best performance of models. The second one, using the Adam optimizer, showed a performance higher than SGD and amounted to 98.33% with a model error of 0.001. Using this neural network architecture gave us a noticeable increase in model accuracy. On test data, the model received an error

of 0.0635. In this case, we notice a slight overfitting of the model. In the third case, when we used the regularization method, we achieved an accuracy of 98.67% and a model error of 0.0516, which is not bad for such a small database. In most cases, different approaches to model building require different approaches. There can be many of them. Each newly constructed neural network architecture will have different results. In the fourth case, when we increased the depth of the neural network using regularization and, with increasing epoch, achieved excellent real results, the model error was `val_loss`: 0.0158 and `val_accuracy`: 0.9933. The general conclusions are as follows: when building models, each approach to building models has its own characteristics, this will be depends firstly on the choice of neural network optimizers, the choice of regularization methods, if there is overtraining, it is always present in deep learning and there is always the correct method for stopping model training, i.e. choice of era. Another important point when building models is the formulation of the problem, selection and tuning of hyperparameters, and selection of the most powerful transfer learning tools in deep learning.

References

1. J. You, X. Li, M. Low, "Deep gaussian process for crop yield prediction based on remote sensing data", In Thirty-First AAAI Conf.on Artificial Intel, 4559–4566 (2017)
2. S.H. Lee, C.S. Chan, S.J. Mayo, How deep learning extracts and learns leaf features for plant classification, *Pattern Recogn*, **71**, 1–13 (2017)
3. S.A. Tsaftaris, M. Minervini, H. Scharr, Machine learning for plant phenotyping needs image processing, *Trends Plant Sci.*, **21**, **12**, 989–91 (2016)
4. A. Fuentes, S. Yoon, D.S. Park, Deep learning-based techniques for plant diseases recognition in real-field scenarios, In: *Advanced concepts for intelligent vision systems* (2020)
5. D. Yang, S. Li, Z. Wang, MF-CNN: traffic flow prediction using convolutional neural network and multi-features fusion, *IEICE Trans Inf Syst*, **102**, **8**, 1526–36 (2019)
6. S.K. Sundararajan, B. Sankaragomathi, D.S. Priya, Deep belief cnn feature representation based content based image retrieval for medical images, *J.Med Syst*, **43**, **6**, 1–9 (2019)
7. P. Melnyk, Z. You, K. Li, A high-performance CNN method for offline handwritten chinese character recognition and visualization, *Soft Comput*, **24**, 7977–87 (2019)
8. J. Li, Y. Mi, Z. Ju, CNN-based facial expression recognition from annotated rgb-d images for human–robot interaction, *Int J Humanoid Robot*, **16**, **04**, 504–5 (2019)
9. G.E. Hinton, R. Salahutdinov, Reducing the dimensionality of data with neural networks, *Science*, **313**, 5786, 504–7 (2006)
10. W. Liu, Z. Wang, X. Liu, A survey of deep neural network architectures and their applications, *Neurocomputing*, **234**, 11–26 (2017)
11. R. Fergus, Deep learning methods for vision, CVPR 2012 Tutorial, www.cs.toronto.edu/~ranzato
12. Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, *IEEE Trans Pattern Anal Mach Intell*, **35**, **8**, 1798–828 (2013)
13. B.R. Sabitov, S. Biihsunova, A. Kashkaroeva, Deep learning Methods for Recognition of Orchard Crops, *IJCSNS*, **22**, **10** (2022)
<https://doi.org/10.22937/IJCSNS.2022.22.10.33>

14. B. Biibosunov, B.R. Sabitov, S. Biibosunova, Machine learning for crop yield forecasting, *Cybernetics and Physics*, **12**, **3** (2023)
15. B.R. Sabitov, A.D. Kartanova, K. Talant uulu, N.S. Seitkazieva, Modeling and forecasting tasks of agriculture based on machine learning, *E3S Web Conf.*, **380**, 15 (2023)