EC 601 Seminar Review
The application of Learning by Passing Tests Algorithm in Fruits Images Classification

Bolun Liu

Department of Electrical Computer Engineering, Boston University

1 Topic

In this report, the application of a machine learning algorithm will be discussed in detail. The goal of this research is using Learning by Passing Tests Algorithm(LPT) to solve fruits images classification problem.

Identification of fruits and vegetables is implemented in different areas. The most common areas are identifications in the retail business, and in areas where the purpose is to ease the harvest in the perspective of agriculture. In the retail business, the identification is mostly done manually by a cashier, or via the self-service systems in a store. And payment of fruits or vegetables in retail stores normally require them to be manually identified[1]. Therefore, complex and time consuming self-service systems may result in customers choosing another grocery store. Since customers are the reason companies survive, their satisfaction is the business's key to success[2]. So the study of allowing machine to identify and classify different types of fruits faster and more accurately means significant to the profitability of retail business.

I intend to do classification with Learning by Passing Test (LPT). This framework consists of a "learner" model and a "tester" model [6]. A huge number of images are divided into different subsets. I apply the learner model to one subset to make prediction and calculate prediction error. The error indicates the learning difficulty for this subset. If the model produces correct output, the tester will generate a harder test. Larger prediction error indicates the test is too difficult and the learner will learn this subset task iteratively until it gets a model with small error. The procedure continues until its convergence.

With LPT method introduced, the classification errors can be significantly reduced compared to original Neural Architecture Search method[6]. It shows that our method has high efficiency in searching better-performing architectures, by creating tests with increasing levels of difficulty and improving the learner through taking these tests. Learning capability and evaluation on the learner could be improved with better quality tests. Also, Acquiring a better architecture in the algorithm does not imply an increase in cost and structural complexity. With a stronger data encoder, the tester is able to better comprehend examples in the test bank and to make better decisions in the process of creating tests.

2 Main Contributions

2.1 CNN object recognition and classification

The paper "Fruit and Vegetable Identification Using Machine Learning for Retail Applications"[3], describes an approach of creating a system identifying fruit and vegetables in the retail market using images captured with a video camera attached to the system. To classify an object, different convolutional neural networks(CNN) have been tested and retrained. Although CNNs has got tremendous success in several object recognition and classification, it has unavoidable limitations

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

like it is unable to encode the position and orientation of object. Besides, CNNs lack of ability to be spatially invariant to the input data.

2.2 Fruit recognition based upon color and shape attributes

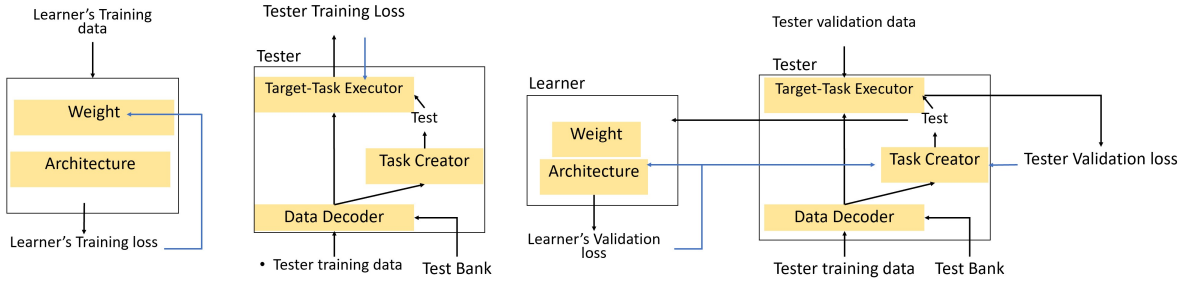
Fruits Shape Variation Analyzer for tomato and other plant species has been developed to analyze shape and size of tomato fruit and other plant species[4], such as Butternut squash, yellow squash, banana pepper, chili pepper, and strawberry. It can find the boundary of different species of plants accurately. This Analyzer can detect the shape variation for tomato and any other two dimensional fruit shape. Generally, it could measure fruit shape objectively by combine controlled vocabulary and mathematical descriptors into the Analyzer. The controlled vocabulary can describe a range species of fruit shape trait. The mathematical descriptors can calculate the fruit trait with a single equation for each trait, include fruit shape, fruit shape triangle and fruit size comprises fruit height, width, mass, area, and perimeter. This method accuracy is different based on the fruit image but the overall is around 80% to 85%[5]. The accuracy could be improved. There are some future works should be implement on the Analyzer in order to improve and enhance the functionality and flexibility of the recognition system for more widely usage. The system should be improved by extending its functions to process and recognize more variety of different fruit images.

2.3 Random decision forest algorithm as the classifier

Various classification methods are applied to classifier of fruit classification. Some researchers perform random decision forest algorithm on sorting papaya into different ripeness stage [7]. With different input features, Each decision tree processes and outputs the most probable class based on its model. Each decision tree could also learn the influence of input feature and thus assign importance to them. At the end, the algorithm selects the class with the most portions. This algorithm also perform well in foods quality evaluation [8]. however, Random decision forest can have a complicated structure and thus requires massive computation power.

3 Method

In this section, I describe the model of Learning by Passing Test(LPT), and introduce an optimization algorithm to solve LPT probem.



(a) Training process of LPT (b) Validation process of LPT

Figure 1: Illustration of learning by passing tests. The Black arrows denote the process of making predictions and calculating losses. The Blue arrows denote the process of updating learnable parameters by minimizing corresponding losses.

3.1 Mathematical setup

Table 1: Notation and Meaning of symbols

Notation	Meaning
T	Created test of tester
M	Intermediately-trained model of learner
R	Prediction error rate
T'	More challenging test created by tester
R'	New error rate achieved by M
M'	Newly-learned model of learner
R"	New error rate achieved by M'

In my research, I applied a novel learning framework called learning by passing tests (LPT). In this framework, there is a "learner" model and a "tester" model. The tester creates a sequence of "tests" with growing levels of difficulty. The learner tries to learn better so that it can pass these increasingly more-challenging tests. Given a large collection of data examples called "test bank", the tester creates a test T by selecting a subset of examples from the test bank. The learner applies its intermediately-trained model M to make predictions on the examples in T. The prediction error rate R reflects how difficult this test is. If the learner can make correct predictions on T, it means that T is not difficult enough. The tester will create a more challenging test $T^{'}$ by selecting a new set of examples from the test bank in a way that the new error rate $R^{'}$ achieved by M is larger than R. Given this more demanding test $T^{'}$, the learner re-learns its model to pass $T^{'}$, in a way that the newly-learned model $M^{'}$ achieves a new error rate $R^{''}$ on $T^{'}$ where $R^{''}$ is smaller than $R^{'}$. This process iterates until convergence.

In my framework, there is a learner model and a tester model, where the learner studies how to perform a target task J_1 such as classification, regression, etc. The eventual goal is to make the learner achieve a better learning outcome with the help of the tester. There is a collection of data examples called "test bank". The tester creates a test by selecting a subset of examples from the test bank. Given a test T, the learner applies its intermediately- trained model M to make predictions on T and measures the prediction error rate R. From the perspective of the tester, R indicates how difficult the test T is. If R is small, it means that the learner can easily pass this test. Under such circumstances, the tester will create a more difficult test $T^{'}$ which renders the new error rate $R^{'}$ achieved by M on $T^{'}$ is larger than R. From the learner's perspective, $R^{'}$ indicates how well the learner performs on the test. Given this more difficult test $T^{'}$, the learner refines its model to pass this new test. It aims to learn a new model $M^{'}$ in a way that the error rate $R^{''}$ achieved by $M^{'}$ on $T^{'}$ is smaller than $R^{'}$. This process iterates until an equilibrium is reached. In addition to being difficult, the created test should be meaningful as well. It is possible that the test bank contains poor-quality examples where the class labels may be incorrect or the input data instances are outliers. Using an unmeaningful test containing poor-quality examples to guide the learning of the learner may render the learner to overfit these bad-quality examples and generalize poorly on unseen data. To address this problem, we encourage the tester to generate meaningful tests by leveraging the generated tests to perform a target task J_2. Specifically, the tester uses examples in the test to train a model for performing J_2. If the performance P achieved by this model in conducting J_2 is high, the test is considered to be meaningful. The tester aims to create a test that can yield a high P.

In the learner' model, there are two sets of learnable parameters: model architecture and network weights. The architecture and weights are both used to make predictions in J_1. The tester's model performs two tasks simultaneously: creating tests and performing target-task J_1. The model has three learnable modules: data encoder, test creator, and target-task executor, where the test creator performs the task of generating tests and the target-task executor conduct J_1. The test creator and target-task executor share the same data encoder. The data encoder takes a data example d as input and generates a latent representation for this example. Then the representation is fed into the test creator which determines whether d should be selected into the test. The representation is also fed into the target-task executor which performs prediction on d during performing the target task J_2.

Table 2: Notation and Meaning of symbols

Notation	Meaning
A	Architecture of the learner
W	Network weights of the learner
E	Data encoder of the tester
C	Test creator of the tester
X	Target-task executor of the tester
$D_{ln}^{(tr)}$	Training data of the learner
$D_{tt}^{(tr)}$	Training data of the tester
$D_{tt}^{(val)}$	Validation data of the tester
D_b	Test bank
J_1	Target task conducted by learner
J_2	Target task performed by tester
L	Predictive loss of learner on the test created by tester
P	Performance achieved by tester model
d	Data example d

3.2 Model

The model is composed a learner model and a tester model. The learner learns and predict the task. Tester aims to create a test bank T and evaluate how the model works with T. This information is used by learner in the next iteration.

For the learner, the training loss was minimized to calculate the optimal network weight W. architecture A was remained unchanged because sizes of A are large and result in over-fitting. Then, training loss is minimized for data encoder E and target-task executor X for tester. Training loss is composed of two parts. The first part denotes the training loss for training on $D_{tt}^{(tr)}$ in task J_2. The second one is related with test creation on data bank D_b. all data in D_b are fed into data decoder to identity whether these data should be selected as test set. These create $/theta(C, E, D_b)$. Also, creator is not learned in this process to avoid over-fitting. The output will be optimized encoder $E^*(C)$ and target-task executor $X^*(C)$. In the final step, architecture in learner model is obtained by making prediction on test created in the previous step $\theta(C, E^*(C), D_b)$.

To learn test creator C which aim to create difficult test set where difficulty is defined by $L(A, W*(A), /theta(C, E^*(C), D_b))$ for test creator C, we maximize this function but also avoid degeneration from happening. Also, validation loss $L(E^*(C), X*(C), D_{tt}^{(val)})$ defines the quality of a test. Small validation loss indicates the test is meaningful.

3.3 Algorithm

In order to solve the LPT problem, I used a multi-level optimization framework to formalize LPT where the tester learns to select hard validation examples to promote the learner to make large prediction errors and then the learner improves its model to correct these prediction errors.
In phase one, the learner trains its network weights on the training set of task J1 with its architecture fixed. In phase two, the tester trains its data encoder and target-task executor on a created test to perform the target task J2, with its test creator fixed. In phase three, the learner updates its model architecture by minimizing the predictive loss on the test created by the tester; the tester updates its test creator by maximizing L and minimizing the loss on the validation set of J2.
To begin with, we update the architecture of the learner by using one-step gradient decent.
Approximating $W^*(A)$ using $W' = W - \xi_{\ln} \nabla_W L\left(A, W, D_{\ln}^{(\mathrm{tr})}\right)$ (where $\xi_{\ln}$ is a learning rate), the gradient can be calculated by,

$$\begin{aligned} &\nabla_A L\left(A, W^*(A), \sigma\right) \approx \\ &\nabla_A L\left(A, W - \xi_{\ln} \nabla_W L\left(A, W, D_{\ln}^{(\mathrm{tr})}\right), \sigma\right) = \\ &\nabla_A L\left(A, W', \sigma\right) - \xi_{\ln} \nabla_{A,W}^2 L\left(A, W, D_{\ln}^{(\mathrm{tr})}\right) \nabla_{W'} L\left(A, W', \sigma\right) \end{aligned} \tag{1}$$

Where $\sigma = \sigma\left(C, E^*(C), D_b\right)$. Secondly, to update the test creator of the tester, we can maximize the following objective using gradient,

$$L\left(A, W', \sigma\left(C, E', D_b\right)\right) / \left|\sigma\left(C, E', D_b\right)\right| - \lambda L\left(E', X', D_{tt}^{(\mathrm{val})}\right) \tag{2}$$

The derivative of the second term w.r.t. C can be expressed by:

$$\nabla_C L\left(E', X', D_{tt}^{(\mathrm{val})}\right) = \frac{\partial E'}{\partial C} \nabla_{E'} L\left(E', X', D_{tt}^{(\mathrm{val})}\right) + \frac{\partial X'}{\partial C} \nabla_{X'} L\left(E', X', D_{tt}^{(\mathrm{val})}\right) \tag{3}$$

Where,

$$\begin{array}{l} \frac{\partial E'}{\partial C} = -\xi_E \gamma \nabla_{C,E}^2 L\left(E, X, \sigma\left(C, E, D_b\right)\right) \\ \frac{\partial X'}{\partial C} = -\xi_X \gamma \nabla_{C,X}^2 L\left(E, X, \sigma\left(C, E, D_b\right)\right) \end{array} \tag{4}$$

Then using finite difference approximation, we have,

$$\begin{array}{l} \nabla_C L\left(E', X', D_{tt}^{(\mathrm{val})}\right) = \\ -\gamma \xi_E \frac{\nabla_C L\left(E^+, X, \sigma\left(C, E^+, D_b\right)\right) - \nabla_C L\left(E^-, X, \sigma\left(C, E^-, D_b\right)\right)}{2\alpha_E} - \gamma \xi_X \frac{\nabla_C L\left(E, X^+, \sigma(C, E, D_b)\right) - \nabla_C L\left(E, X^-, \sigma(C, E, D_b)\right)}{2\alpha_X} \end{array} \tag{5}$$

where $E^{\pm} = E \pm \alpha_E \nabla_{E'} L\left(E', X', D_{tt}^{(val)}\right)$ and $X^{\pm} = X \pm \alpha_X \nabla_{X'} L\left(E', X', D_{tt}^{(val)}\right)$.

For the first term in Eq.(2), by using chain rule to calculate its derivative w.r.t C, we get,

$$\nabla_C L\left(A, W', \sigma\left(C, E', D_b\right)\right) = \frac{\partial E'}{\partial C} \nabla_{E'} L\left(A, W', \sigma\left(C, E', D_b\right)\right) \tag{6}$$

Which can be simplified by Substituting Eq.(4) into Eq.(6) and approximating

$\nabla_{C,E}^2 L\left(E, X, \sigma\left(C, E, D_b\right)\right) \times \nabla_{E'} L\left(A, W', \sigma\left(C, E', D_b\right)\right)$ with

$\frac{1}{2\alpha_E}\left(\nabla_C L\left(E^+, X, \sigma\left(C, E^+, D_b\right)\right) - \nabla_C L\left(E^-, X, \sigma\left(C, E^-, D_b\right)\right)\right)$.

In this end, the derivative of $\left|\sigma\left(C, E', D_b\right)\right| = \sum_{d \in D_h} f\left(d, C, E'\right)$ w.r.t C can be written as,

$$\sum_{d \in D_b} \nabla_C f\left(d, C, E'\right) + \frac{\partial E'}{\partial C} \nabla_{E'} f\left(d, C, E'\right) \tag{7}$$

Thirdly, updating the data encoder and target-task executor of the tester, using the equations below.

$$\begin{array}{l} E' = E - \xi_E \nabla_E \left[L\left(E, X, D_{tt}^{(\mathrm{tr})}\right) + \gamma L\left(E, X, \sigma\left(C, E, D_b\right)\right)\right] \\ X' = X - \xi_X \nabla_X \left[L\left(E, X, D_{tt}^{(\mathrm{tr})}\right) + \gamma L\left(E, X, \sigma\left(C, E, D_b\right)\right)\right] \end{array} \tag{8}$$

where ξ_E and ξ_X are learning rates.

Finally, updating the weights of the learner by descending $\nabla_W L\left(A, W, D_{ln}^{(tr)}\right)$ until the architecture variables get converged.

ALGORITHM Learning by passing tests

While no converged do

1.Update the architecture of the learner by descending the gradient calculated in Eq.(1)

2.Update the test creator of the tester by ascending the gradient calculated in Eq.(2)-(7)

3.Update the data encoder and target-task executor of the tester using Eq.(8)

4.Update the weights of the learner by descending $\nabla_W L\left(A, W, D_{ln}^{(tr)}\right)$

end

3.4 Discussion of the method

With application of the LPT method, the classification errors can be significantly reduced compared to original Neural Architecture Search method[6]. With a stronger data encoder, the tester can better

understand examples in the test bank and can make better decisions in creating tests. Tests with better quality can more effectively evaluate the learner and promote its learning capability. Moreover, the LPT is able to search better-performing architectures without a significant increase in network size and search cost. Although LPT might perform well with large amount of data. Its complex model limits its ability for solving relatively small data set. More applications are required to verify the accuracy and generalization to different schemes.

4 Experiment

4.1 Datasets

Data set used in the experiments is called Fruit 360. It contains 67692 training images and 22688 testing images, from 131 classes. We choose 10 fruit classes from it including Apple, Banana, Lemon, Lychee, Mango, Orange, Peach, Pear, Strawberry and Watermelon in figure 1, which consist of a 4k8 training set and a 1k5 testing set. Then we split the training set into a new 2.4k training set and a 2.4k validating set. The image size is transformed from 100*100 to 32*32 to fit in the network. During architecture search, the training set is used as the training data $D_{ln}^{(tr)}$ of the learner ln and the training data $D_{tt}^{(tr)}$ of the tester. The validation set is used as the test bank Db and the validation data $D_{tt}^{(val)}$ of the tester. During architecture evaluation, the combination of tt the training data and validation data is used to train the large network stacking multiple copies of the searched cell.

Figure 2: Fruit classes

4.2 Experiment Setup

The framework is a general one that can be used together with any differential search method. Specifically, we apply our framework to a NAS method: DARTS. The candidate operations include 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. In LPT, the network of the learner is a stack of multiple cells, each consisting of 7 nodes. For the data encoder of the tester, we tried ResNet-18 and ResNet-50. For the test creator and target-task executor, they are set to one feed-forward layer. and are both set to 1. For our data set, during architecture search, the learner's network is a stack of 8 cells, with the initial channel number set to 16. The search is performed for 50 epochs, with a batch size of 8. The hyper-parameters for the learner's architecture and weights are set in the same way as DARTS. The data encoder and target-task executor of the tester are optimized using Stochastic gradient descent with a momentum of 0.9 and a weight decay of 3e-4. The initial learning rate is set to 0.025 with a cosine decay scheduler. The test creator is optimized with the Adam optimizer with a learning rate of 3e-4 and a weight decay of 1e-3. During architecture evaluation, 20 copies of the searched cell are stacked to form the learner's network, with the initial channel number set to 36. The network is trained for 50 epochs with a batch size of 8. The experiments are performed on a single RTX2060 Super gpu.

4.3 Results and View of Work

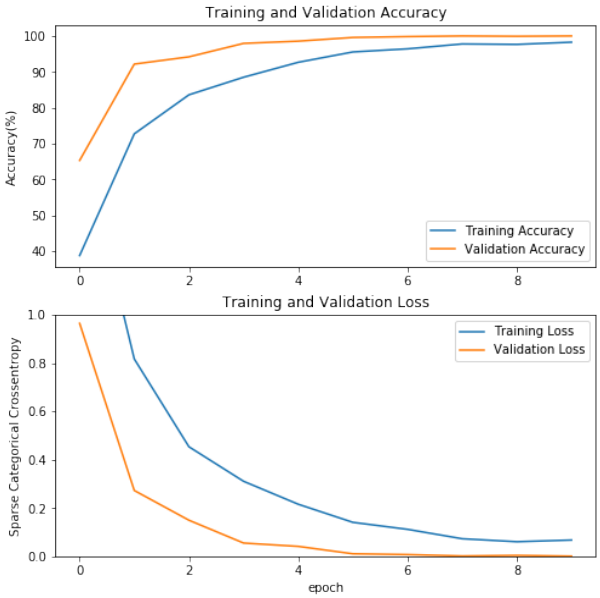


Figure 3: Results of training

The Figure above shows the line charts of the results. The first chart represents the training and validation accuracy of our model. From the chart, it can be seen that our model's training accuracy and validation accuracy increase as the epoch becomes larger. The maximum value of the training accuracy is 98.27%. The maximum value of the validation accuracy is 100%. The second chart represents the training and validation loss of the model. From the chart, it can be seen that the model's training loss and validation loss decrease as the epoch becomes larger. The minimum value of the training loss is 6.74e-02. The minimum value of the validation accuracy is 7.85e-04.

Table 3: Comparison on classification error of Fruit360 between LPT and other methods

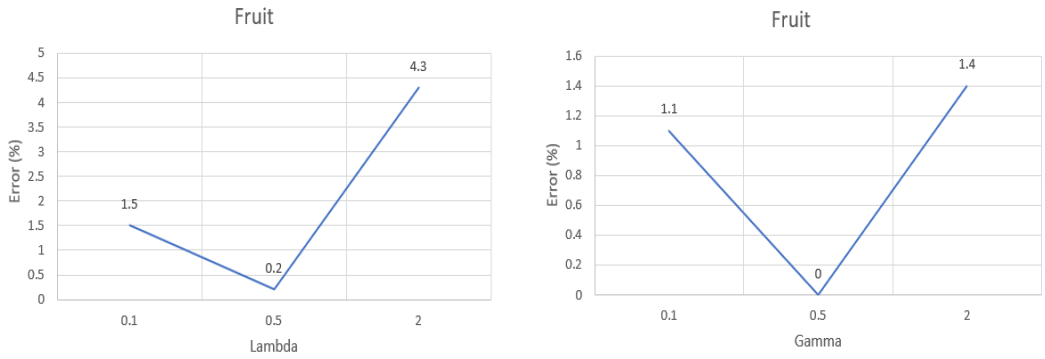
Method	Error(%)
LPT(our)	0.08
CNN	4.52
NN	1.47
VGG-16	6.65
Inception	15.33
Resnet	23.89

Table above shows the comparison of different methods' testing accuracy on the Fruit 360 dataset. These methods include our own method and methods used by other people on Kaggle. It can be seen that the testing accuracy of our method, Learing by Passing Test(LPT), has the lowest error rate among these methods. This also demonstrates the efficiency of the LPT method in searching the best architecture.

The classification error of our dataset could reach 0 after 25 epochs, which is relatively high compared with cifar10 and Cifar100. This is because it is relatively easy to recognize each class from fruit360 (our dataset), since the proportion of each objects are the same, the characteristics of each class are relatively simple to be classified, furthermore, there are almost no noise since the objects are not in a scene of daily life, instead, a white sheet of paper is placed behind as background.

In LPT method, by passing the test with growing level of difficulty, the architecture is kept improving by the learner. These tests play a important role in leading the learner to identity the

weakness on their architecture and to improve their architecture. From the test bank, the tester selects a subset of difficult examples to evaluate the learner. The method creates a new test based on the learner's performance in the previous round. The tester chooses a subset of difficult instances to assess the learner. This new test brings a bigger challenge to the learner hence then inspires the learner to develop its architecture, then it can pull through the new challenge.



(a) How errors changes with Lambda

(b) How errors changes with Gamma

Figure 4: Errors variation with Lambda or Gamma

It can be seen from Figure 4.(a) that how classification errors varies as λ increases. As shown, when λ increases from 0.1 to 0.5, the error decreases. But further increasing λ will force the error to increase. In terms of the testers, the aim of λ is to find a balance between difficulty and significance of the tests. In this case, on one hand, increasing of λ inspire the testers to create more significant tests. With more meaningful tests, learner can be assessed more reliably. One the other hand, if λ is too large, the testers will emphasis on improving the significance of the tests then lose sight of the difficulty of them. Lacking sufficient difficulty, the tests may not be compelling enough to motivate the learner for improvement.

It can be seen from Figure 4.(b) that how classification errors varies as γ increases. Ad shown, when γ increases from 0.1 to 0.5, the error decreases. But further increasing γ will force the error to increase. On one hand, under a larger γ, the created test plays a more important role in training the tester to perform the target task, which inspire the testers to create more meaningful tests. On the other hand, if γ goes too large, the created test will play a leading role in training which brings the following risk: if the test does not make sense, a low-quality data-encoder will be generated which could further reduce the quality of test creation.

5 Conclusion and Next Steps

In this research, the Learning by passing tests(LPT) is utilized for the fruit classification. LPT is a new learning approach which consist of a tester model which create test with appropriate difficulty and a learner model which learn from training data and test feedback. Multi-level optimization framework is used for solving convergence problem and update tester and learner parameters. The model is applied with LPT on Fruit 360 data set. The model achieves a high learning rate and it also shows a high accuracy when testing. Compared with other methods, the LPT method gets the highest testing accuracy, which illustrates the efficiency of LPT method in searching for a better architecture. Training parameters like λ and γ are evaluated with different values to obtain the optimal training and validation accuracy and test errors, it shows that a higher λ or γ could improve the architecture while further increasing them will lead to a degradation to the performance of the architecture.

References

Rojas-Aranda J.L., Nunez-Varela J.I., Cuevas-Tello J.C., Rangel-Ramirez G. (2020) Fruit Classification for Retail Stores Using Deep Learning. In: Figueroa Mora K., Anzurez Marín J., Cerda J.,

Carrasco-Ochoa J., Martínez-Trinidad J., Olvera-López J. (eds) Pattern Recognition. MCPR 2020. Lecture Note3s in Computer Science, vol 12088. Springer, Cham.

F. Femling, A. Olsson and F. Alonso-Fernandez, "Fruit and Vegetable Identification Using Machine Learning for Retail Applications," 2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Las Palmas de Gran Canaria, Spain, 2018, pp. 9-15, doi: 10.1109/SITIS.2018.00013.

Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

Marin Talbot Brewer, L.L., Kikuo Fujimura, Nancy Dujmovic, Simon Gray, Esther van der Knaap, Development of a Controlled Vocabulary and Software Application to Analyze Fruit Shape Variation in Tomato and Other Plant Species. Plant Physiology, 2006. 141: p. 15-25.

A. R. Jimenez, R.C., and J. L. Pons,, A survey of Computer Vision Methods for Locating Fruit on Trees. ASAE,, 2000. 43: p. 1911-1920.

Xuefeng Du, Pengtao Xie. Learning by Passing Tests, with Application to Neural Architecture Search. arXiv:2011.15102 (2020).

L. F. S. Pereira, S. Barbon, N. A. Valous, D. F. Barbin, Predicting the ripening of papaya fruit with digital imaging and random forests, Computers and electronics in agriculture 145 (2018) 76-82.

P.M.Granitto, F. Gasperi, F. Biasioli, E. Trainotti, C. Furlanello, Modern data mining tools in descriptive sensory analysis: a case study with a random forest approach. (2007) Food Qual. Prefer. 18, 681–689.

Kaggle.com. n.d. fruits-360-cnns-v2. [online] Available at: <https://www.kaggle.com/aarongomez/fruits-360-cnns-v2> [Accessed 15 March 2021].

Kaggle.com. n.d. Acc: Fruits Recognition Using NNnline] Available at: <https://www.kaggle.com/muhammadimran112233/99-acc-fruits-recognition-using-nn> [Accessed 15 March 2021].

Kaggle.com. n.d. Fruits - 360-Identification[92% accuracy]. [online] Available at: <https://www.kaggle.com/kundankumarmandal/fruits-360-identification-92-accuracy> [Accessed 15 March 2021].