Assignment 4 & 5 : CNN Classifier

Team Name : The Salvator Brothers
School of Engineering and Applied Science, Ahmedabad University

Computer Vision - CSP502, Winter 2021

1st Kirtan Kalaria
AUL202005
Ahmedabad University
Ahmedabad, India
kirtan.k@ahduni.edu.in

2nd Manav Vagrecha
AU1841022
Ahmedabad University
Ahmedabad, India
manavkumar.v@ahduni.edu.in

CONTENTS

***Abstract*—In this assignment, our objective is Classification of Citrus Leaves Data using CNN classifier. Here, we are comparing performances of different optimizers and hyper-parameters on the basis of different metrics like Accuracy, Precision, Recall.**

***Index Terms*—CNN classifier, Citrus Leavers Dataset, Optimizers, Performance Metrics,**

I. MOTIVATION AND BACKGROUND

Image classification is amongst the fundamental tasks handled by CNN. The goal in classification is to assign a label to an image. The classification comprehends the image. In this assignment, the objective is to understand, design, and implement a CNN classifier. We must not just implement the CNN classifier but understand it as well.

II. DATA INFORMATION

A. Description

The original dataset contains 759 images of healthy and unhealthy citrus fruits and leaves. However, as of now the owners only export 594 images of citrus leaves with the following 4 labels: Black Spot, Canker, Greening, and Healthy. The exported images are in PNG format and have the dimension 256×256.

B. Resources

Following are few resources from where we can get the dataset

- https://www.tensorflow.org/datasets/catalog/citrus_leaves
- https://data.mendeley.com/datasets/3f83gxmv57/2
- https://www.kaggle.com/dtrilsbeek/citrus-leaves-prepared

III. DATA PREPROCESSING

ImageDataGenerator was used to generate training, validation and testing data [60%,20%,20%] from the dataset. This allowed us to randomly augment the training data by

- zooming in and out (30%),
- rotating(±180°),
- height and width shifting(30%),
- horizontal and vertical flipping.

Finally, both training and validation images were re-scaled to [0,1].

IV. STRUCTURAL ANALYSIS OF MODEL

Layers:

1) Convolution (16 filters, 3×3 kernel, ReLU)
2) MaxPooling (/2)
3) Convolution (32 filters, 3×3 kernel, ReLU)
4) MaxPooling (/2)

5) Convolution (64 filters, 3×3 kernel, ReLU)
6) MaxPooling (/2)
7) Convolution (64 filters, 3×3 kernel, ReLU)
8) MaxPooling (/2)
9) Convolution (64 filters, 3×3 kernel, ReLU)
10) MaxPooling (/2)
11) Flatten [4092 units]
12) Dense (512, ReLU)
13) Dense (4, Sigmoid)

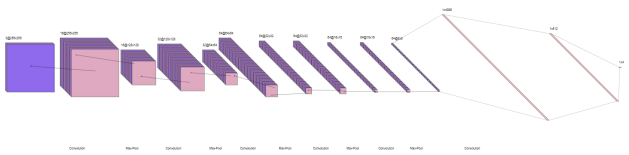

Fig. 1. Architecture of Model

Inspiration for our architecture is drawn from Alexnet. We have a simple dataset so it was important to control the number of parameters in order to control variance. This particular model has been reached after more than a hundred tries and iterative steps in order to improve it.

All tries at adding regularisation, changing activation functions, ordering of layers, number of layers and more, have resulted in very unfavorable accuracies of (< 4). This is a sweet spot we had reached.

V. Training

A. Optimizers

Type of Optimizers with their Mathematical Formulations

- Gradient Descent with momentum

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw$$

$$v_{db} = \beta \cdot v_{dw} + (1 - \beta) \cdot db$$

$$W = W - \alpha \cdot v_{dw}$$

$$b = b - \alpha \cdot v_{dw}$$

- RMSprop

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw^2$$

$$v_{db} = \beta \cdot v_{dw} + (1 - \beta) \cdot db^2$$

$$W = W - \alpha \cdot \frac{dw}{(\sqrt{v_{dw}} + \epsilon)}$$

$$b = b - \alpha \cdot \frac{db}{(\sqrt{v_{db}} + \epsilon)}$$

- Adam

$$\theta_{t+1} = \theta_t - \frac{\eta \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

where

$$m_t = (1 - \beta_1)g_t + \beta_1 m_{t-1}$$

$$v_t = (1 - \beta_2)g_t^2 + \beta_2 v_{t-1}$$

We have tried Stochastic Gradient Descent (SGD) with momentum, Adam and RMSprop. SGD performed the worst and RMSprop performed the best.

B. Hyper-parameters

- Number of Epochs
- Batch Size

We have kept the number of epochs as 35 as to neither underfit nor overfit the model, and the ratio of batch sizes of training and validation data (32 and 8) close to the ratio of their share in the dataset (3).

1) Adam:

- Learning Rate / Step Size (α) : The proportion that weights are updated (e.g. 0.001). Larger values (e.g. 0.3) results in faster initial learning before the rate is updated. Smaller values (e.g. 1.0E-5) slow learning right down during training
- Exponential decay rate for the first moment (β_1) : (e.g. 0.9).
- Exponential decay rate for the second moment (β_2) (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient (e.g. computer vision and NLP problems).
- ϵ is a very small number to prevent any division by zero in the implementation (e.g. 10E-8).

2) RMSProp:

- learning rate (α) RMSprop uses an adaptive learning rate instead of treating the learning rate as a hyperparameter. This means that the learning rate changes over time.
- ρ : ideally, it is around 0.9,
- momentum : the most widely used value is 0.9
- ϵ = 1e-07
- We selected $\alpha = 4 \times 10^{-4}$ and kept other parameters to be default.

3) Stochastic Gradient Descent:

- learning rate (α)
- momentum
- Here, After tuning the hyper-parameters, we have selected learning rate = 0.01 and momentum = 0.2

VI. Results

A. Explanation

The results from testing the model with testing data, using different optimizers are as follows:

We can see that the maximum accuracy reached by the two best optimizers is about 80%, which is not bad considering the small size of dataset. From our observation, a sufficiently low learning rate along with a large number of epochs result

.	Metrics→	Loss	Accuracy	Precision	Recall
Optimizers ↓	.	.	.	.	.
SGD		0.9554	0.4832	0.6321	0.3408
Adam		0.5202	0.7849	0.8049	0.7374
RMSprop		0.4764	0.8156	0.4140	0.9944

TABLE I
OPTIMIZERS AND DIFFERENT PERFORMANCE METRICS

in the most effective validation and testing accuracy. Learning curves served as a guide to determine the proper learning rate. In case of RMSprop, we can see that the learning curves are quite 'jumpy' or fluctuating, however, overall, with increase in number of epochs, the validation loss decreases and validation accuracy increases, so we know that the model is learning.

A possible solution was to further decrease the learning rate and use an optimizer that utilizes momentum. However, when that was tried, we found that it results in relatively poorer loss and accuracy and hence the model doesn't train as well as desired. On the other hand, the decrease in learning rate even by a factor of 5 resulted in the weights to get stuck at some local minimum. Therefore, even with the volatility in RMSprop, the testing metrics are satisfactory, hence it is a good choice and learning can be considered satisfactory.

Adam is the also a very good option and yields balanced metrics. It has accuracy close to RMSprop yet significantly higher precision and recall than the latter. Overall, we can say that Adam is the best choice of optimizer in this case.

VII. CONCLUSION

Here as shown in the table, Adam and RMSprop have performed almost the similar where Adam seems to be the best choice as all the metrics have good and balanced values. Our final testing accuracy is 80%.

REFERENCES

[1] V. Fung, "An overview of resnet and its variants," 17-Jul-2017. [Online]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035. [Accessed: 08-Mar-2021].

[2] "Vgg16 - convolutional network for classification and detection," 24-Feb-2021. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/. [Accessed: 08-Mar-2021].

[3] "Keras Conv2D: Working with CNN 2D Convolutions in Keras," MissingLink.ai. [Online]. Available: https://missinglink.ai/guides/keras/keras-conv2d-working-cnn-2d-convolutions-keras/. [Accessed: 08-Mar-2021].

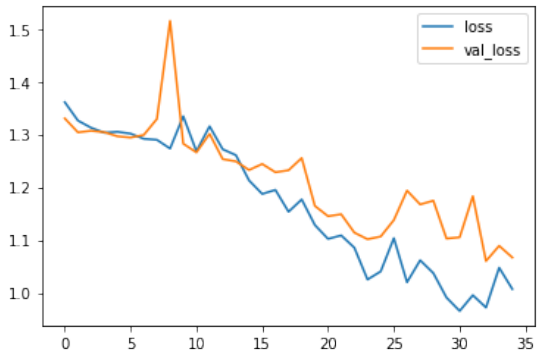

Fig. 2. Loss Curve - Using SGD

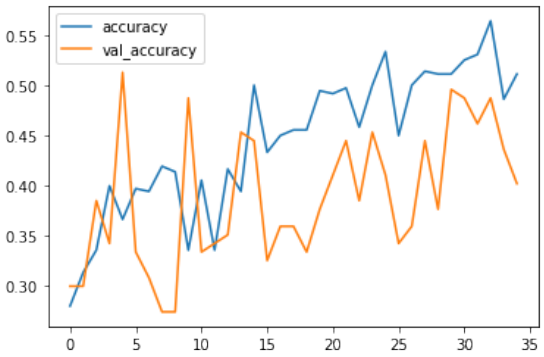

Fig. 3. Accuracy Curve - Using SGD

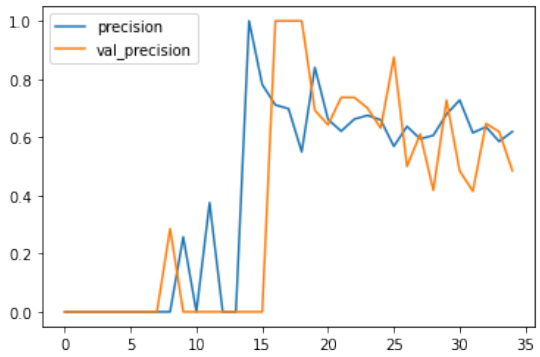

Fig. 4. Precision Curve - Using SGD

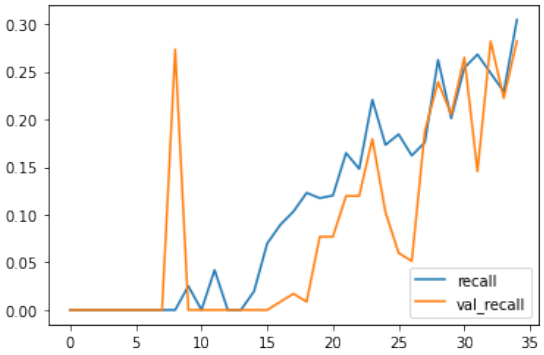

Fig. 5. Recall Curve - Using SGD

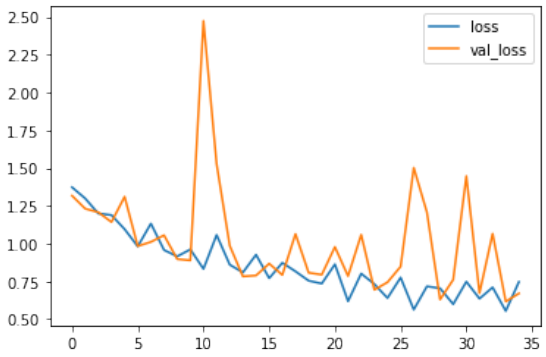

Fig. 6. Loss Curve - Using RMSProp

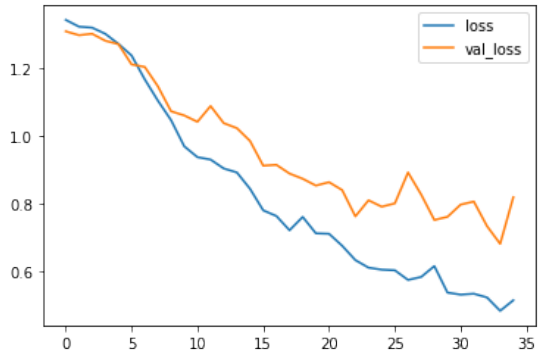

Fig. 10. Loss Curve - Using Adam

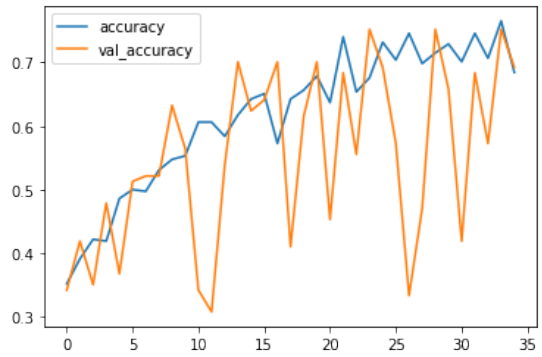

Fig. 7. Accuracy Curve - Using RMSProp

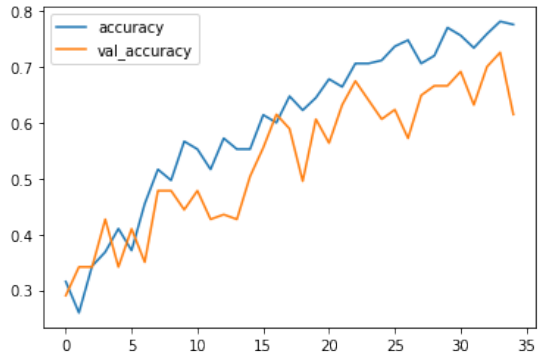

Fig. 11. Accuracy Curve - Using Adam

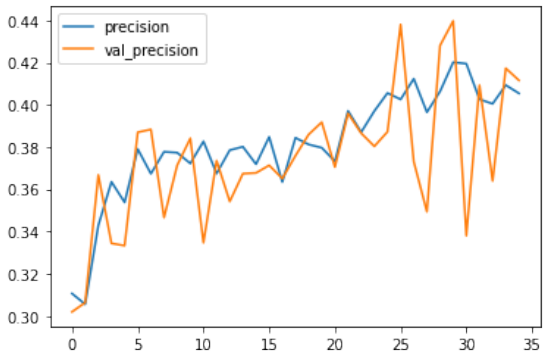

Fig. 8. Precision Curve - Using RMSProp

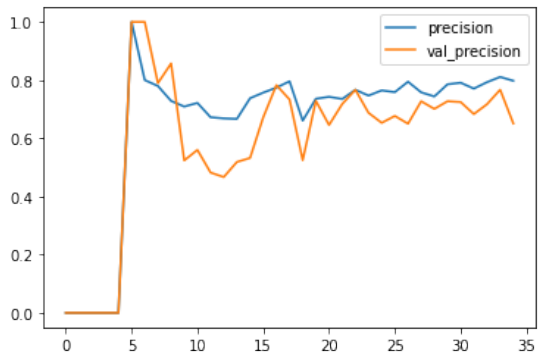

Fig. 12. Precision Curve - Using Adam

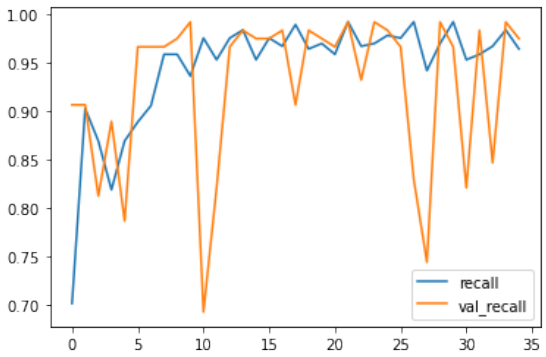

Fig. 9. Recall Curve - Using RMSProp

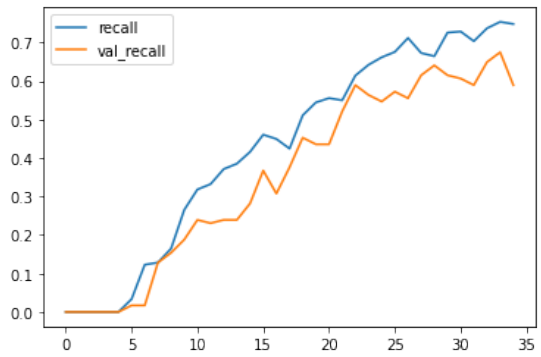

Fig. 13. Recall Curve - Using Adam