Multi-class weather classification using convolutional neural networks

Amirmohammad Shahbandegan, Computer Science Department, Lakehead University

I. Introduction

The aim of this assignment is to implement a convolutional neural network to classify the weather status of a given set of images.

A. Dataset

The data is composed of 1125 unobstructed images of different weather which are labeled as *cloudy*, *sun rise*, *sun shine*, and *rainy* [1]. A few sample images from the dataset are shown in Figure 1 and the distribution of classes can be found in Table I.

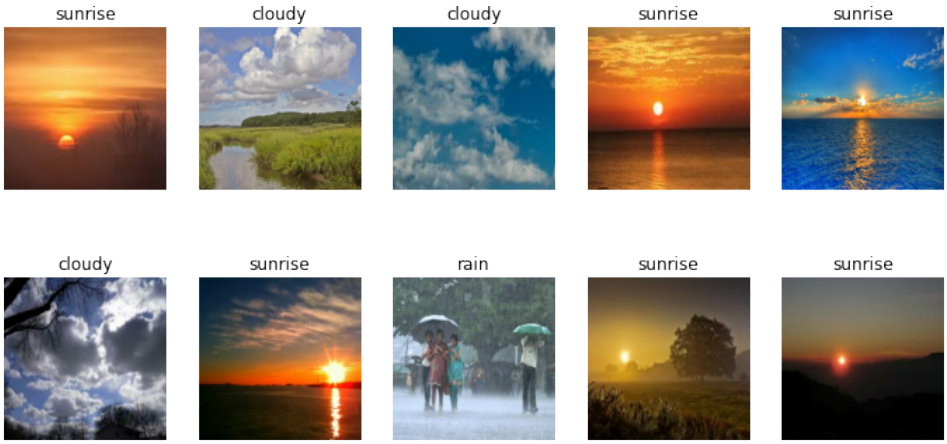

Fig. 1. Sample images drawn from the dataset and their corresponding class names.

TABLE I
DISTRIBUTION OF IMAGES AMONG THE FOUR CLASS LABELS

	Cloudy	**Sunshine**	**Rainy**	**Sunrise**
Number of Instances	300	235	215	357

B. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models most commonly used in image processing tasks. These networks gained a high popularity in the past decade for their ability in reducing the number of network parameters [2]. The most important element of a convolutional neural network is the convolution layer. The convolution for one pixel in the next layer is calculated according to Formula 1

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{1}$$

C. VGG16 Architecture

VGG16 [3] is a deep CNN architecture proposed by Simoyan et. al. in 2014. It consists of 16 weight layers and an input size of 224×224 RGB pixels. The architecture of this network is depicted in Figure 2. It consists of convolution, pooling and fully connected layers. This model is chosen for this assignment because of its high performance and simplicity compared to newer architectures that require a higher computational power.

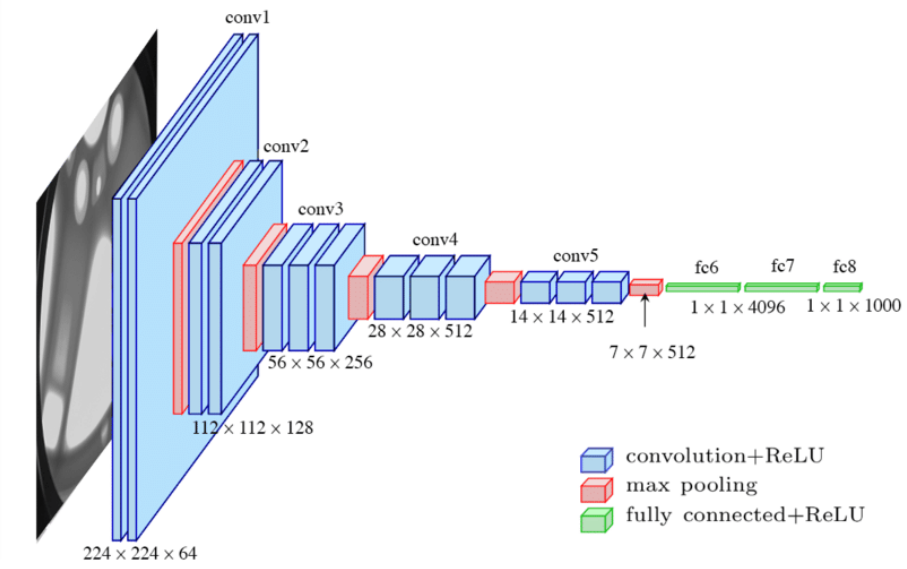

Fig. 2. VGG16 network architecture with convolution, pooling, and fully connected layers [4].

TABLE II
PERFORMANCE OF TRAINED MODELS

	Optimizer	Learning Rate	Epochs	Data Augmentation	Early Stopping	Training Accuracy	Testing Accuracy
Model A	adam	0.001	100	No	No	100.00%	92.28%
Model B	adam	0.001	100	Yes	No	93.44%	90.21%
Model C	adam	0.001	100	Yes	Yes	94.22%	**92.58%**

II. METHODOLOGY

To asses the quality of the designed network based on the VGG16 architecture, the dataset is randomly split into two groups, 70% for training and 30% for testing. Three different models were experimented to analyze the effect of regularization methods such as data augmentation and early stopping.

The first model (Model A) follows the basic VGG16 model as explained in Section I. Second model (Model B), adds five data augmentation layers before the first layer of model A. These augmentation layers are used in this model: random flip, random rotation, random zoom, and random contrast. The third model (Model C) is based of Model B. Model C utilizes early stopping with a patience of 20 epochs to avoid overfitting whereas model B does not. The total number of trainable parameters in all three models is $134,268,738$. The optimizer used in this work is the Adam optimizer [5] with a fixed learning rate of 0.001.

The models are implemented using Python programming language and Tensorflow library on Google Colaboratory environment. The code used to run the experiments are submitted along with this report.

III. EXPERIMENTAL RESULTS

All three models are trained with the same training data for 100 epochs and the accuracy of the models are then calculated using a common testing set so that the results are comparable. A summary of the performance of the models and their respecting hyperparameters are shown in Table II. The learning curves of models A, B, and C are depicted in figures 3, 4, and 5, respectively. It can be seen that model A started to overfit the training data after almost 50 epochs. Figure 3 shows that after 50 epochs the validation loss begins to increase whereas the training loss keeps on declining, implying that overfitting is happening. In contradiction, no signs of overfitting can be seen in model B. As it can be seen on Figure 4, the validation loss keeps on declining with the training loss showing that the data augmentation layers used in this model are properly regularizing the model and avoiding the overfitting. Model C closely follows model B since the only difference between the two models is that model C incorporates an early stopping method to check and stop the training should overfitting happen. However, since the data augmentation method used in model B is working well, overfitting will not happen in the first 100 epochs of this

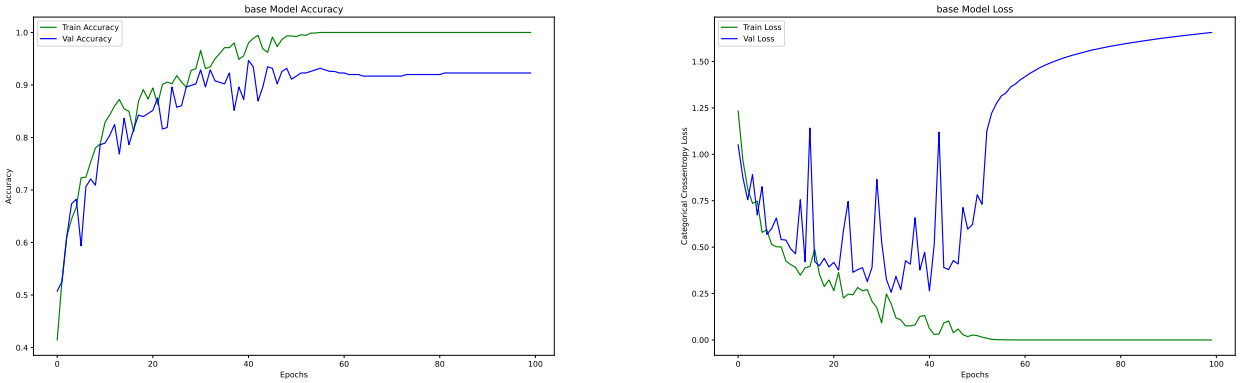

Fig. 3. Learning curves for model A.

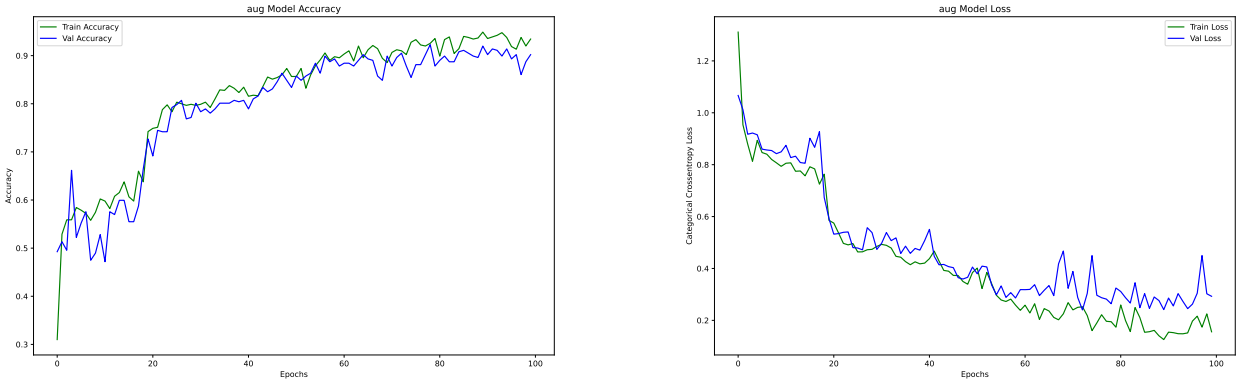

Fig. 4. Learning curves for model B.

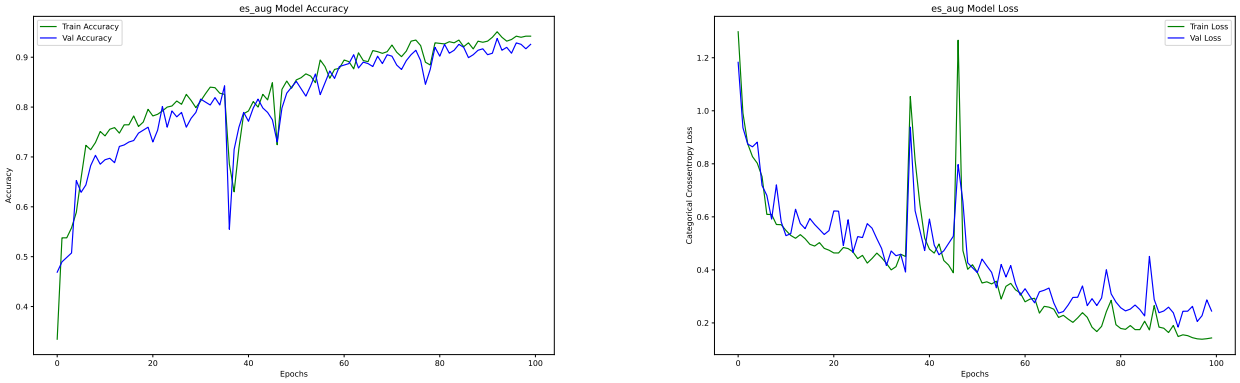

Fig. 5. Learning curves for model C.

model and the training does not stop early. There are no significant differences between model B and C in the first 100 epochs of training. Comparing Figures 4 and 5 confirms that these methods are both trained in the same way.

IV. Conclusion

This work experimented with three different models to analyze the overfitting issue in deep neural networks and how it can be overcome by using regularization methods such as data augmentation and early stopping. The experimental results show that by utilizing synthetic data, the network can learn a generalizable pattern in the image data and reduce the chance of overfitting. To further improve this work, it is possible to analyze how other regularization methods such as dropout, L1, and L2 regularization can affect the performance and generalizability of the model. It is also possible to run models B and C for more than 100 epochs to see when will overfitting happen in these models and how early stopping can stop the training when it happens.

References

[1] A. G. Oluwafemi and W. Zenghui, "Multi-class weather classification from still image using said ensemble method," in *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*. IEEE, 2019, pp. 135–140.

[2] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] M. Ferguson, R. ak, Y.-T. Lee, and K. Law, "Automatic localization of casting defects with convolutional neural networks," 12 2017, pp. 1726–1735.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.