A Machine Learning Model for Cloud Classification

Caroline Wendt

MATH 469 • Spring 2020 • Department of Mathematics • Colorado State University

Significance

In the the nervous system, a neuron is the fundamental unit of biological computation that detects, processes, and transmits chemical and electrical signals within an interconnected organic structure. As the modern technological landscape is increasingly reliant on data-driven processes to integrate information, devices have not only evolved as an extension of the human nervous system in a metaphorical sense; however, machines are the foremost intermediary between human ingenuity and invention. Naturally, an analog to the human nervous system has been developed in programming; an artificial neural network (ANN) is a multilayered computing model that enables a machine to learn from observational data. ANNs provide a versatile and accurate method that is widely used in data analytics (e.g., classification and regression) with applications that extend to image and speech recognition as well as data processing. Here, we examine an application to atmospheric science. Further, deep neural networks–albeit more difficult to train than shallow networks–are a powerful tool in practice. A Convolutional Neural Network (CNN) is a particular type of deep network that considers the spatial structure of images; CNN's sophisticated architecture (e.g., local receptive fields, shared weights, pooling) enables accurate image classification. The present project discusses the implementation of a CNN in R to classify cloud images with implications relevant to climate related research and machine learning with smaller datasets.

Theory

Neuron model and mathematical foundations

The subsequent equations adhere to the following notation:

- I: the number of neurons in a layer (index i)
- J: the number of weights in a neuron (index j)
- K: the number of layers in a network (index k)
- x: input (analogous to chemical and electrical signals; outputs from other neurons)
- w: weight assigned to each connection between neurons in consecutive layers (analogous to connection strength)
- b: bias for inactivity based on a threshold assigned to each connection (analogous to an effect on the firing threshold of an active neuron)

As an information processing mechanism, an ANN is comprised of numerous layers of individual neurons that relate through activations; ultimately, the system will integrate inputs to compute a single output. The algorithm attempts to model a function that defines the relationship between inputs and outputs from observed data. The output of a single neuron is given by

$$\text{out}_i = \mathbf{w}_i^t \mathbf{x} + b_i$$

The output of a single layer in matrix form is expressed as

$$\text{out} = \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,j} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,j} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i,0} & w_{i,1} & \cdots & w_{i,j} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_j \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_i \end{bmatrix}$$

Thus, for the output vector for an entire layer we have

$$\textbf{out} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Activation function

Supervised learning models aim to acquire the ability–otherwise known as generalization–to make accurate predictions. That is, the model seeks to understand outputs of observed inputs (training) in order to predict the outputs of unobserved inputs (testing). Each output goes through an activation function. There is an unlimited number of mappings between any given pair of inputs and outputs of observed data. While the optimal function need not fit all of the observed data, we must identify and apply one that generalizes well.

Activation functions are necessary to model non-linear functions; in the absence of non-linear activation functions, the network output would be a composition of linear functions, which would also be linear. We cannot approximate a non-linear function well with a linear function. As such, in the case that the function we are trying to approximate is non-linear, we would obtain inaccurate results.

A rectified linear unit (ReLU) is a popular modern activation function that is easily trained. ReLU is rooted in a biological analogy that discerns the inactive and active states of a neuron based on a particular threshold. ReLU is defined as

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

where x is a scalar.

The output from each neuron is passed into the ReLU; the ReLU operation is performed on the output; the result from the ReLU operation is the input of the next layer. This is expressed as follows

$$\text{ReLU}\big(\mathbf{W}\mathbf{x} + \mathbf{b}\big)$$

Cost function

The goal of training is to find the global minimum of the cost function. In the training process, we minimize the cost function. The smaller the cost function, the closer the output of the network is to the respective desired output. The cost function that we will use is given by

$$\mathbf{C} = (\mathbf{x}^{(K)} - \mathbf{y})^2$$

where $\mathbf{x}^{(K)}$ is the output of the last layer of the network and $\mathbf{y}$ is the desired output vector.

Backpropagation

To find the global minimum of the cost function, each weight must be tuned to its optimal value. After each training sample, the cost function is re-calculated and the weights are updated in the direction of the gradient of the cost function. This method is known as stochastic gradient descent and is expressed in terms of the partial derivatives of the cost function with respect to the weights of the network as follows

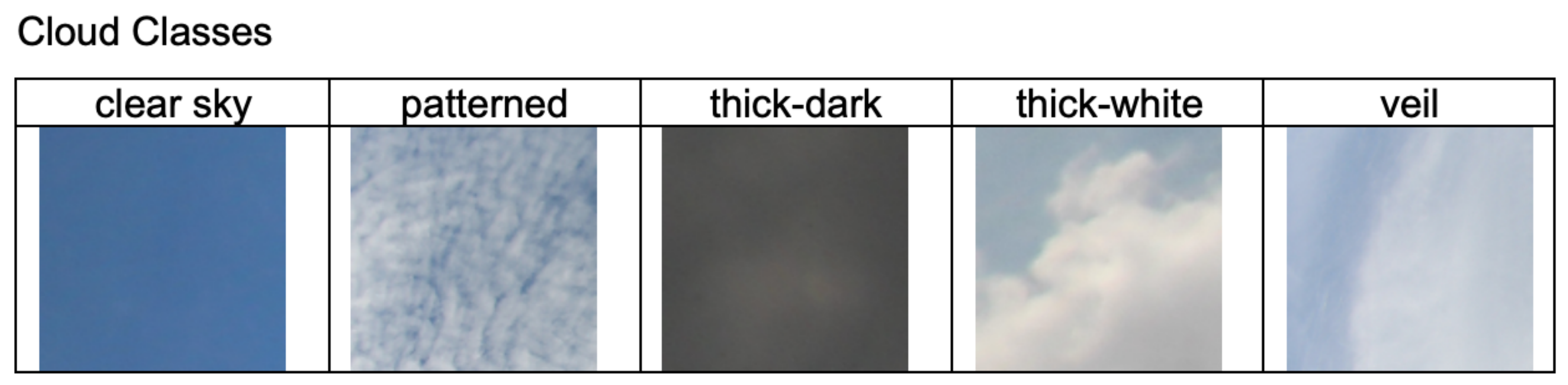
$$w_{i,j}^{(k)}(\text{new}) = w_{i,j}^{(k)}(\text{previous}) + \mu\left(\frac{\partial \mathrm{C}(w_{i,j}^{(k)})}{\partial w_{i,j}^{(k)}}\right)$$

where μ is a constant known as the learning rate which controls how much each weight is adjusted per training sample. Each derivative is calculated using the backpropagation technique in which partial derivatives of each weight are calculated using repeated applications of the chain rule.

Application

Model

A sample of cloud images (n = 784) was obtained from the Singapore Whole-sky Imaging Categories (SWIMCAT) database; the images were classified into five categories.



The RGB images were resized from 125 x 125 to 32 x 32 pixels via bilinear image interpolation. The data was split into training (80%) and testing (20%) sets by random assignment with ratios of each cloud class representative of the sample composition; in sum, there were 628 training samples and 156 test samples. We implemented a CNN in the R interface to `Keras`, a high-level neural networks API for efficient experimentation. The first layers were comprised of the convolutional base–three 2D convolutional layers separated by two 2D max pooling layers–to configure the CNN to process inputs of shape (height, width, channels). Each of these layers output a 3D tensor; the width and height dimensions decreased as the depth of the network increased. To complete the model's architecture, the last (4, 4, 64) output tensor was flattened into a 1024 length vector then input into two dense layers to perform classification. Each of the previous layers used ReLU activation. The final dense layer had five outputs corresponding to the number of image classifications and a softmax activation. In total, the model had 122,245 parameters. We compiled the model with the Adam optimizer at a learning rate of 0.001 and the categorical crossentropy loss function for integer targets. Finally, we trained the model on 150 epochs in batches of 32 images with a 10% validation split.

Performance

Accuracy and loss

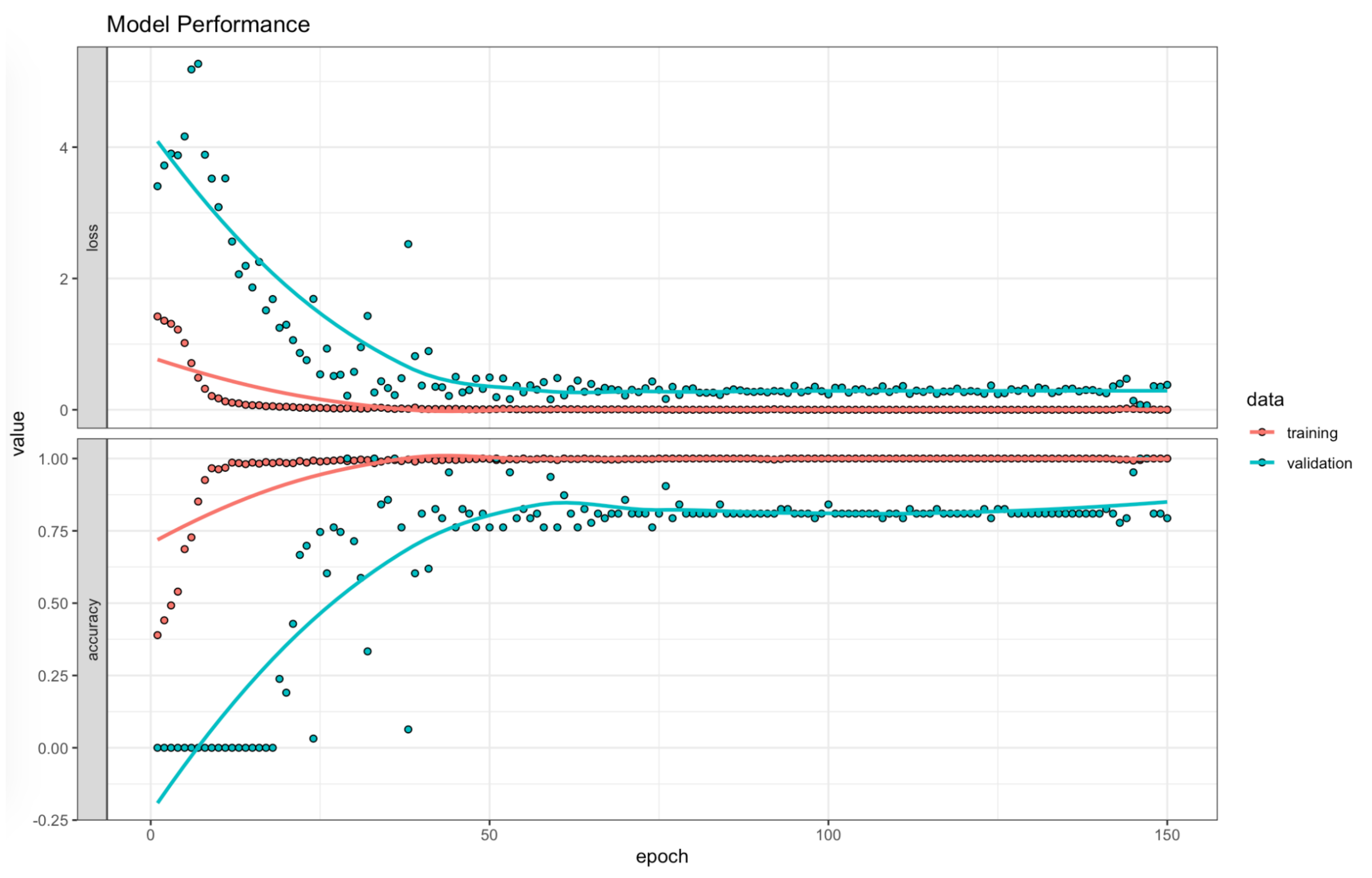
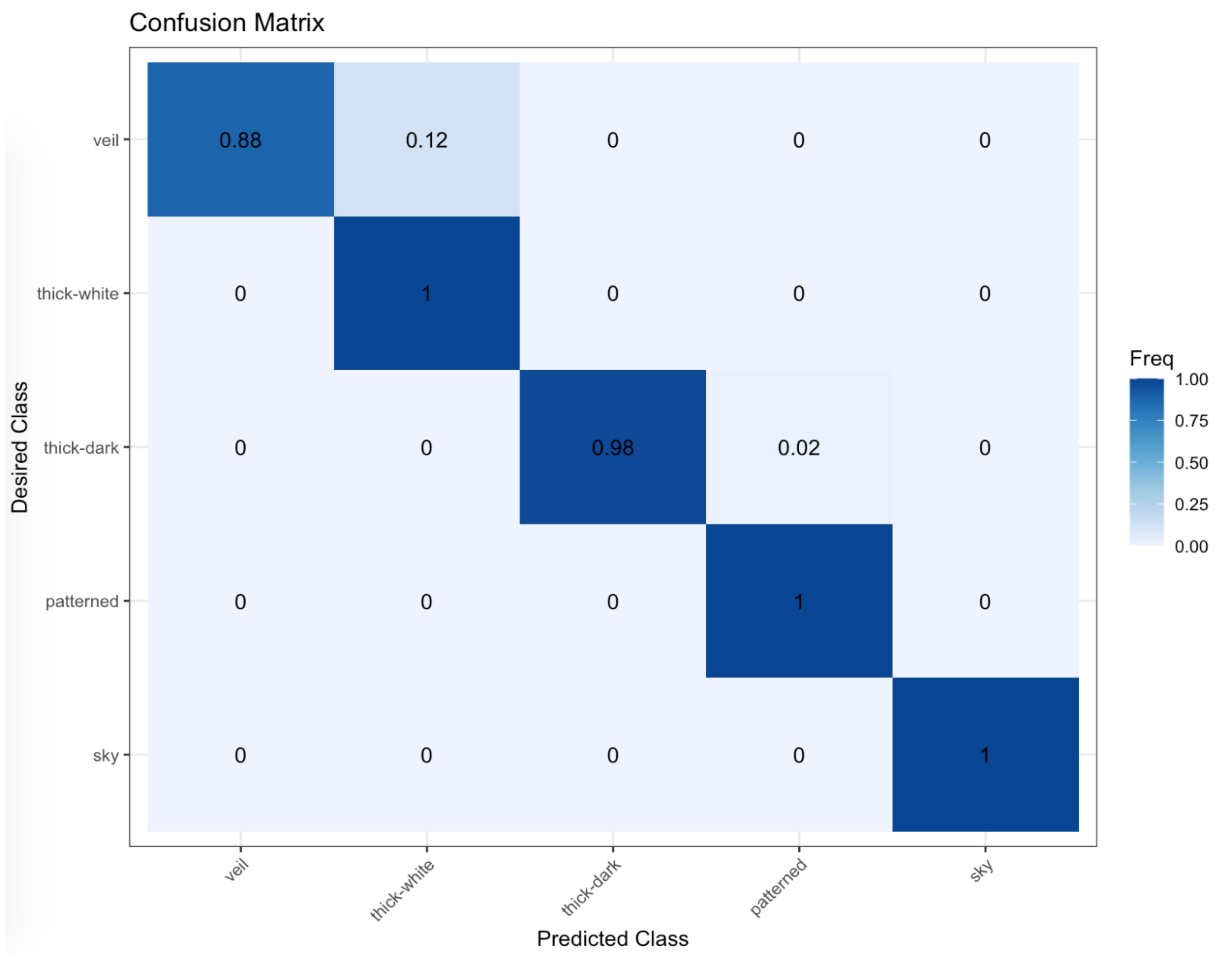


Figure 1: Our model achieved high accuracy with **98%** correctly classified cloud images. The loss metric of **0.05** calculated on the training and validation samples indicates low classification error across iterations of optimization.

Prediction



Desired Class \ Predicted Class	veil	thick-white	thick-dark	patterned	sky
veil	0.88	0.12	0	0	0
thick-white	0	1	0	0	0
thick-dark	0	0	0.98	0.02	0
patterned	0	0	0	1	0
sky	0	0	0	0	1

Figure 2: Combinations of predicted and actual values to assess classification performance by cloud type normalized to reflect sensitivity and specificity.

References available upon request.