CSYE 7370 Final Project

Changning Liu 001540758

Introduction

Abstract

Have you been watching the classic cinematic masterpiece *Alien vs. Predator*, and thought "wait, which one is which again?" I usually cannot distinguish them. By using simple Neural Network with tensorflow, movie viewing will be a breeze. No more pausing to do a taskly google search on your phone, just use a machine-learning algorithm. The algorithm classifies images as either being 'alien' or 'predator'.

This project reads in a classifiable set of images from Kaggle and transform those images into TensorFlow Datasets that can be used for DNN training. The TensorFlow Keras API is used to create a custom feedforward NN model and then a Keras Estimator is created to access the performance.

Dataset

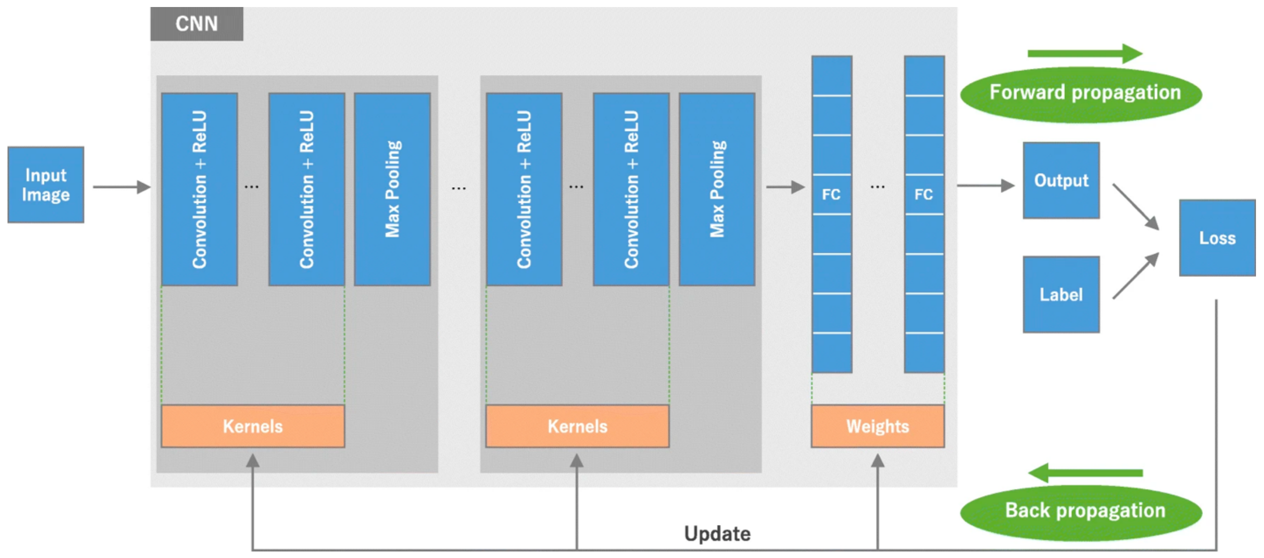
We were able to get a dataset containing Alien and Predator images from Kaggle where the dataset was listed as being an open database.

The dataset can be found here and the Terms of Use can be found here. Section 5 states that we promise to abide by "all copyright notices, trademark rules, information, and restrictions contained in any Content you access through the Services." Because we have no plans to exploit this content there are no ethical restrictions in using this data.

The data is split on Kaggle with about 700 images in the training data and 200 images in the test data. We agreed that this split is reasonable and continued to use it for our project.

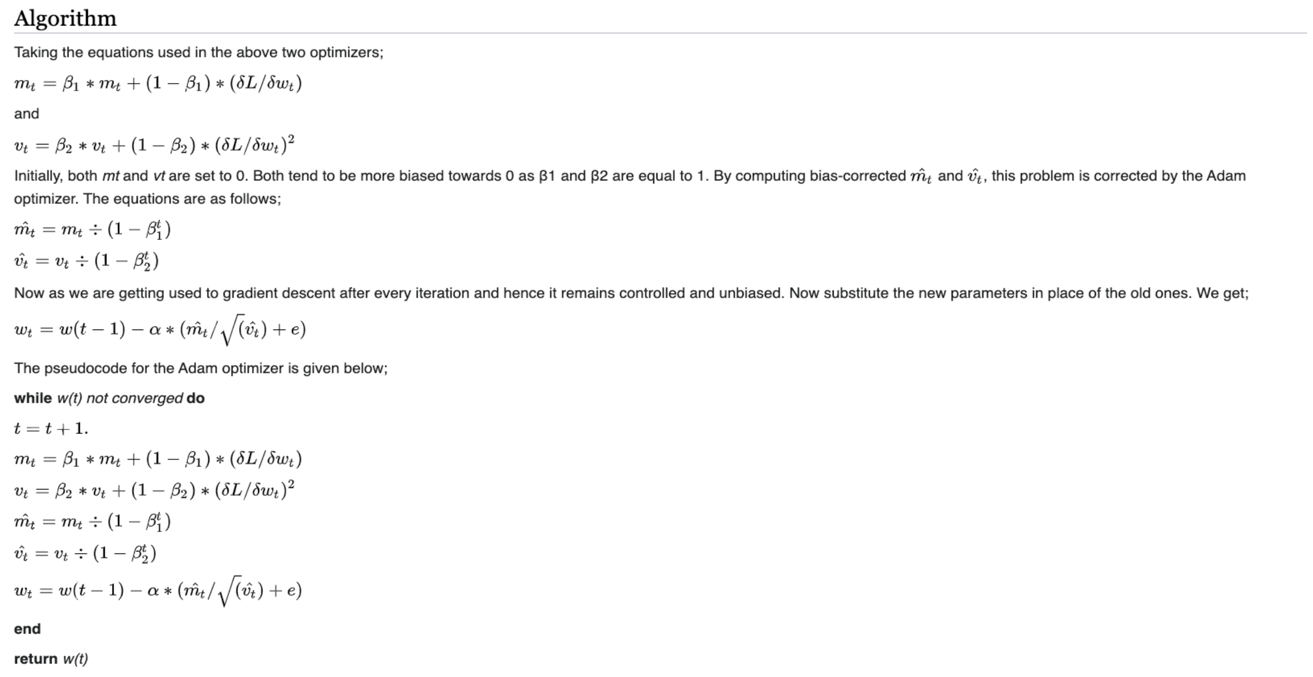
CNN

Convolutional neural network (CNN), a class of artificial neural networks that has become dominant in various computer vision tasks, is attracting interest across a variety of domains, including radiology. CNN is designed to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers. This review article offers a perspective on the basic concepts of CNN and its application to various radiological tasks, and discusses its challenges and future directions in the field of radiology. Two challenges in applying CNN to radiological tasks, small dataset and overfitting, will also be covered in this article, as well as techniques to minimize them. Being familiar with the concepts and advantages, as well as limitations, of CNN is essential to leverage its potential in diagnostic radiology, with the goal of augmenting the performance of radiologists and improving patient care.



Adam Algorithm

Adam is an algorithm for gradient-based optimization of stochastic objective functions. It combines the advantages of two SGD extensions — Root Mean Square Propagation (RMSProp) and Adaptive Gradient Algorithm (AdaGrad) — and computes individual adaptive learning rates for different parameters.



Algorithm

Taking the equations used in the above two optimizers;

$m_t = \beta_1 * m_t + (1 - \beta_1) * (\delta L / \delta w_t)$

and

$v_t = \beta_2 * v_t + (1 - \beta_2) * (\delta L / \delta w_t)^2$

Initially, both *mt* and *vt* are set to 0. Both tend to be more biased towards 0 as β1 and β2 are equal to 1. By computing bias-corrected $\hat{m_t}$ and $\hat{v_t}$, this problem is corrected by the Adam optimizer. The equations are as follows;

$\hat{m_t} = m_t \div (1 - \beta_1^t)$

$\hat{v_t} = v_t \div (1 - \beta_2^t)$

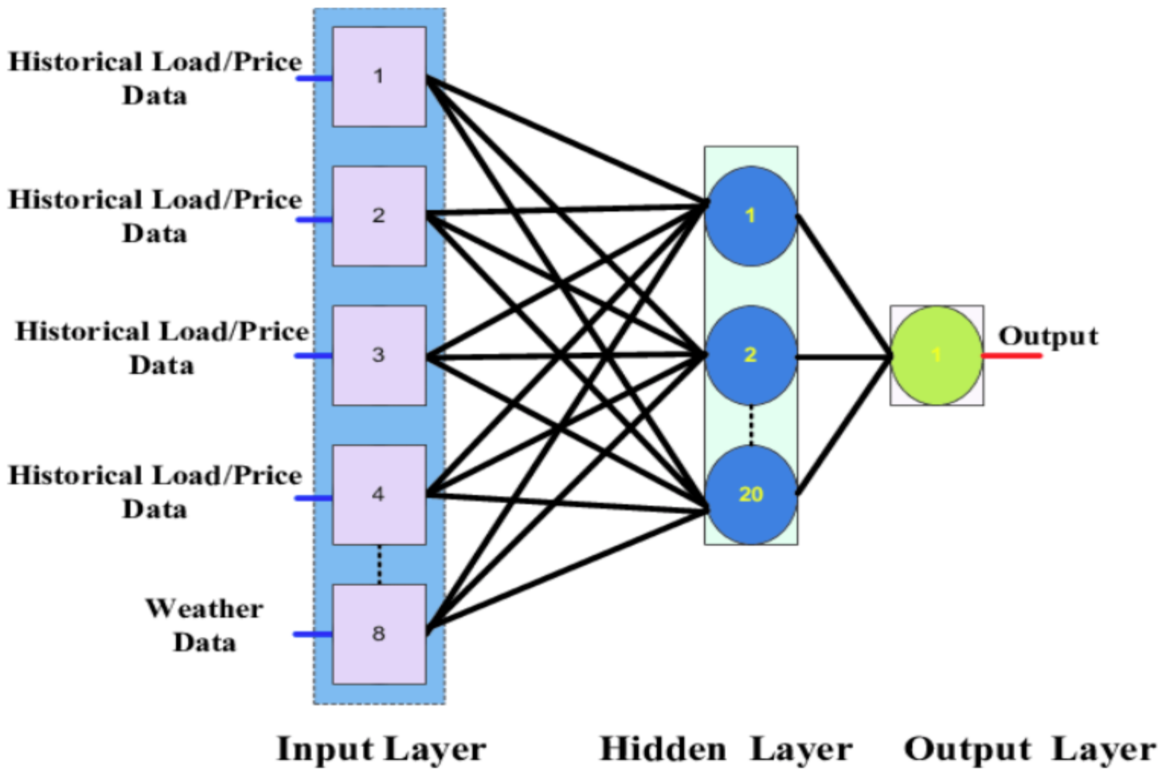
Now as we are getting used to gradient descent after every iteration and hence it remains controlled and unbiased. Now substitute the new parameters in place of the old ones. We get;

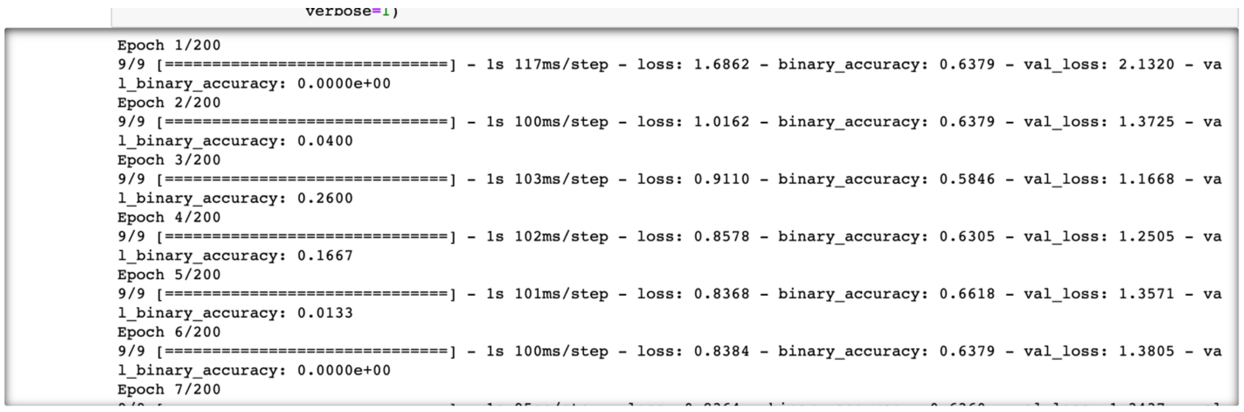
$w_t = w(t-1) - \alpha * (\hat{m_t} / \sqrt{(\hat{v_t})} + e)$

The pseudocode for the Adam optimizer is given below;

```
**while** *w(t) not converged* **do**
$t = t + 1.$
$m_t = \beta_1 * m_t + (1 - \beta_1) * (\delta L / \delta w_t)$
$v_t = \beta_2 * v_t + (1 - \beta_2) * (\delta L / \delta w_t)^2$
$\hat{m_t} = m_t \div (1 - \beta_1^t)$
$\hat{v_t} = v_t \div (1 - \beta_2^t)$
$w_t = w(t-1) - \alpha * (\hat{m_t} / \sqrt{(\hat{v_t})} + e)$
**end**
**return** *w(t)*
```

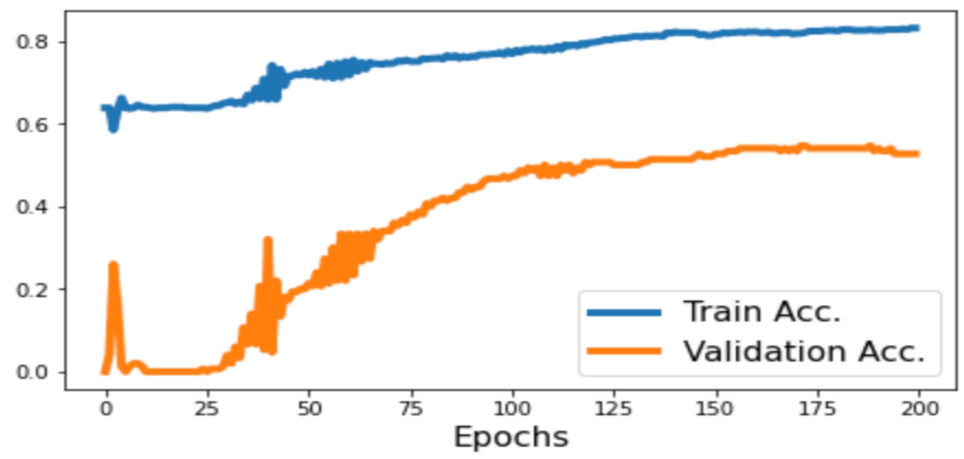
NN Model

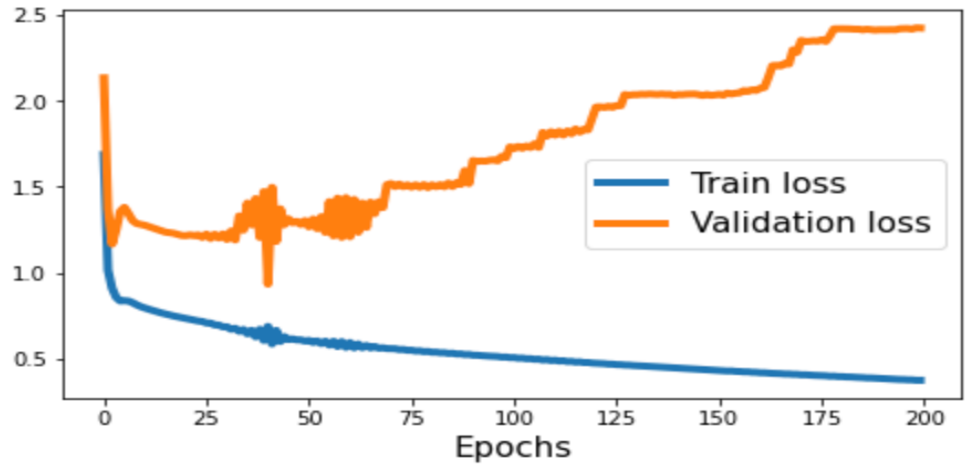

Historical Load/Price Data
Historical Load/Price Data
Historical Load/Price Data
Historical Load/Price Data
Weather Data
1
2
3
4
8
1
2
20
1
Output
Input Layer
Hidden Layer
Output Layer



```
                         verbose=1)
Epoch 1/200
9/9 [==============================] - 1s 117ms/step - loss: 1.6862 - binary_accuracy: 0.6379 - val_loss: 2.1320 - va
l_binary_accuracy: 0.0000e+00
Epoch 2/200
9/9 [==============================] - 1s 100ms/step - loss: 1.0162 - binary_accuracy: 0.6379 - val_loss: 1.3725 - va
l_binary_accuracy: 0.0400
Epoch 3/200
9/9 [==============================] - 1s 103ms/step - loss: 0.9110 - binary_accuracy: 0.5846 - val_loss: 1.1668 - va
l_binary_accuracy: 0.2600
Epoch 4/200
9/9 [==============================] - 1s 102ms/step - loss: 0.8578 - binary_accuracy: 0.6305 - val_loss: 1.2505 - va
l_binary_accuracy: 0.1667
Epoch 5/200
9/9 [==============================] - 1s 101ms/step - loss: 0.8368 - binary_accuracy: 0.6618 - val_loss: 1.3571 - va
l_binary_accuracy: 0.0133
Epoch 6/200
9/9 [==============================] - 1s 100ms/step - loss: 0.8384 - binary_accuracy: 0.6379 - val_loss: 1.3805 - va
l_binary_accuracy: 0.0000e+00
Epoch 7/200
```

Evaluation and results





```
print('\nTest Acc. {:.2f}%'.format(eval_results['binary_accuracy']*100))
```

```
Test Acc. 76.56%
```

With all the work that went into this project, I am beyond pleased with the results. As you can see in the the training accuracy from model.fit() gets up to about 85%, and with the validation accuracy we see steady improvement, showing the model is learning, caping out just over 50%.

As for loss, the training dataset shows a sharp and then steady decrease and the validation dataset shows a decrease than then increase. Meaning the algorithm was penalized for a bad prediction.

LICENSE