Capstone Project File

-Nick Hack

Domain Background: Hurricane damage evaluation is a major part of community recovery. Identification of damaged areas allows federal agencies and insurers the ability to start servicing hardest hit areas. This can be a labor intensive process, FEMA one the worlds largest disaster relief organization hands out excel based surveys for local residents and authorities to fill out damage related information with a 30 day timeline to submit for aide (Preliminary Damage Assessments | FEMA.gov). This process can add weeks onto the process of distributing funds and other disaster relief efforts. Image classification models provide a unique opportunity to quickly identify areas that have hurricane or storm damage allowing relief agencies to focus their outreach efforts to areas in greatest need.

Problem Statement: Manual Hurricane and storm damage assessments are time consuming and increase the time to disbursement of relief to communities in need. Convolution Neural Network models provide an opportunity to classify damaged areas through satellite image classification. This classification process will provide areas to focus on faster than working with local authorities to manually assess, on the order of minutes to hours vs days to weeks.

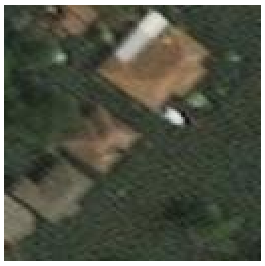
Datasets and Inputs: Kaggle dataset of satellite images of hurricane damage. Provided in Project proposal submission. The Kaggle data set divided into satellite images showing damaged and undamaged areas of Texas after Hurricane Harvey. Data originally taken from: https://ieee-dataport.org/open-access/detecting-damaged-buildings-post-hurricane-satellite-imagery-based-customized and can be cited with http://dx.doi.org/10.21227/sdad-1e56 and the original paper is here: https://arxiv.org/abs/1807.01688 (Sourced: Satellite Images of Hurricane Damage | Kaggle)

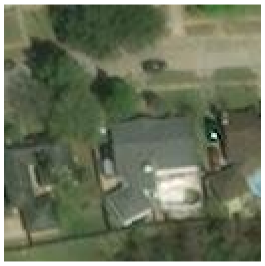
Benchmark model: There have been other models created on this data set. I have chosen to use the pertained ResNet CNN model. This is a model that has been used throughout the Udacity course and is a good baseline model. The Benchmark accuracy for the ResNet model is 70% for this data set based on the work completed by RATNA SAMBHAV who submitted his results to Kaggle for public viewing on the dataset (Univ.Ai AI2 Project | Kaggle).

Evaluation Metrics: The Kaggle dataset is labeled images. Evaluation of the model will be based on accuracy of the model to classify the test dataset against the labeling of damaged vs undamaged. Based on other models that have been created on this data set an accuracy of 70% is the benchmark I will be attempting to achieve for the pretrained ResNet CNN model.

Project Design: Using the methods learned through out the Udacity course, I will first import the image data into Amazon S3 and then complete some simple heuristic analysis of the data. This will help to identify any bias in the image data or associated information attached to each image. After a pretrained Resnet CNN model will be used to train, test, and validate classification of images by damaged and non-damaged. Once the model has been trained and the best hyperparameters chosen based on highest accuracy of classification, an endpoint and will be setup so images can be submitted for classification.

Data Exploration: The dataset is a series of satellite images of houses some with hurricane damage as shown below in image 1A. and houses without damage as shown below in image 1B.

1A. 

1B. 

There is some variability within the images within the model the input shape is defined as defined here: nn.Linear(16 * 53 * 53, 120).

Implementation: Earlier in this report I detailed the Dataset and inputs so I will briefly submit and overview here as well. The dataset I used came from Kaggle, specifically I used the train_another file with 5000 images of damaged and non-damaged hurricane images, and the test_another file with 9000 hurricane images which is unbalanced at the ratio of 8000/1000 damaged/undamaged images.

Based on previous projects with image classification, mainly the dog image project. I chose the same metrics for determining validity and performance of the model.

Performance Metric:

```
        objective_metric_name = 'average test loss'
        objective_type = 'Minimize'
        metric_definitions = [{'Name': 'average test loss', 'Regex': 'Test set\\: Average loss\\: ([0-
9\\.]+)'}]
```

I also defined a range of hyperparameters in which to utilize in the training of the model to optimize the performance of the model. Shown below are the parameters as utilized in many of the previous projects (learning rate, batch size, and epochs) along with the corresponding ranges that will be utilized in the training of the model.

Hyperparameters:

```
hyperparameter_ranges = {
 "lr": ContinuousParameter(0.001, 0.1),
"batch_size": CategoricalParameter([16, 64, 256, 1024]),
 "epochs": IntegerParameter(2, 4)
}
```

Refinement: I initially started with utilizing a pretrained resnet model but was having issues with the estimator and entry point python file. With this set back I went back to the Kaggle site and reviewed the work that I originally referenced as a benchmark model, by RATNA SAMBHAV who submitted his results to Kaggle for public viewing on the dataset (Univ.Ai AI2 Project | Kaggle). I noticed that they utilized a convolutional neural network (CNN). This model was constructed using the keras python package. Throughout this course we have been learning machine learning techniques using the pytorch python package, for that reason I chose to construct a CNN rather than use the pretrained model that I initially discussed using. The below model architecture defines the model parameters and the in the forward function the model computation is defined. The architecture of the model is as follow two convolutional layers and three fully connected layers as defined in the pytorch tutorial pages (Training a Classifier — PyTorch Tutorials 2.0.0+cu117 documentation). I chose to reduce the number of layers for the sake of training speed, and planned on adding more layers if the accuracy of the model was reduced past that of the benchmark set by RATNA SAMBHAV within his model shown here (Univ.Ai AI2 Project | Kaggle).

Model:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
         self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
         self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 53 * 53, 120)  # fix the input shape
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 2)

    def forward(self, x):
        x = self.pool(torch.relu(self.conv1(x)))
        x = self.pool(torch.relu(self.conv2(x)))
        x = x.view(-1, 16 * 53 * 53)  # fix the input shape
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
         x = self.fc3(x)
        return x
```

Model Evaluation and Validation: The final model achieved an accuracy of 82% and the best hyperparameters that achieved the above accuracy are listed below.

```
Best Hyperparameter Values:
{'_tuning_objective_metric': '"average test loss"',
'batch_size': '"64"',
'epochs': '4',
'lr': '0.008925392204913216',
'sagemaker_container_log_level': '20',
'sagemaker_estimator_class_name': '"PyTorch"',
'sagemaker_estimator_module': '"sagemaker.pytorch.estimator"',
'sagemaker_job_name': '"pytorch-training-2023-04-26-22-36-05-052"',
'sagemaker_program': '"hpo_try.py"',
'sagemaker_region': '"us-east-1"',
'sagemaker_submit_directory': '"s3://sagemaker-us-east-1-095368982544/pytorch-training-2023-04-26-22-36-05-052/source/sourcedir.tar.gz"'}
```

The model accuracy improved upon the results set in the benchmark model of 70% by RATNA SAMBHAV his model shown here (Univ.Ai AI2 Project | Kaggle). To determine the statistical significance I performed a statistical test of diffence in the accuracy between the two models. The null hypothesis being the proportion of true predictions is the same between the two models. With the given performance results below. This approach was identified and replicated from the blog post by Jason Brownlee on the Machine Learning Mastery website on August 21st 2020 https://machinelearningmastery.com/hypothesis-test-for-comparing-machine-learning-algorithms

Results: Benchmark model: Accuracy = 70% Trained on N=10000 images, True Positive N = 7000, and True Negative N = 3000

My Model: Accuracy = 82% Trained on N=10000 images, True Positive N = 8200, and True Negative N = 1800

The statistical test equation:

Z=(p1-p2)/sqrt(p_pool*(1-p_pool)*(1/n1+1/n2))

Where p_pool is:

(TP1+TP2)/(N1+N2)

The values of the above variables:

P1 = .7, P2 = .82, n1 = n2 = 10000

TP1 = 7000, TP2 = 8200, N1 = N2 = 10000

Calculation:

Z = (.82-.7)/sqrt(.75*(1-.75)*(1/10000+1/10000)) = 18.73

P_value = .0039

Given the P_value of <.05 we reject the null hypothesis and can say that the accuracy results are statistically different between my model results and the benchmark results. With my model outperforming the benchmark. This provides enough evidence that the problem of image classification of damaged and undamaged hurricane images is solved.