Vrinda Pandey

COE379L

Project 3

- Data preparation: what did you do? (1 pt)

A couple steps went into the data preparation for the project. First directories were made for each of the classification groups (damage and no damage) to divide into train and test splits.

Then the data was randomly sampled to place into the train and test split directories at a) 0.8:0.2 ratio

Next the data (image) attributes were identified by plotting a sample image to identify the key features for model training.

Finally the data was normalized as necessary (RGB) 1/255 rescaling

- Model design: which architectures did you explore and what decisions did you make for each? (2 pts)

I explored the CNN, the lenet-5, and the alternate lenet-5 from the research paper.

For the CNN: Class Example

- Input layer: Chose 64 filters for a good set for pattern detection.
- Second Convolution: 32 filters to save computation and prevent overfitting
- Third Convolution: 32 filters to maintain complexity
- Dense Layer: 84 neurons for compressed representation
- Output Layer: softmax used to ensure multi-class output sums to 1

For the lenet-5 : Original Class Example

- Layer 1: 6 filters to extract basic features. Average pooling to downsampled. Original

- Layer 2: 16 filters for more layers of complexity. Original Lenet 5
- Dense Layer: 120 and 84 neurons for feature discriminations
- Output Layer: softmax as the problem is a multi-class classification

For the alternate lenet-5:

This final alternate lenet-5 was refined from the original for higher performance on more complex input data and binary classification tasks such as ours as the multi classification problem only has 2 classes. In comparison to the original lenet 5 this one has:

- Layer 1: 32 filters instead of 6 filters to extract more details on colored images
- Layer 2: 64 filters instead of 16 filters for more complexity and richer feature maps
- Dense Layer: 256 instead of 120 neurons for more expressiveness in the model and 128 instead of 84 for more abstract feature discrimination
- Output Layer: sigmoid instead of softmax as the problem is being set to binary classification

- Model evaluation: what model performed the best? How confident are you in your model? (1 pt)

For me the alternate lenet 5 performed the best. The paper's goal is tailored exactly to our problem: image classification of Harvey damage so I believe the researchers refined the parameters refined specifically to this problem. I'm the most confident about this model and can say it has my 96% confidence based on the models performance metric. I think this model can be used as a great baseline metric for sure and depending on the intended use can even be a reliable predictor. Potentially if there is a lot of money on the line (insurance, etc.) we could refine the model a bit for more certainty.

- Model deployment and inference: a brief description of how to deploy/serve your model and how to use the model for inference (this material should also be in the README with examples) (1 pt)

As briefly scripted in my read me deploying the model using the inference server is fairly simple. In short one can follow the following steps:

- Download all files from vrindapandey/harveyDamage
- Run the docker compose up command to get the inference server running through the docker-compose.yml file present in the repository
- Test images using a script to automate or individually using the curl command:

  curl -X POST http://172.17.0.3:5000/inference -F "image=@data/damage/-93.66109_30.212114.jpeg"

  (make sure the image points to your image)

- Docker compose down when all testing is complete on the server