Interpretation3
Teo Speece

In your write-up, include the following:

- (5 points) **Problem:**

  To train a convolutional neural network for some image recognition task. 3 models are required one with 2 convolutional layer and 2 pooling layers. The second model needs require changes to parameters and the 3 model uses a new layer. Specific to my dataset being able to predict the object on the ground is helpful to identify 50 year floodplains. Vegetation will take in some of the water, while roads or any other impermeable surface will not therefore make the floodplain larger. The floodplain is important because it determines which houses are required to purchase flood insurance. In

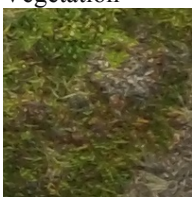
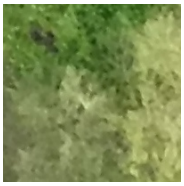
- (5 points) **Data:**

  The dataset used contain aerial imagery of vegetation and roadways. The dataset already came with a test and train folder, unfortunately the test folder had less than 20% of the total images that may result in overfitting. The data was obtained from kaggle.com.

  Roadway

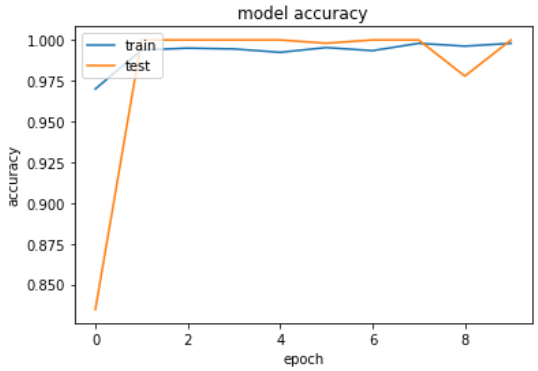
  
  
  

  Vegetation

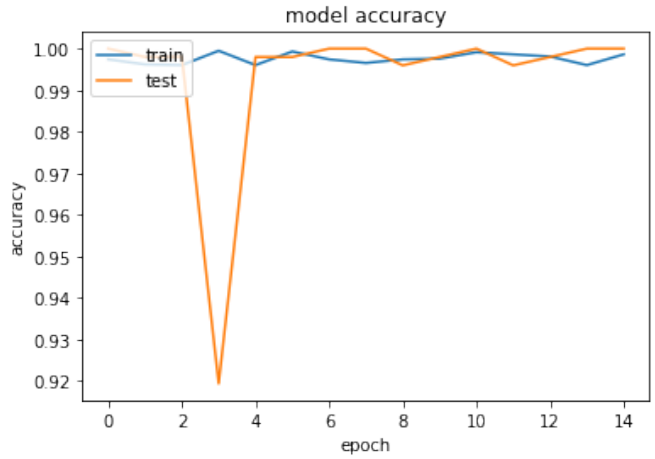
- (10 points) **First Model**:

  To create the first model, the cat dog model 3 was used. The model has 3 convolutional layers and 3 pooling layers . The model showed high accuracy so few epochs (10) were used to avoid overfitting. The model did not need to include to many layers or many parameters tuned because the images are very distinguishable. The accuracy was very high, but in a realistic setting the images will be split up categorically in very similar classes like semi permeable roadways and roadways which would affect the accuracy.

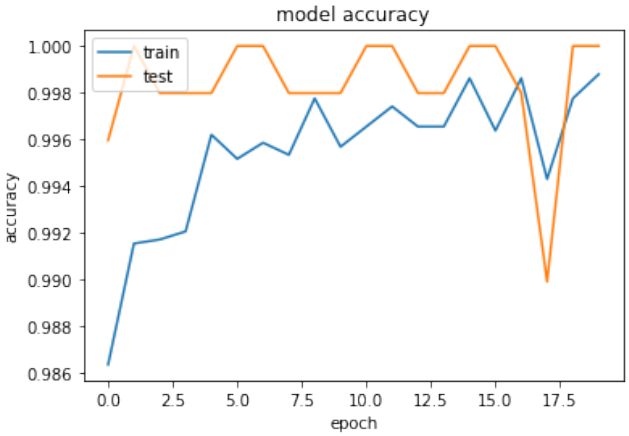


- (15 points) **Second Model:**

  In the second model, the number of filters was increased to 64 for all Conv2d layers, and the pool sizes were increased to (4,4). The filter is the dimensionality of the output space. Poolsize is the size of the pooling matrix. Increasing the filter size increase the effective receptive field of the filter and increases the amount of information that it can capture. The downside is longer computation time. The increased pool size (4,4) will decrease the size of the output matrix which leads to a more generalizable model and lower overfitting. This also made the computation time quicker. The model did mostly the same as model 1, but had an odd dip in accuracy at 4 epochs. Again not many epochs were needed to avoid overfitting.



- (15 points) **Exploration Part:** (two options) A was chosen
  The new layer I chose to experiment with was Gaussian Noise which is used to mitigate overfitting. Overfitting is a problem when neural networks pick up on common patterns specific to the images we have even though they may not be useful elsewhere. Adding noise to an image reduces some of these patterns and can help avoid overfitting. They are used in CNNS to make models more generalizable. Data source: https://keras.io/layers/noise/ and https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced Although, the model appears to be worse than the first two models it does better (when just looking at accuracy). The range is much lower and has generally higher accuracy (values that are not 100% are much closer to 100%). With GaussianNoise we would expect overfitting to be less as well.


model accuracy
train
test
accuracy
epoch
1.000
0.998
0.996
0.994
0.992
0.990
0.988
0.986
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5