Bridge Inspection Through UAV Drones

Alexis Diaz, Yanlin Guo, Hussam Mahmoud, Brandon Perry, Carina Morroni

Department of Civil & Environmental Engineering

Colorado State University, Fort Collins, CO, USA

WALTER SCOTT, JR.
COLLEGE OF ENGINEERING
COLORADO STATE UNIVERSITY

Introduction

Around 600,000+ Bridges need a minimum Bi-Yearly Inspection

Leading to at least 1.2 million Inspections Occurring Pre-Year

The Propose Of this Research is to compensate the short coming of Humans by having Drones inspect and examine the Bridge

The Research was Preformed on Two Colorado Bridges

The Flight Time was Two Hours at Each Site

The Drone Inspection makes the Process Safer, Easier & more Cost Efficient w/ Automating & Quantifying Data

Neural Network Setup

A Neural Network was Set Up to Process Images and Identify Cracks in the Steel of the Bridges

The Neural Network used was a CNN (Convolutional Neural Network)

The CNN Process the Images by Running them through Filters

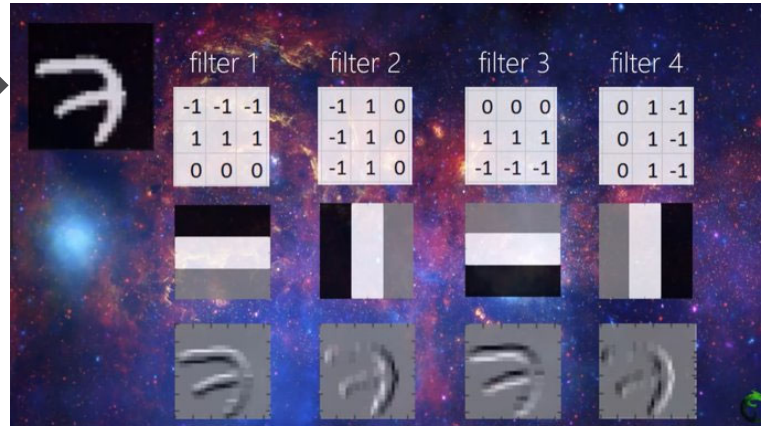

Figure 1. Shows an example of a Neural Network Running
Network trained to look for edges – the white parts of each photo show the top, bottom, left, and right edges of the original photo

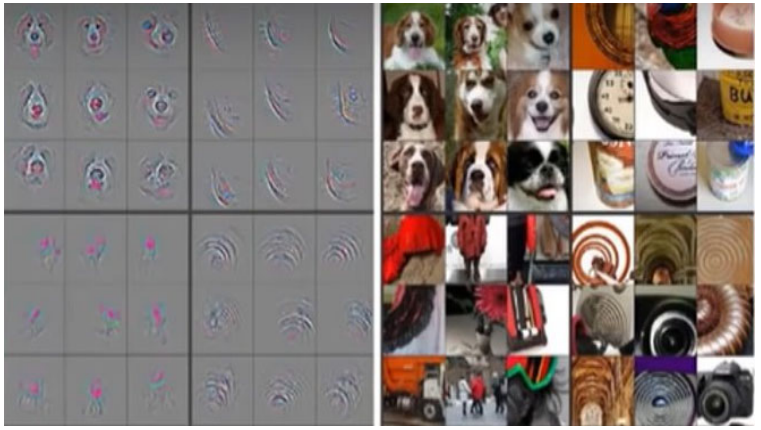

Figure 2. Shows images of a Neural Network after processing different images
With more Filters in the Network, a Neural Networks can detect more Sophisticated Patterns, such as Dog Faces

Methods of Research

My Research Consisted of Running the Neural Network Under different Conditions to find the Most Efficient Network

To find the most Efficient Network we changed the images being put into the System while also being changing the Network itself

Adding & Deleting Conv/Decnov Layers Changes the Time & Size of the Network

```
# Initializer
initializer = 'he_normal'

# Input
inputs = tf.keras.Input(shape=(128, 128, 1), name="Grayscale_Images", dtype=tf.float32)

# Conv1
x = layers.Conv2D(64, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(inputs)
x = layers.Conv2D(64, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.BatchNormalization()(x)
x = layers.AveragePooling2D(pool_size=(2, 2), strides=(2,2), padding='same')(x)

# Conv 2
x = layers.Conv2D(128, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.Conv2D(128, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.BatchNormalization()(x)
pool_128 = layers.AveragePooling2D(pool_size=(2, 2), strides=(2,2), padding='same')(x)

# Conv 3
x = layers.Conv2D(256, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(pool_128)
x = layers.Conv2D(256, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.Conv2D(256, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.Conv2D(256, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.BatchNormalization()(x)
pool_256 = layers.AveragePooling2D(pool_size=(2, 2), strides=(2,2), padding='same')(x)
x = layers.Dropout ( 0.4 ) (pool_256)

# Conv 4
x = layers.Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.Conv2D(512, (3, 3), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = layers.BatchNormalization()(x)
pool_512 = layers.AveragePooling2D(pool_size=(2, 2), strides=(2,2), padding='same')(x)
x = layers.Dropout ( 0.4 ) (pool_512)

# DeConv 1
x = layers.Conv2DTranspose(512, (4, 4), strides=(1, 1), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = x + pool_512

# DeConv 2
x = layers.Conv2DTranspose(256, (4, 4), strides=(2, 2), padding='same', activation="relu", kernel_initializer=initializer)(x)
x = x + pool_256

# DeConv 3
x = layers.Conv2DTranspose(2, (16, 16), strides=(8, 8), padding='same', activation="relu", kernel_initializer=initializer)(x)

# SoftMax
prediction = layers.Softmax()(x)

model = tf.keras.Model(inputs=inputs, outputs=prediction, name="Model")
model.compile(optimizer=Adam(learning_rate=initial_lr),
              loss=tf.keras.losses.CategoricalCrossentropy(),
              metrics=['accuracy',f1_m,precision_m, recall_m])
                      #tfa.metrics.F1Score(num_classes=2, average="micro", threshold=0.9, dtype=tf.float32),
                      #tf.keras.metrics.MeanIoU(2, name='MIoU', dtype=tf.float32)])

return model
```

Figure 3. Represents the Code Used for Creating the Model of the Network; Editing this Code Changes size and Efficiency

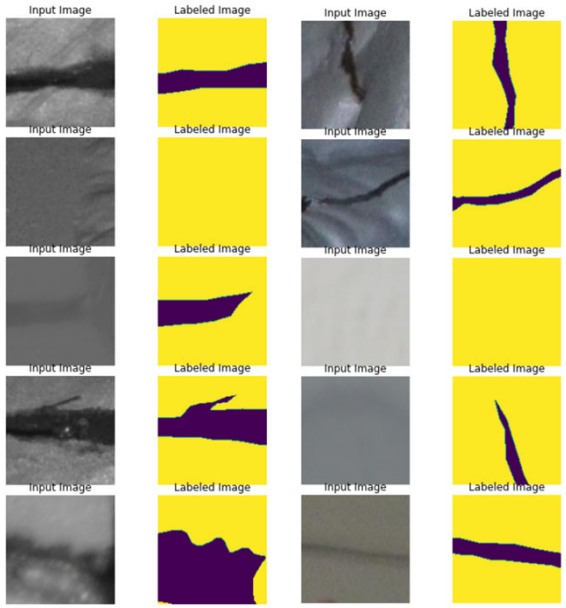

Figure 4. Shows Input Images of the Images that are Run though the Network to Identify the Cracks in the Steel;
Different Images were put into the Network to test the efficiency of different Images

Results

The Networks are run with different Conditions to find the Most Efficient Network

In these Results we were finding the Ideal Network while also running the Network with Black & White Images

We looked for the Network to have Minimal Loss and High Accuracy

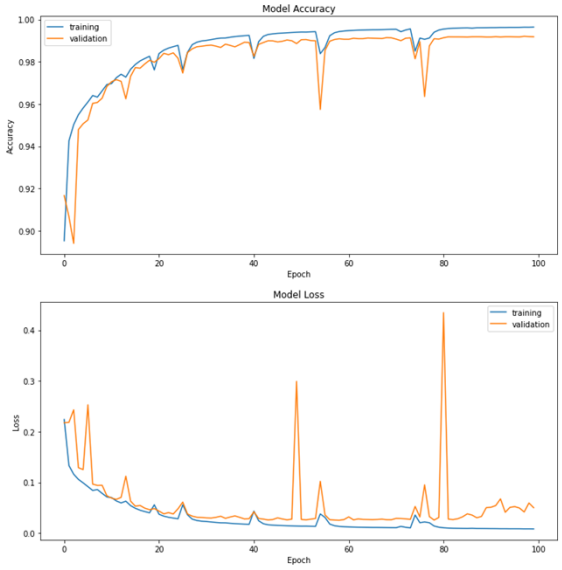

Figure 5. The Graph shows the Results from running the Neural Network with Black and White Images with Conv 5 being Deleted from the Network's Code

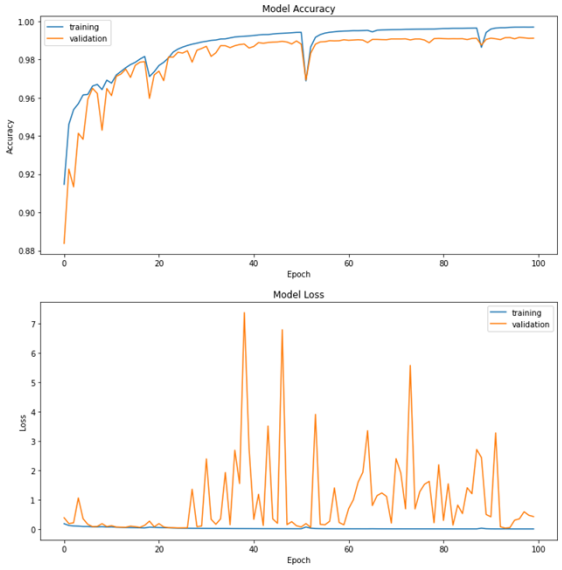

Figure 6. The Graph shows the Results from running the Neural Network with Black and White Images with Conv 7 being Deleted from the Network's Code

Next Steps

The Next Steps in the Research would be to Optimize the Network In Order to find the the Most Efficient Network that could be used to Identify a Crack in Steel

Additional Training to the Network will be needed to be done, the Training would be needed to get the Network trained to be able to process all different kinds of Images from the Steel of the Bridges

What benefits did you get from your SURE Experience?

The SURE Experience has given me access to an Experience that I normally would not have until becoming a Graduate Student. Participating in the Graduate Research shows me the work required in the graduate courses. The Research also connects me with Graduate Students and Professors from my Area of Study

References & Acknowledgements

All Graphs and Images of the Bridges/Network are taken directly from the Research

The Research was Conducted with support from Colorado State University and the Mountain-Plains Consortium, a University Transportation Center funded by the U.S. Department of Transportation. (FASTACT Grant No. 69A3551747108)

Thank you to the Suzanne and Walter Scott Foundation, Tointon Family Foundation, The Filsinger Family, and Contributors to the Dean's Innovation fund for making the SURE program possible

Thank you

Colorado State University