CSC 180 – Project 3

Computer Vision using GPU and Transfer Learning

By: Quinn Roemer & Logan Hollmer

Model/Code Design

Model differences

- Quinn trained custom CNN's with batch normalization
- Logan train his CNN's without batch normalization but with a focus on having multiple convolution layers
- Quinn also did more exploration with transfer learning and tried out several different models including VGG16 (with varied up sampling), ResNet50,and DenseNet201

```
cnnModel = Sequential()
cnnModel.add(Conv2D(32, kernel_size=(3, 3), activatio
cnnModel.add(BatchNormalization())
cnnModel.add(Conv2D(32, kernel_size=(3, 3), activatio
cnnModel.add(BatchNormalization())
cnnModel.add(MaxPooling2D(pool_size=(2, 2)))
cnnModel.add(Dropout(0.25))
cnnModel.add(Conv2D(64, kernel_size=(3, 3), activatio
cnnModel.add(BatchNormalization())
cnnModel.add(Dropout(0.25))
cnnModel.add(Conv2D(128, kernel_size=(3, 3), activati
cnnModel.add(BatchNormalization())
cnnModel.add(MaxPooling2D(pool_size=(2, 2)))
```

Quinn's model with batch normalization

```
model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1),
                 activation=act,
                 input_shape=(32, 32, 3), padding='same'))
model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1),
                 activation=act, padding='same'))
model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1),
                 activation=act, padding='same'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2, 2)))
```

Logan's model with multiple convolution layers

Model/Code Design

```
#Create model with upsampling
newModel = Sequential()
newModel.add(UpSampling2D(input_shape=(32, 32, 3), size=(7, 7)))
```

Quinn's Code for up sampling

```
#Add an upsampling layer and some others...
newOut = vgg_model.output
newOut = GlobalAveragePooling2D()(newOut)
newOut = Dense(1024, activation='relu')(newOut)
```

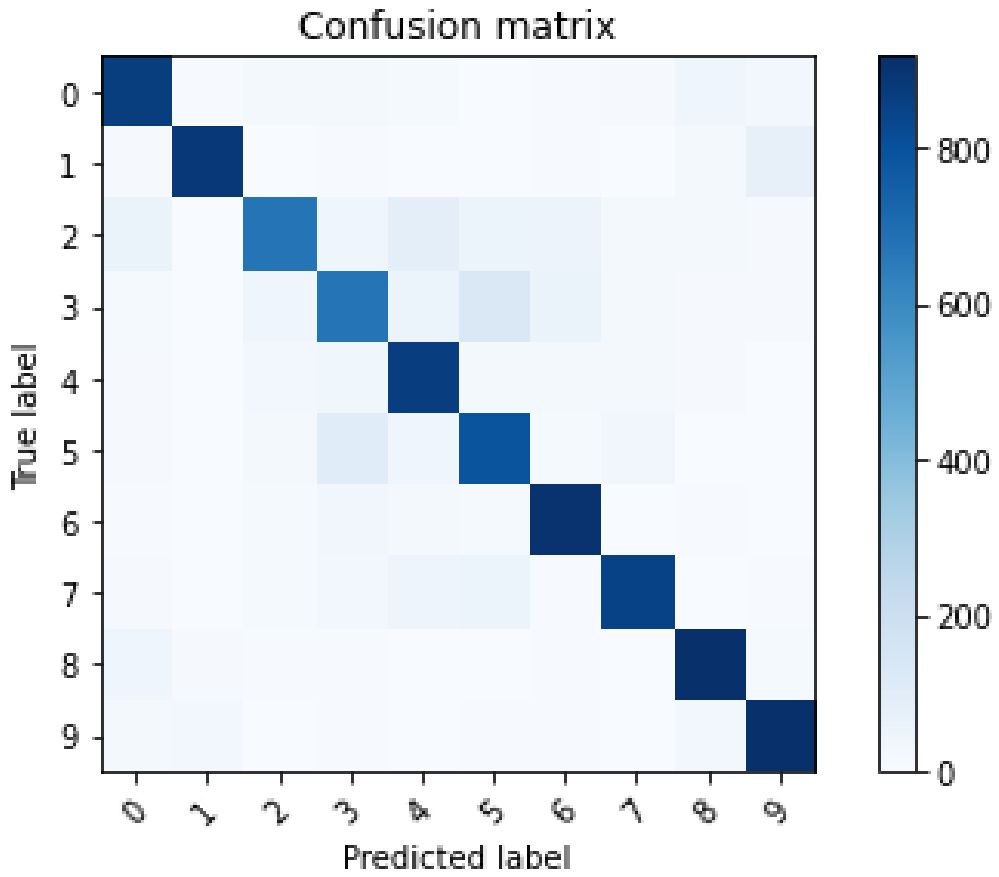
Quinn's Flattening

Data/Code Differences

- Different approaches to up sampling
- Different approaches for flattening

Findings/Results

F1 score: 0.8349



Model Architecture

conv2d_4 (Conv2D)	(None, 30, 30, 32)	896
batch_normalization (BatchNo	(None, 30, 30, 32)	128
conv2d_5 (Conv2D)	(None, 28, 28, 32)	9248
batch_normalization_1 (Batch	(None, 28, 28, 32)	128
max_pooling2d_3 (MaxPooling2	(None, 14, 14, 32)	0
dropout_5 (Dropout)	(None, 14, 14, 32)	0
conv2d_6 (Conv2D)	(None, 12, 12, 64)	18496
batch_normalization_2 (Batch	(None, 12, 12, 64)	256
dropout_6 (Dropout)	(None, 12, 12, 64)	0
conv2d_7 (Conv2D)	(None, 10, 10, 128)	73856
batch_normalization_3 (Batch	(None, 10, 10, 128)	512
max_pooling2d_4 (MaxPooling2	(None, 5, 5, 128)	0
dropout_7 (Dropout)	(None, 5, 5, 128)	0
flatten_3 (Flatten)	(None, 3200)	0
dense_8 (Dense)	(None, 512)	1638912
batch_normalization_4 (Batch	(None, 512)	2048
dropout_8 (Dropout)	(None, 512)	0
dense_9 (Dense)	(None, 128)	65664
batch_normalization_5 (Batch	(None, 128)	512
dropout_9 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 10)	1290

Findings/Results

Type	Average F1 Score	Activation Function	Optimizer
Custom CNN	0.8349	Relu	Adam
VGG16 with Upsampling layer	0.7389	Relu	Adam
Custom CNN	0.7347	Sigmoid	Adam
DenseNet201	0.7185	Relu	Adam
Custom CNN	0.7082	Tanh	SGD
VGG16	0.7002	Relu	Adam
ResNet50	0.4075	Relu	Adam

Quinn’s top models of various types

Task Division, Challenges Encountered, and what we Learned

Task Division

- Both group members created fully functional programs to pre-process data, train, and test our models.
- Quinn compiled and edited the notebook along with writing a section of the report.
- Logan created the presentation and wrote the first two and last section of the report.

Challenges Encountered

- We struggled with RAM however, we both had access to the high RAM environments.
- We also had much more trouble getting a model with a high F1 score. Many of our models seemed to preform quite badly. We found that using custom models with many layers gave us the best results

What we Learned

- We learned that there is several different ways to get data into the correct shape. We could use the reshape functionality in NumPy but we found we could also use the UpSampling2D functionality in Keras
- We also learned that you don’t always need to use the Flatten layer. A GlobalAveragePooling2D layer from Keras can replace a Flatten layer.
- Batch normalization can greatly help improve the results