

Architectural Heritage Elements Dataset Image Classification Modeling with CNN

By: Ray Thomas

Introduction:

This project will use the “Architectural Heritage Elements” dataset, which can be found at the following link:

<https://datahub.io/dataset/architectural-heritage-elements-image-dataset>

The zip files for the training dataset is downloadable using the following link:



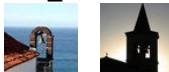


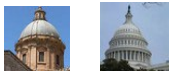


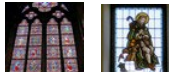

https://correo.cartif.es/home/joslla@cartif.es/Briefcase/Architectural_Heritage_Elements_image_Dataset/Architectural_Heritage_Elements_Dataset_64%28creative_commons%29_revised.zip

Likewise the zip file for the test dataset is downloadable at:

https://correo.cartif.es/home/joslla@cartif.es/Briefcase/Architectural_Heritage_Elements_image_Dataset/Dataset_test_AHE_64.zip

The dataset is comprised of images taken of Architectural elements. Using the dataset, the primary goal of this project is to build and evaluate a Convolutional Neural Network (CNN) for image classification.

The training dataset contains 10,130 observations and 10 classes. The classes, their corresponding sizes, and 2 examples of each are given below:

1. altar – 828 	2. apse – 505 	3. bell_tower – 1,057 	4. column – 1,914 	5. dome(inner) – 589 
6. dome(outer) – 1,175 	7. flying_buttress – 405 	8. gargoyle – 1,562 	9. stained_glass – 998 	10. vault – 1,097 

The test dataset contains 1,404 observations. Of these 11 classes, a subset of the 6 largest classes is chosen to build the CNN classifier. Each observation is a 64 pixel x 64 pixel RGB color image.

In order to minimize learning bias towards classes that have a larger number of observations, we utilize SMOTE to equalize the classes. This results in all 6 classes containing 1,914 observations for a grand total of 11,484 observations in the training dataset.

Model Architecture:

The model is built using TensorFlow and Keras. The CNN portion comprises of two convolutional layers, with their corresponding max-pool layers in between. A RELU activation function is utilized in between each layer. This is followed by a flattening layer. This is then fed into a Multi-Layer Perceptron for automatic classification. This MLP contains a dense layer which uses dropout in instances where the size of the hidden dense layer is large. This is followed by a final dense layer which uses SoftMax activation to give the final probabilities.

The project explores differences models that use combinations of 32 vs 64 channel sizes for the convolutional layer, with a window size of 3x3 and a max-pool size of 2x2. For the flattened layer, the size will be determined by multiplying the 3 components of the second max-pool layer. For example, when the second max-pool layer results in an output image of size 14 pixel x 14 pixel with 32 channels, then the flattened layer will be a long vector of size $14 \times 14 \times 32 = 6,272$. And for the case of 64 channels in the second max-pool layer, the flattened layer size is $14 \times 14 \times 64 = 12,554$.

When considering the options for potential values for the size of the hidden layer (h), we use the parsimony principle, which tells us that the ratio of the number of informations and the total number of parameters should be greater than or equal to 1. Following this principle, we find that since we have 11,484 training observations and 6 classes, the number of informations provided is $11,484 \times 6 = 68,904$. Therefore to satisfy the principle, the values of h that are used for the different permutations of the 32 and 64 channel mixtures is $h = [9, 3, 2]$ with their corresponding number of parameters being $\text{numPARAMS} = [66661, 57051, 63828]$. Thus, the parsimony ratios for those 3 models are $[1.03, 1.21, 1.08]$. Though these ratios all correspond to theoretically “robust” models, because the latter two models utilize an h-value that is smaller than the number of classes, this results in models with very poor training and test accuracies. Therefore, we attempt to compensate by stipulating that the value of h must be greater than or equal to the number of classes.

We compile all 8 of these models with 100 epochs and $\text{batch_size} = \text{sqrt}(\text{len}(\text{training_data}))/2$. Then, the actual value of $\text{batch_size} = 54$ when rounded. The following are the parameters and accuracy rates for the 8 models:

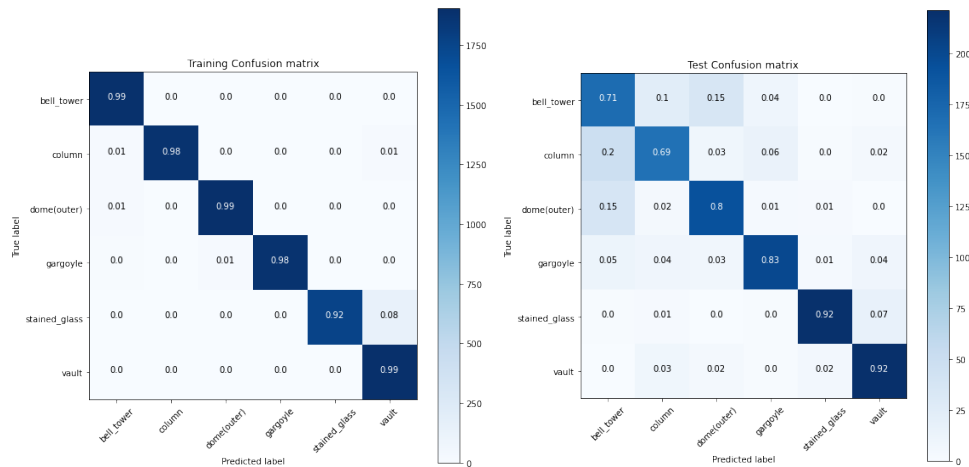
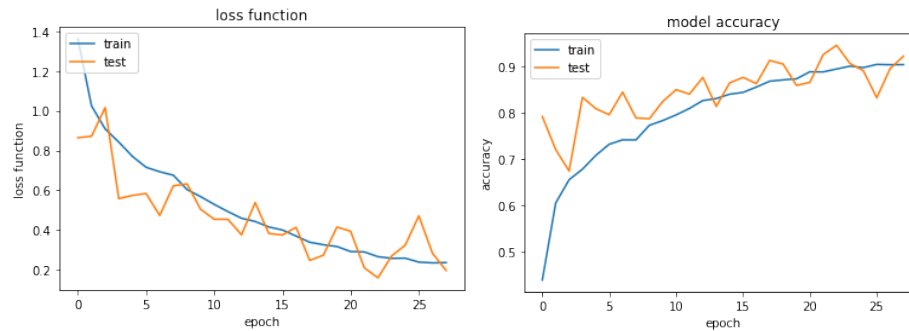
Model 1	Channel 1: 32 Channel 2: 32	h: 40	Train Loss: 0.03 Training Acc: 0.99	Test Lost: 1.22 Test Accuracy: 0.79
Model 2	Channel 1: 32 Channel 2: 64	h: 56	Test Lost: 0.02 Test Accuracy: 0.99	Test Lost: 1.21 Test Accuracy: 0.81
Model 3	Channel 1: 64 Channel 2: 64	h: 56	Train Loss: 0.03 Training Acc: 0.99	Test Lost: 1.54 Test Accuracy: 0.80
Model 4	Channel 1: 32 Channel 2: 32	h: 9	Train Loss: 0.05 Training Acc: 0.99	Test Lost: 1.78 Test Accuracy: 0.75
Model 5	Channel 1: 32 Channel 2: 64	h: 3	Test Lost: 1.82 Test Accuracy: 0.17	Test Lost: 1.82 Test Accuracy: 0.17
Model 6	Channel 1: 64 Channel 2: 64	h: 2	Test Lost: 1.82 Test Accuracy: 0.17	Test Lost: 1.82 Test Accuracy: 0.17
Model 7	Channel 1: 32 Channel 2: 64	h: 6	Test Lost: 0.88 Test Accuracy: 0.57	Test Lost: 1.51 Test Accuracy: 0.43
Model 8	Channel 1: 64 Channel 2: 64	h: 6	Test Lost: 0.11 Test Accuracy: 1.51	Test Lost: 4.38 Test Accuracy: 0.65

As the above table indicates, the 2 best models are those that utilized the dimension of the flattening layer to establish the h-value. We proceed to utilize the EarlyStopping() callback to determine the best epoch to stop learning (m*).

Model 2 Outputs after Using EarlyStopping():

Layer (type)	Output Shape	Param #
conv2d_32 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_32 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_33 (Conv2D)	(None, 29, 29, 64)	18496
max_pooling2d_33 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten_16 (Flatten)	(None, 12544)	0
dense_32 (Dense)	(None, 56)	702520
dropout_10 (Dropout)	(None, 56)	0
dense_33 (Dense)	(None, 6)	342

=====
 Total params: 722,254
 Trainable params: 722,254
 Non-trainable params: 0



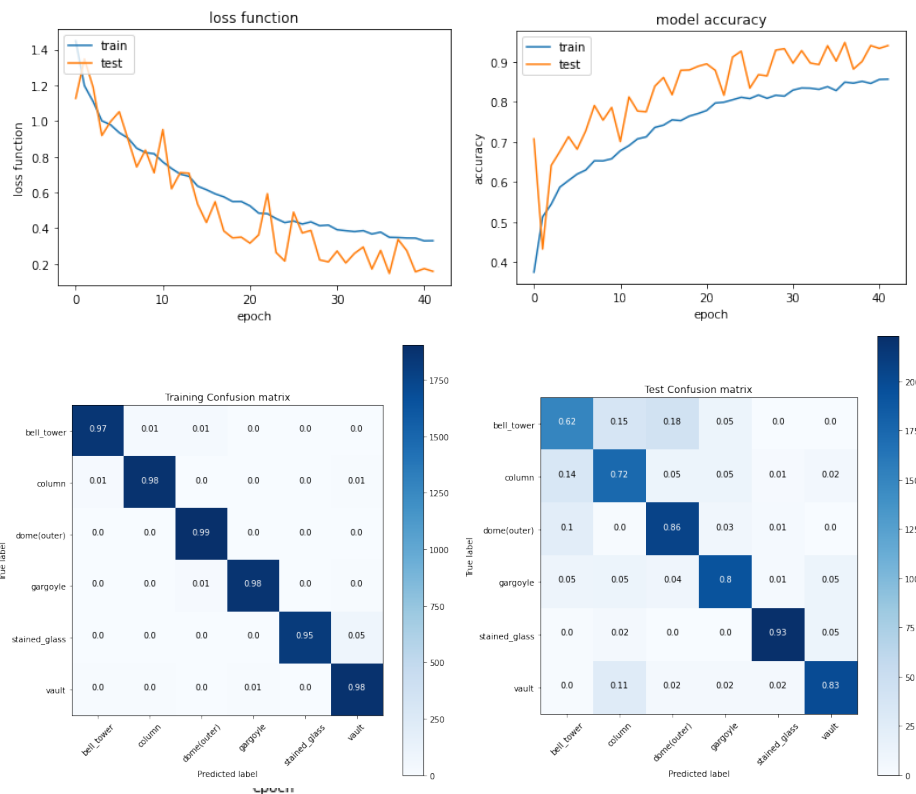
359/359 [=====] - 3s 7ms/step - loss: 0.0775 - accuracy: 0.9763
 Training loss: 0.07745140790939331 / Training accuracy: 0.976314902305603
 45/45 [=====] - 0s 7ms/step - loss: 0.7220 - accuracy: 0.8118
 Test loss: 0.7219769954681396 / Test accuracy: 0.8118055462837219

After having utilized EarlyStopping() based on validation loss and permitting a patience level of 5, Model 2 found the best m* value to be 28 with a test accuracy of 82%.

Model 3 Outputs after Using EarlyStopping():

Layer (type)	Output Shape	Param #
conv2d_34 (Conv2D)	(None, 62, 62, 64)	1792
max_pooling2d_34 (MaxPooling2D)	(None, 31, 31, 64)	0
conv2d_35 (Conv2D)	(None, 29, 29, 64)	36928
max_pooling2d_35 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten_17 (Flatten)	(None, 12544)	0
dense_34 (Dense)	(None, 56)	702520
dropout_11 (Dropout)	(None, 56)	0
dense_35 (Dense)	(None, 6)	342

=====
 Total params: 741,582
 Trainable params: 741,582
 Non-trainable params: 0



```

359/359 [=====] - 3s 8ms/step - loss: 0.0750 - accuracy: 0.9760
Training loss: 0.07500675320625305 / Training accuracy: 0.9759665727615356
45/45 [=====] - 1s 9ms/step - loss: 0.7960 - accuracy: 0.7944
Test loss: 0.7960227727890015 / Test accuracy: 0.7944444417953491
  
```

Likewise, after utilizing EarlyStopping() with the same parameters, Model 3 found the best m^* value to be 42 with a test accuracy of 79%.

Conclusion:

Having attempted the above methods and mixture of parameter tunings, we have found that the overall best model in the experiments resulted from a CNN where the first convolutional layer contained 32 channels, had a window size of 3x3, and a max-pool layer with a pool size of 2x2. And the second

convolutional layer contained 64 channels with the rest of the parameters remaining the same. The h-value of 56 was determined by considering the dimension of the flattening layer.

One potential way to optimize the model further would be to increase the size of the dataset, one that is ideally more balanced in the classes. Increasing the dataset will increase the number of informations available. This will in return have an overall positive effect on the parsimony ratio, thus making the models more robust.

References:

Architectural Heritage Elements Image Dataset.

<https://datahub.io/dataset/architectural-heritage-elements-image-dataset>