MACHINE LEARNING
CAPSTONE PROJECT

Davi Sales Barreira
September 16th, 2017

1. DEFINITION

PROJECT OVERVIEW AND PROBLEM STATEMENT

In the Civil Engineering field, the use of aggregates is very common practice in many situations. Understanding their properties is therefore important due to possible effects that these may cause. One example is in the production of concrete asphalt, in which aggregates superficial texture are known to affect the stiffness of the concrete asphalt (Singh, Zaman e Commuri, 2012) and to be related with it's Flow Number (Pazos, 2015), which is a parameter used to measure the rutting potential of asphalt concrete mixtures (Wisconsin Highway Research Program, 2013).

Measuring aggregates superficial texture is no trivial matter. In Brazil, there are still no standardized procedure for evaluating such property. To solve this problem, researchers have come up with different techniques. One of such techniques was developed by Masad (2005) and uses an equipment called Aggregate Image Measurement System (AIMS). This equipment captures high quality images of the aggregates surface and utilizes wavelet analysis to measure a *Texture Index* (MASAD, 2005). Such index is a continuous variable that is used to classify aggregates as shown in the table below:

Texture Index	Aggregate Classification
0 - 165	Polished
165 – 275	Smooth
275 – 350	Low Roughness
350 – 460	Moderate Roughness
> 460	High Roughness

Table 1 – Aggregate texture classification (MASAD, 2005)

Although the use of the AIMS equipment may solve the problem of measuring superficial texture for aggregates, such method requires researchers to have access to such equipment and to specialized technicians trained to operate it. Therefore, simpler methods for obtaining such Texture Index are imperative.

Convolutional Neural Networks (CNN) have been extensively used with regards to image analysis. Therefore, the aim of this project is to build a CNN that estimates the Texture Index of aggregates using their images. The Texture Index predicted is the same obtained through the use of AIMS, and the goal is to replicate the results obtained when using this equipment.

For sake of clarity, although the aggregates can be classified in different categories, this project concerns a regression task. The CNN model will predict the Texture Index, a continuous variable, and not the aggregate category.

2. METRICS

Two evaluation metrics were used. The first one was the R-Squared metric. This metric determines how well the model “explains” the variation in the data compared to using the mean. The goal is to obtain an R-Squared closer to 1 as possible.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2}$$

$y_i = Real\ texture\ index$
$\hat{y}_i = Predicted\ texture\ index$
$\bar{y} = Mean\ texture\ index$

The second metric was the mean absolute error (MAE) of the model. While the R-Squared is a relative metric (performance compared to using the mean), the MAE is an absolute metric and it’s magnitude is evaluated to see if the error in the model is small enough to give reliable results. It was considered that a MAE smaller than 75 was reliable, since this is the range size of the smallest classification bucket (Low Roughness, texture surface goes from 275 to 350).

$$MAE = \frac{\sum_{i=1}^{n} |(y_i - \hat{y})|}{n}$$

2. ANALYSIS

DATASETS AND VISUALIZATION

The dataset was provided by the Department of Transport Engineering (DET) of the Federal University of Ceara (UFC). It consists of 1425 images of aggregates (Figure 1) with their respective Texture Indexes. The Texture Indexes were obtained using the AIMS equipment. The aggregates analyzed were gathered from three quarries located in the state of Ceara, Brazil. The CNN model used the images as input and the output is the Texture Index.

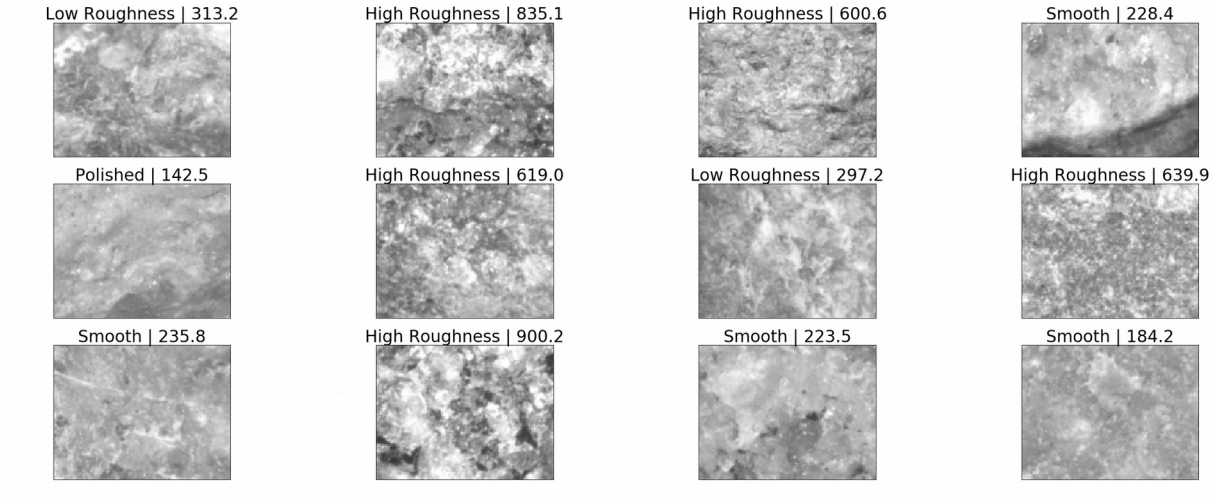

Figure 1 – Aggregates superficial texture

The original images are 480 by 640 pixels with grayscale. For the model, the number of pixels in each image was decreased, and the effects of image quality were evaluated. The distribution among each texture category present in the dataset is shown in Figure 2, while Figure 3 shows the distribution of the Texture Index.

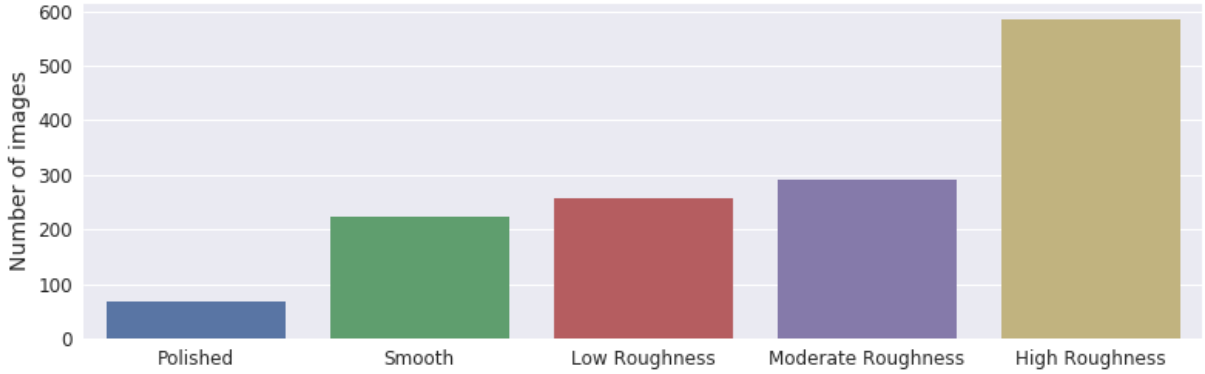

Figure 2 – Texture categories distribution

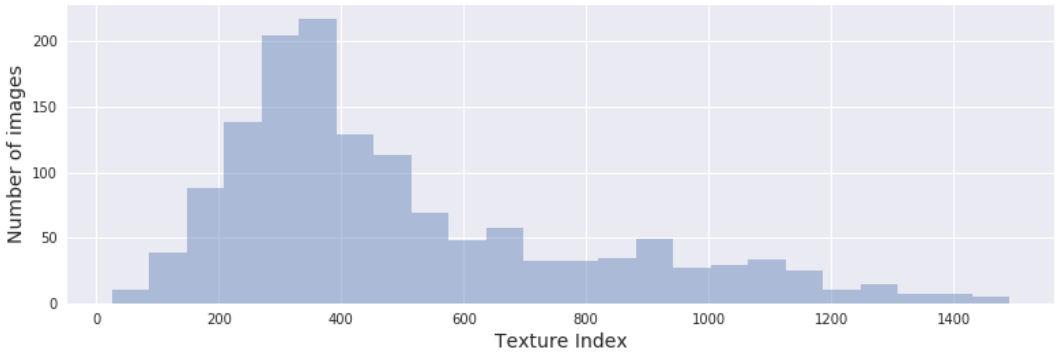

Figure 3 – Texture Index distribution

As can be seen, the majority of the images are of High Roughness aggregates, but inside such category, the Texture Index varies in a wide range.

ALGORITHMS AND TECHNIQUES

The Convolutional Neural Networks model was created using the Keras library, with Tensorflow as backend. This algorithm is a type of Neural Network in which spatial information is taken into account by the use of Convolutional layers, which makes it very suitable for analyzing images, since not only the value of each pixel is important, but also their spatial distribution.

The architecture of a CNN may present several different types of layers. The ones that were used in this project are briefly described below:

- Convolutional layers are responsible for learning spatial information. In this type of layer, a filter slides through the image to generate the values in the nodes of the hidden layer. These filters represent features of the images that are learned by the model during the training process;

- Pooling layers usually take Convolutional layers as input. Their main purpose is to reduce the dimensionality of the model, therefore, avoiding problems such as overfitting. A common type of Pooling layer is the Max Pooling layer, which samples the input and takes only the maximum value. The size of each sample is called the pooling size;

- Dropout layers are also used for preventing overfitting. The main idea here consists of randomly "shutting down" some nodes, so different parts of the Network are trained. This allows the CNN to train most of the nodes, therefore, building redundancies and avoiding "overtraining" some of them while "undertraining" others. The probability of "shutting down" a node is a hyperparameter that must be chosen;

- Fully-connected layers are connected to all nodes in the input, instead of only to a portion of them, as in the case of Convolutional layers. These are the same type of hidden layers used in regular Neural Networks.

BENCHMARK MODEL

There are no approximate methods to obtain Texture Index, only using the AIMS equipment. Therefore, the benchmark used was a simple Neural Network consisting of only one hidden layer and one Dropout layer. The hidden layer contained 1000 nodes and the dropout probability was 20%.

3. METHODOLOGY

DATA PREPROCESSING

A couple of steps must be done to adjust the format of the original data, as present in the "DataPreparation" Notebook. Some of these steps consist of reducing the quality of the images, saving them as *numpy* arrays, adjusting the format of the original file with the Texture Index values and saving a new *CSV* file with the proper format to be used in Keras. The original images were reduced to 20%, 10% and 30% of the original size, as shown in Figure 4. Also, some of the images were errors (Figure 5) and were removed from the original sample.

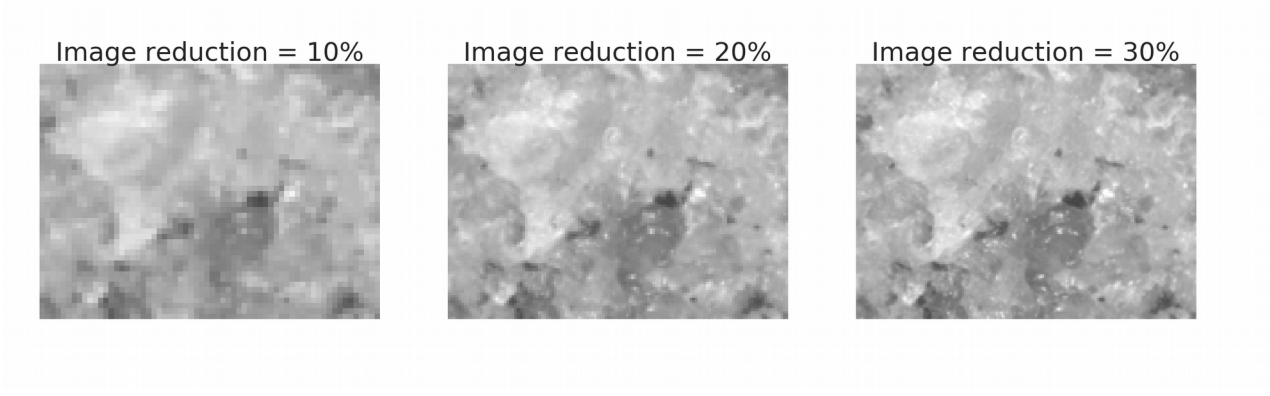

Figure 4 – Image quality example

Due to the dataset not being very large, data augmentation was implemented by flipping the images 180º. Since the Texture Index must be invariant to rotation, such method seemed appropriate.

After adjusting the data format and increasing the sample size, the pixel values were normalized by dividing them by 255. The dataset was then split into a training set, a validation set and a test set, each containing 64%, 16% and 20% of the data respectively.

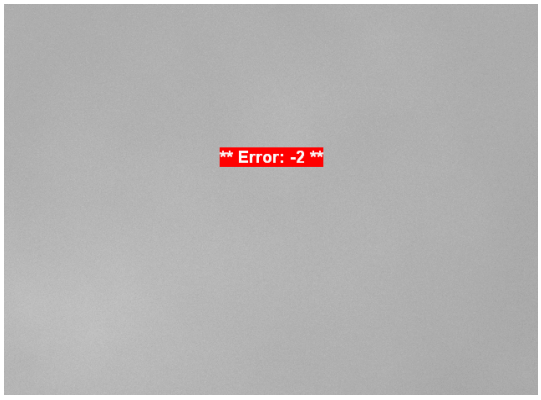

Figure 5 – Error image

IMPLEMENTATION AND REFINEMENT

After preparing the data, the workflow consisted of three main steps:

1. Creating and training the initial CNN models;
2. Improving the CNN model by applying piecemeal changes;
3. Evaluating the effect of image quality in the final CNN model.

In the first step, the baseline model consisting of a simple Neural Network was developed in Keras. After that, an initial CNN model was created based on an image classifier architecture[1]. Such model was adjusted for regression instead of classification. This adjustment consisted of removing the final *softmax* layer. Figure 6 shows the model structure implemented in Keras.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 96, 128, 48)	816
max_pooling2d_1 (MaxPooling2	(None, 48, 64, 48)	0
conv2d_2 (Conv2D)	(None, 48, 64, 96)	18528
max_pooling2d_2 (MaxPooling2	(None, 24, 32, 96)	0
conv2d_3 (Conv2D)	(None, 24, 32, 192)	73920
max_pooling2d_3 (MaxPooling2	(None, 12, 16, 192)	0
dropout_1 (Dropout)	(None, 12, 16, 192)	0
flatten_1 (Flatten)	(None, 36864)	0
dense_1 (Dense)	(None, 500)	18432500
dropout_2 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 1)	501

Figure 6 – CNN Model Structure in Keras

1 www.github.com/udacity/aind2-cnn/blob/master/cifar10-classification/cifar10_cnn.ipynb

These models were then trained using a batch gradient descent algorithm with a batch size of 30, and 20 epochs. The model improvement was evaluated against the validation set using the R-Squared as metric, and the mean squared error as loss function. The optimal set of weights were saved, and the models were then compared to the test set in terms of R-Squared and MAE.

After training the baseline model and the initial CNN, some small changes were sequentially applied to this CNN model, which consisted in changing it's architecture and hyperparameters. After each change, the model was again trained and assessed against the validation set. If a change produced an improvement compared to the initial model, it was kept and built upon, otherwise it was dropped. The different steps tried during this process of finding an optimal model are shown in Table 2. The best one was then chosen and compared with the test set to check if it actually performed better then the baseline Neural Network and the initial CNN.

Step applied	**Impact on the model**
Changed the kernel size from 2 to 3	Worsened MAE and R-Squared
Increased filter size of all Convolutional layers by 2	Didn't improve the model
Added another Convolutional layer with 192 filters	Didn't improve the model
Increased Fully-connected layer from 500 nodes to 800	Improved MAE
Increased Fully-connected layer from 800 nodes to 1000	Improved MAE
Added another Fully-connected layer with 1000 nodes	Improved MAE and R-Squared
Added another Fully-connected layer with 500 nodes	Improved MAE
Increased final Fully-connected layer from 500 nodes to 1000	Didn't improve the model
Added another Fully-connected layer with 800 nodes	Didn't improve the model
Changed pooling size of Pooling layer from 2 to 3	Worsened MAE and R-Squared
Changed Dropout probabilities to 0.2	Improved MAE and R-Squared

Table 2 – CNN architecture, changes applied

Steps 1 and 2 were performed using the pictures with 20% of the original size. This intermediate quality was chosen so it wouldn't take too long to train, since many different models were tried. Also, the images would still present a fairly high quality compared to the "10%" reduction, as observable in Figure 4.

After choosing the best CNN model, the effect of image quality was analyzed. This same model was trained in the images with 30% and 10% of the original size. Again, the models were assessed against their respective test sets in terms of R-Squared and MAE. Their results were then compared between each other to better understand the impact of image quality in the prediction of the Texture Index.

4. RESULTS

The final CNN model (Figure 7) presented the following architecture and hyperparemeters:

1. Three Convolutional layers, with kernel size of 2, stride of 1, and "same" padding. The first layer has 48 filters, the second has 96, and the third has 192. The activation function is the "ReLu" function;
2. Three Pooling layers using "Max Pooling", with a pooling size of 2;

3. Three Fully-connected layers, with the first two having 1000 nodes, and the third one having 500 nodes. The activation function is also a “ReLu” function;
4. Two Dropout layers, each with a 0.2 dropout probability.

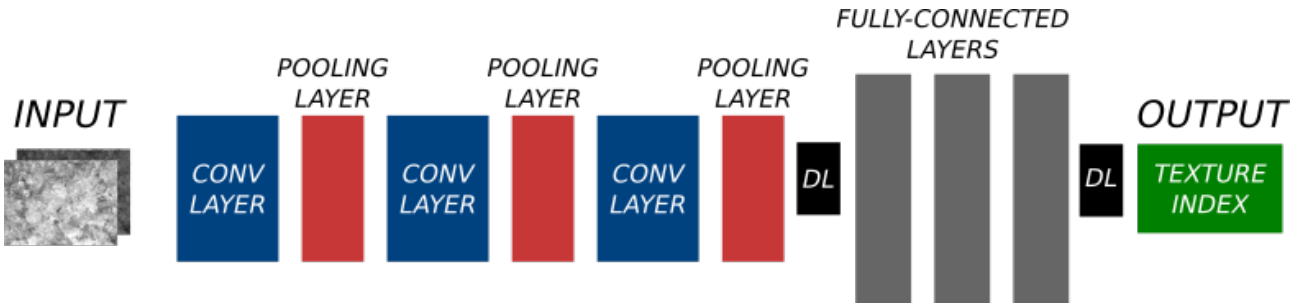

Figure 7 – Final CNN architecture

The evaluation of this CNN was done by comparing it’s results with the ones obtained through the baseline model. As expected, the simple Neural Network wasn’t able to predict accurately the Texture Index. As shown in Figure 8, this model mostly predicted a Texture Index around the mean of the test set, resulting in an R-Sqaured of 0.03 and a MAE of 235.6.

The initial CNN performed significantly better than the baseline model, with an R-Squared of 0.988 and MAE of 23.33. The final CNN was able to achieve an R-Squared of 0.994, and a MAE of 18.3. Figure 9 shows the distribution of the predicted results of the final CNN against the test data.

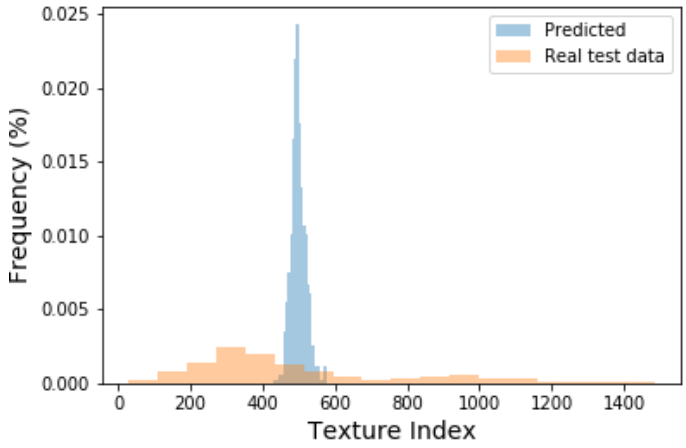

Figure 8 – Baseline Neural Network prediction against real test data

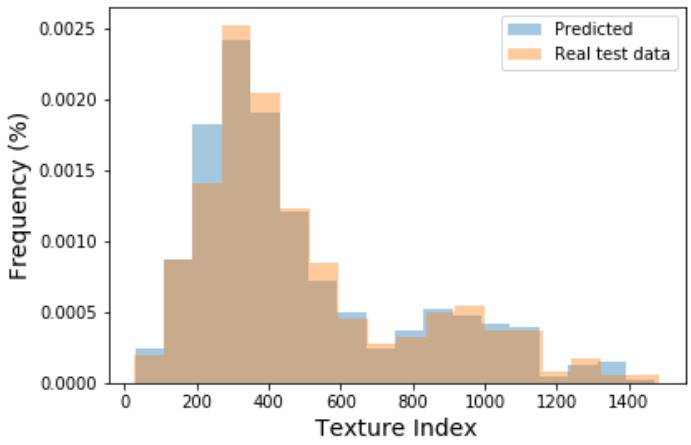

Figure 9 – Final CNN model prediction against real test data

When plotting the residuals[2] against the Texture Index (Figure 10), it's noticeable that the model absolute error increases with the Texture Index. Therefore, the model performs better for smaller textures, and gradually gets worse. On the bright side, the classification of aggregates in different texture categories just goes up to a Texture Index of 460 (see Table 1), so the model actually performs better in the region where the categorical classification falls.

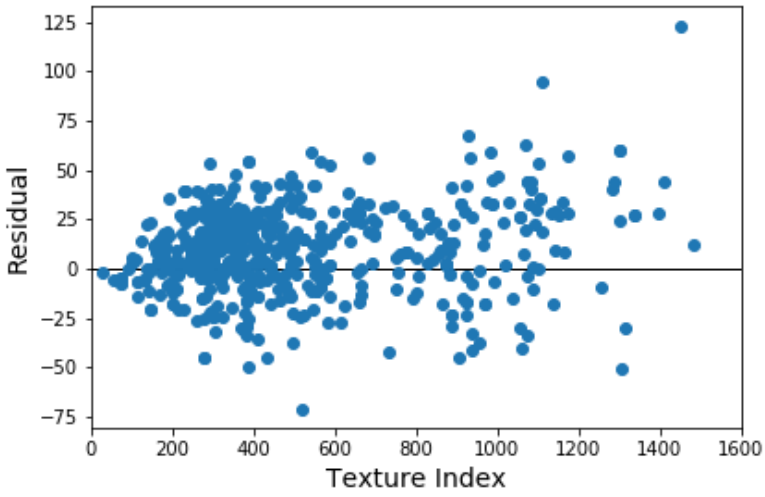

Figure 10 – Residuals plot

Finally, the CNN model was then trained with the images of different sizes. Figure 11 compares the evaluation metrics obtained for the images with each image quality. As expected, the image with only 10% of the size performs worse than the other two. The improvement is not as large from 20% to 30% as it is from 10% to 20%. Such finding may imply that using the image with full size might not give a much better result than the ones already obtained, as the evaluation metrics seem to be getting into a plateau.

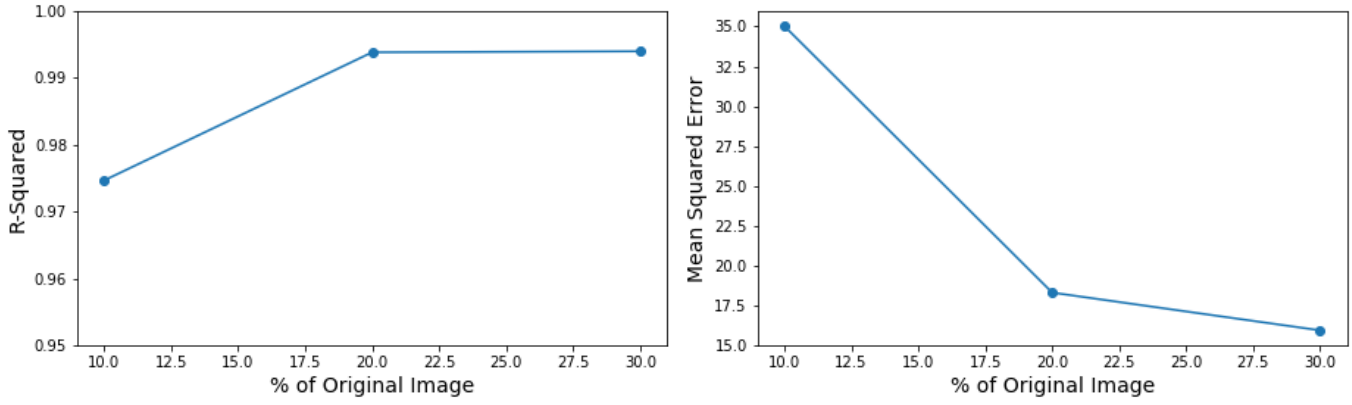

Figure 11 – Evaluation metrics for different image qualities

2 Residual = Real value – Predicted value

5. CONCLUSION

REFLECTION

This project main objective consisted of developing a CNN model to predict the Texture Index of aggregates. Several activities were undertaken to finally achieve this goal:

1. Data collection and preparation;
2. Create baseline model and initial CNN, and train the models;
3. Apply changes to improve the initial CNN and evaluate the results;
4. Train the model for images of different quality and evaluate the impact in the model performance.

From the results obtained, I consider that the project was successful in achieving it's goal. For the images with 20% of the original size, the CNN was able to get an R-Squared of 0.994 and a Mean Absolute Error of 18.3, which was better than the threshold of 75 that was established. In terms of image quality, it was shown that it can impact the model accuracy, especially in terms of MAE. Although, changing the images from 20% of the original size to 30% didn't seem to improve much the results. Such findings are encouraging, since good results can be obtained even if the images do not have a very high definition.

In the beginning of the project, I started developing in Tensorflow, which proved to be much harder than I had expected. After changing to Keras, the whole process of developing the CNN became a lot easier. During the stage of improving the original CNN model, I realized that this would take a very long time, since I was doing all the training in my personal notebook. I then decided to try Floyhub[3], which was very helpful since it gave me much more speed during this tinkering stage.

During this project, one of the most challenging parts was the data preparation. The original data wasn't properly organized and a lot of coding was necessary to relate each image with it's Texture Index. Another challenging step was learning about the problem that I was trying to solve, since it was quite technical and I didn't know much about the topic beforehand.

IMPROVEMENT

Although I consider that this project was successful, there is still a lot of room for improvement. First, the dataset was quite small compared to other datasets used for training CNN, such as the CIFAR-10 dataset. Therefore, collecting more data could definitely improve the model reliability. Also, all the images that were used were obtained through the AIMS equipment. It would be interesting to take pictures using a different camera, to check if the model could still achieve such good performance. This would also expand the usage of the model, "liberating" it from the need of the AIMS equipment, since researchers would be able to take their own pictures of the aggregates and use them to estimate the Texture Index.

Finally, instead of tinkering with the original CNN model to get to a better model, some Transfer Learning could be used to adapt successful preexisting CNN models, and checking which ones perform better for this type of problem.

3 https://www.floydhub.com/

FREE-FORM VISUALIZATION

Figure 12 shows a small sample of images, the real Texture Index and the one predicted with the CNN.

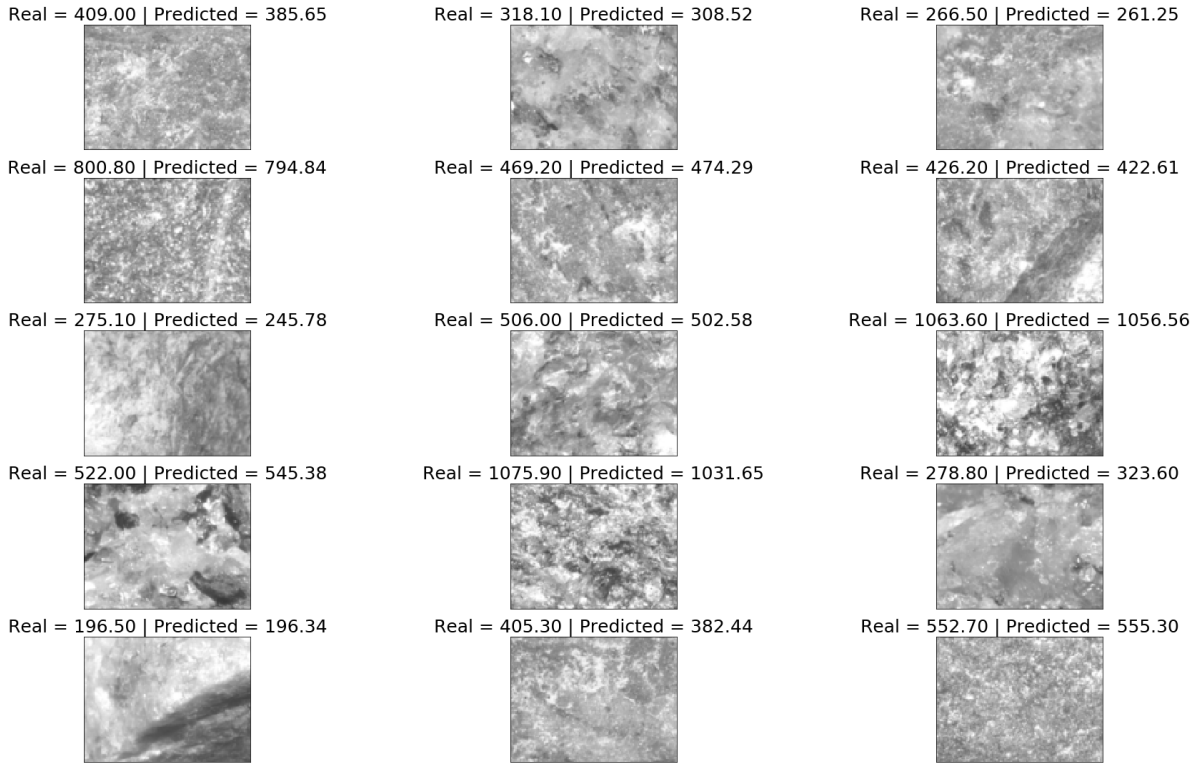

Figure 12 – Visualizing the models prediction

REFERENCES

MASAD, E. A. Aggregate Imaging Measurement System (AIMS): Basics and Applications. Texas Transportation Institute. The Texas A&M University System. Project performed in cooperation with the Texas Department of Transportation and the Federal Highway Administration. Report no FHWA/TX-05/5-1707-01-1, 2005.

PAZOS, A. G. Efeitos de Propriedades Morfológicas de Agregados no Comportamento Mecânico de Misturas Asfálticas. Dissertação (Mestrado em Engenharia Civil) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015.

SINGH, D.; ZAMAN, M.; COMMURI, S. Inclusion of Aggregate Angularity, Texture, and Form in Estimating Dynamic Modulus of Asphalt Mixes. Road Materials and Pavement Design, v. 13, n. 2, p. 327-344, 2012.

WISCONSIN HIGHWAY RESEARCH PROGRAM. Flow Number as a Discriminating HMA Mixture Property - No. 0092-09-0. Wisconsin, 2013.